Simulations of microscopic traffic models

Lorenzo Sani

October 29, 2021

Abstract

This short relation aims to present the results of the simulation of some different traffic models. The simulated system was chosen to be as simple as possible in order to understand the differences between the models on modelling common phenomena occurring in traffic flow.

1 Introduction

The topics of urban transportation networks and traffic flow dynamics were extensively developed during the last five decades. Many mathematical models were studied to determine and forecast all parameters of a transportation network such as traffic intensity over all network elements, volumes of traffic, mean traffic speeds, delays and time losses, and so on. Nowadays the literature is full of models of any kind for the traffic flow dynamics trying to adopt the newest discoveries of mathematical modelling to modern problems in urban transportation.

Traffic flow dynamics is the general topic of interest in the following treatment. All these models could be generally classified in macroscopic and microscopic models. Macroscopic models usually exploit statistical mechanics laws to understand the properties of the traffic flow dynamics by unfolding in time systems' macroscopic features such as kinetic energy, potential, linear density, or entropy [1]. Microscopic models on the other hand usually define some general rules all the elements have to respect, and then the system is let to evolve freely showing the overall behaviour. In this analysis the focus is on traffic flow dynamics models, specifically only different microscopic models are compared. The differences in the overall behaviours are pointed out and then discussed in terms of the differences between the general rules imposed.

2 Models

First microscopic models were proposed in the 1950's. They all have a common organization structure. Every car is numerated by the subscript n, according to its order on the road, that is assumed to be a (closed) line of constant length. It is assumed that the acceleration, and then speed and position, of the n-th car is defined by the states of the neighbour cars, the most important effect being produced by the immediately preceding car $n-1$. This car is often called the *leader* and then the entire class of micromodels is often called *"follow-the-leader"* models [2]. The coordinate and speed of the car n will be denoted, respectively, by x_n and v_n. In most of the cases these are continuous models.

The factors that mostly define acceleration of the nth car are:

1. speed of the given car w.r.t. the leader, $\Delta v_n(t) = v_n(t) - v_{n-1}(t)$;
2. intrinsic car speed $v_n(t)$, defining the safe interval of movement;
3. distance to the leader $d_n(t) = x_{n-1}(t) - x_n(t)$ or, as a variant in some models, the "pure" distance $s_n(t) = d_n(t) - l_{n-1}$, taking into account the car length l_n.

Correspondingly, in the general form the movement of cars obeys a system of ordinary differential equations:

$$\dot{v}_n(t) = f(v_n, \Delta v_n, d_n). \tag{1}$$

Different models provide different f functions.

Another class of micromodels, that differs from "*follow-the-leader*" model class, is the Cellular Automata (CA) class. The main difference, that is important to underline before a more detailed treatment, is that these models are often discrete in space and in time.

All the continuous models used are provided with both Euler scheme and Runge-Kutta-2 scheme for the time evolution. The discrete one is implemented as it is. In the following, the models used will be discussed in detail.

2.1 Follow-the-Leader (FTL) Model

In the very first implementations of the follow-the-leader mechanism, it was assumed that each driver is able to adapt its speed to that of the leader. The specific model used defines two different regimes of adaptation depending on the distance d_n and the computed safety distance $d_s = d_\tau + d_{min}$. Here $d_\tau = v_n(t) \cdot \tau$ is the distance the car would cover keeping constant its velocity, τ is the time step, d_{min} is a constant minimal distance whose value is chosen consider the cars' mean size and a realistic value for the distance between cars in a traffic jam.

The differential equations, for the two regimes, are then given by:

1. for $d_n \leq d_s$, $\dot{v}_n(t) = \alpha_l \cdot (d_n - d_s)$;

2. for $d_n > d_s$, $\dot{v}_n(t) = \alpha_l \cdot [v_{n-1}(t) - v_n(t)]$;

where $1/\alpha_l$ is a positive constant modelling the characteristic time of adaptation. The first regime represent the "free motion" of cars while the latter the car following regime.

2.2 Optimal Speed Model

Another class of models, similar from a mathematical point of view to the latter, are those based on the assumption that the drivers want to approach their speed a desired value, here called v_{max}. Using this point of view, every driver has a "safe" speed v_s which depends on the distance to the leader through $v_s = \frac{1}{\tau}(d_n - d_{min})$. Here, again, τ is the time step, d_{min} is a constant minimal distance computed similarly to FTL model one. These are called Optimal Speed models [3].

In this implementation, the regimes of acceleration are three. Again the fundamental quantities are d_n and d_s, and their difference defines the regime to adopt and then the following differential equations:

1. for $d_n \gg d_s$, $\dot{v}_n(t) = \alpha_o \cdot [v_{max} - v_n(t)]$,
2. for $d_n > d_s$, $\dot{v}_n(t) = \alpha_o \cdot [v_{n-1}(t) - v_n(t)]$,
3. for $d_n \leq d_s$, $\dot{v}_n(t) = 5\alpha_o \cdot [v_s(t) - v_n(t)]$.

As mentioned before, $1/\alpha_o$ is a positive constant modelling the characteristic time of adaptation. In this formulation, the first regime represent the "free motion" of a car, the second the car following modality, while the last the optimal distance motion. The difference between the conditions of the first two regimes has to be numerically defined in the implementation.

2.3 Cellular Automata (CA) Model

The Cellular Automata (CA) concept was firstly introduced by von Neumann in the 1950's in connection with the abstract theory of self-reproducing computers. By CA is meant an idealized representation of a physical system with discrete time and space and the system elements having a discrete set of feasible states.

In these kind of models the car coordinate and speed, as well as time, are discrete variables. The road line is decomposed into conventional "cells" of identical length Δx, each cell being at any time instant either occupied by a single car or empty. The states of all cells are simultaneously updated according to a certain set of rules at each time step. The specific set of rules defines the specific CA model. The used formulation is a slightly modified version of the original proposed by Nagel-Schreckberg [4]. The speed of the car is also discretized and can assume values from 0 to v_{max}, the unitary quantization step Δv is defined.

The state of all cars in the system is updated at each step $t \longrightarrow t+1$ according to the following rules:

Step 1: *Acceleration.* If $v_n < v_{max}$, then the speed of the nth car is increased by Δv, if $v_n = v_{max}$, it remains unchanged:

$$v_n \longrightarrow \min(v_n + \Delta v, v_{max}). \tag{2}$$

Step 2: *Deceleration.* If $d_n \leq v_n$, then the speed of the nth car is reduced to $d_n - 1$:

$$v_n \longrightarrow \min(v_n, d_n - 1). \tag{3}$$

Step 3: *Random perturbations.* If $v_n > 0$, then the speed of the nth car can be reduced by Δv with probability p; speed remains unchanged if $v_n = 0$:

$$v_n \xrightarrow{p} \max(v_n - \Delta v, 0). \tag{4}$$

Step 4: *Movement.* Each car moves ahead by the number of cells corresponding to its new speed

$$x_n \longrightarrow x_n + v_n. \tag{5}$$

Step 1 reflects the general tendency of the drivers to move as fast as possible without exceeding the maximum admissible speed. Step 2 ensures that crashes will not occur. Step 3 introduces an element of stochasticity, and could be modelled for taking into consideration the differences in drivers' behaviour, that are more or less aggressive/reactive.

3 Implementation

The goal of this study is to simulate a system of cars running on a closed line route, a circle. It is then produced a simple `python` program with a friendly UI (TODO: cite repo) that allows the user to choose the model to simulate, and to tune some of the model parameters such the number of cars N in the system, the radius of the route R, and the initial filling percentage of the route line. The UI shows also the graph of the velocities of the cars and the linear density on the route line in terms of distance between two following cars. A part of the UI is meant to show the evolution of cars in a 3D grid. A useful help menu is provided to explain some of the features and the available actions. An example of the UI is shown by Fig.1. The libraries required to run the program are: `pyqtgraph`, `scipy`, `PyQt5`, `PyOpenGL`, `PyOpenGL_accelerate`, and their dependencies.

A necessary step before implementing any model in `python` code is to re-scale the space and time variables of the system. The space scale was chosen to reflect the intrinsic properties of the system: the cars' size and the minimum distance between

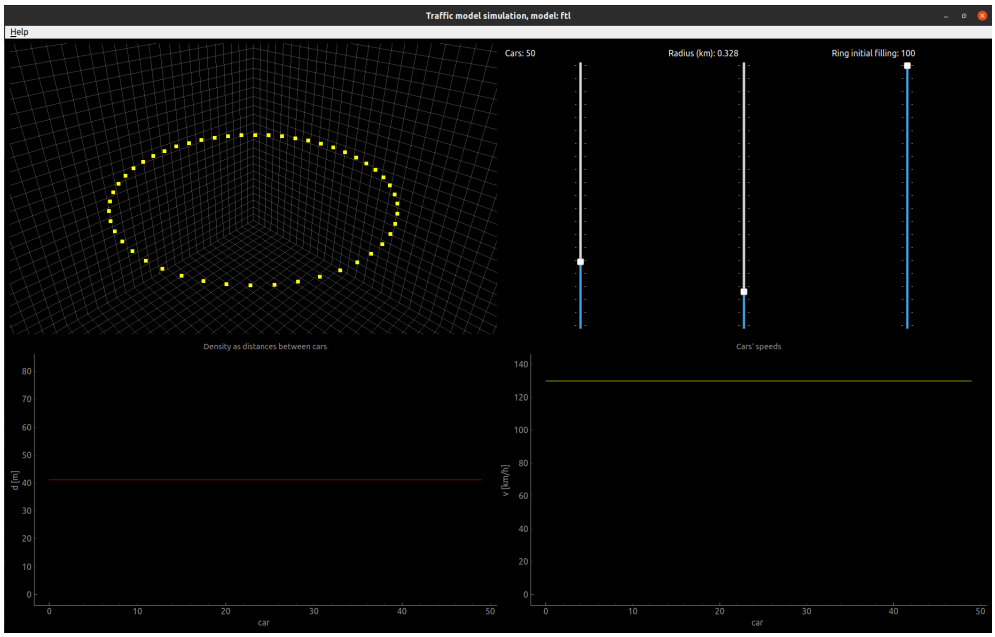

Figure 1: An example of the UI of the *python* program

two following cars, i.e. $\tilde{x} = l_c + d_{min}$, where $l_c = 4$m is the cars' size, and $d_{min} = 1$m is the minimum distance between two following cars. The time scale was chosen to reflect the update interval of the UI, $\tau = 1$s. Another space re-scaling is performed to map the re-scaled space units to UI units in order to draw the cars easily. These parameters are defined, and then could be modified, in `baselines.py`.

Every model is implemented as discussed above, with a slight modification only for the CA model. Since CA model is discretized it is necessary to subdivide in unit cells the route domain. Moreover to make effective and consistent the rules (1)-(3) with the movement rule (4), at the movement step the velocity is multiplied by the time scale τ. The user can always choose the scheme to use, or Euler or Runge-Kutta-2, the latter is the default.

Once the simulation starts the desired number of cars is equally spaced along the route line. Any time the user can modify the parameters using the sliders, this modification will restart the simulation in the same window. It is possible to send some commands to the simulation after the focus has been put, clicking on it, on the pane where cars are moving. The available commands are:

Pause/Resume: by pressing keyboard button "*Space*" the simulation will be put on pause when running, or will resume when on pause.

Kill: by pressing keyboard button "*Esc*" the simulation will close.

Dump: by pressing keyboard button "*D*" the positions, speeds, distances of the cars will be dumped to compressed `.npy` files.

Traffic Light: by pressing keyboard button "*T*" the first car will slow down simulating the presence of a traffic light.

Perturbation 1: by pressing keyboard button "*1*" an external perturbation on the speeds of all cars will applied on the system. The speed of every car, in *km/h*, is added to a random uniformly sampled integer $i \in [-5, +5]$.

Perturbation 2: by pressing keyboard button "*2*" an external perturbation on the speeds of all cars will applied on the system. The modification to speed is the same as *Perturbation 1* but multiplied by a factor 2, i.e. $i \in [-10, +10]$ in this case.

Perturbation 3: by pressing keyboard button "*3*" an external perturbation on the speeds of all cars will applied on the system. The modification to speed is the same as *Perturbation 1* but multiplied by a factor 3, i.e. $i \in [-15, +15]$ in this case.

Perturbation 4: by pressing keyboard button "*4*" an external perturbation on the speeds of all cars will applied on the system. The speed of every car will assume a random value $v \in \mathcal{U}[0, v_{max}]$, uniformly sampled.

Perturbation 5: by pressing keyboard button "*5*" an external perturbation on the speed of the first car will applied. The modification to that speed is the same as *Perturbation 1.*

Perturbation 6: by pressing keyboard button "*6*" an external perturbation on the speed of the middle car, i.e. the 50-th car if there are 100 cars in the system, will applied. The modification to that speed is the same as *Perturbation 1.*

4 Experiments

Some experiments are performed to compare these models above. There will be put the focus on the stability of such models and on which conditions they could be considered stable. In the end, the general problem of having a traffic light on a specific place of the road line will be discussed along with its effect on the stability of the system.

4.1 Stability

In order to evaluate the stability of these models, the simulation starts with the cars equally spaced on the road line having the maximum possible speed, v_{max}. By increasing the number of cars on the line, or by decreasing the radius of the circle, the system will reach a critical state at which it cannot keep the initial conditions, but one or some cars are obliged to break to avoid a crash.

Some quantities are leading this analysis, in fact the safe distance for the continuous models, i.e. TFL model and Optimal Speed model, has a minimum value $(d_s)_{min} = \tilde{x} = l_c + d_{min} = 5\text{m}$ and a maximum $(d_s)_{max} \simeq 41.1\text{m}$ depending on the speed of the current car. It is reasonable to predict the critical state to be the one at which the space between car is equal to $(d_s)_{max}$. It is also reasonable to predict that, whenever the system is initially in other states in which the initial space is lower than $(d_s)_{max}$ but higher than $(d_s)_{min}$, the system is able to achieve a "congested" stable states where cars reach a common speed lower than v_{max}. Following this line of reasoning, any initial state, in which the initial space available to each car is close to $(d_s)_{min}$, should lead to a "deeply congested" state where cars are not able to move at all. The discrete model is expected to present similar behaviours but with different mechanisms since it is lots different in nature w.r.t. to the others.

Different values for the parameters α_l, α_o were tested, precisely the values chosen were 2.0, 1.0, 0.5, 0.4, and 0.1. The stability of these models is strongly related to the time adaptation parameter.

The "congested" and "deeply congested" states mentioned before appear as predicted for every model simulated in any flavor given by the choices of the parameters α_l and α_o. The scheme used also does not make the difference. For $N = 50$, the radius of the "congested" state is $R = 0.328\text{km}$, while the radius for the "deeply congested" state is $R = 0.039\text{km}$. For $N = 100$, the radius of the "congested" state is $R = 0.655\text{km}$, while the radius for the "deeply congested" state is $R = 0.079\text{km}$.

The next step to study the stability is to perturb a little the system using the perturbation defined previously in Sec. 3 when the initial state is the "congested" one. Here the parameters α_l, α_o make the difference, and because of this FTL model, Sec. 2.1, and Optimal Speed model, Sec. 2.2, are discussed simultaneously.

For values of α_l and α_o equal to 2.0 and 1.0, any perturbation is absorbed by the system that is able, waiting a reasonable amount of time, to recover the "congested" state. When α_l and α_o are equal to 0.5 the situation does not change for "Perturbation 1-4", while for "Perturbation 5" and "Perturbation 6" a valley in the speed field is forming for the FTL model and a quite deep pit for the Optimal Speed model, as shown by Fig. 2. For values of α_l and α_o equal to 0.4, FTL model is not able to recover the "congested" state for any perturbation presenting one or more deep pit in the speed field for "Perturbation 1-4" and breaking completely down towards a "deep congested" state for "Perturbation 5-6". Optimal Speed model is able to recover "congested" state for "Perturbation 1-4", while for "Perturbation 5-6", which seem to be the most dangerous perturbations, it breaks down similarly to FTL model. For values of α_l and α_o equal to 0.1, none of the two models is able to recover the "congested" state.

The other experiments using CA model show that "congested" state is strongly unstable and not resilient to any perturbation. It is necessary to quadruple the radius of the circle to see some resilience, and only for "Perturbation 1". In the most of the situations CA model breaks up showing one or more peaks in the speed field resulting

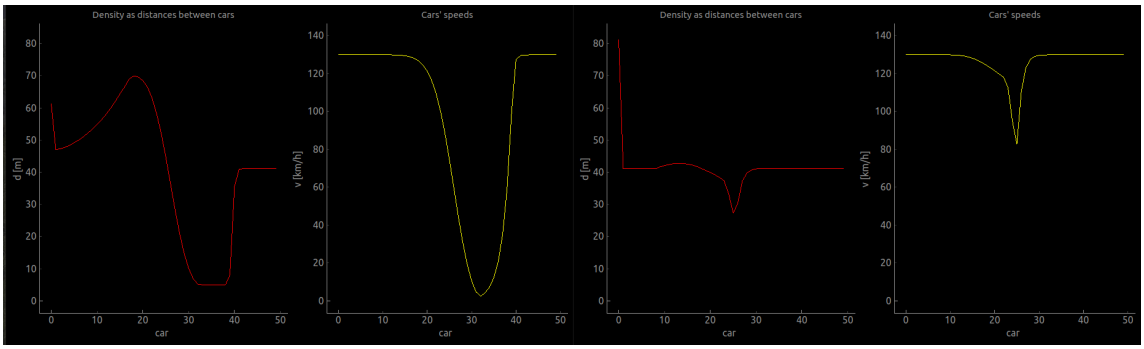

Figure 2: These are the graphs for the speed field and the distance field when "Perturbation 5" is applied to the "congested" state with $N = 50$ cars and α_l and α_o equal to 0.5. The first tow panes on the left represent the FTL model, while the latter two Optimal Speed model.

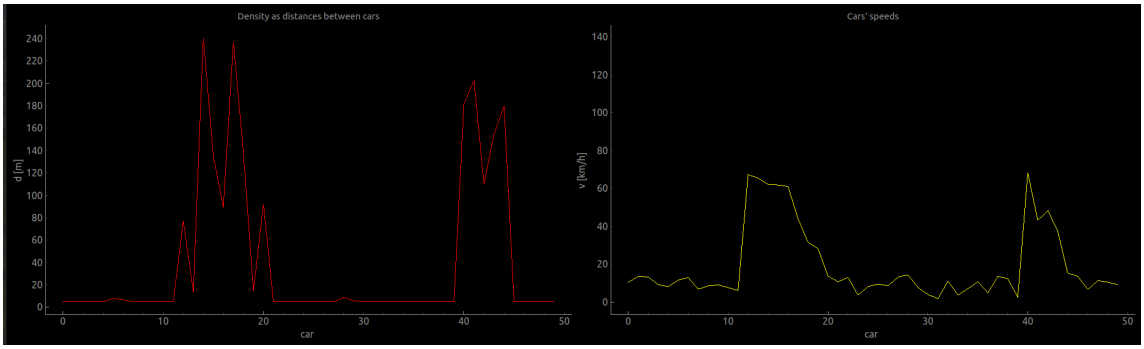

Figure 3: These are the graphs for the speed field and the distance field when "Perturbation 1" is applied to the "congested" state with $N = 50$ cars of the CA model.

in one or more zones of deep congestion. An example of this is shown by Fig. 3.

4.2 Traffic Light

One of the most common tools to control traffic flow in a transportation network is the traffic light. These are still widely used in common urban networks despite most of them have been replaced by roundabouts. Traffic light has been modeled by decreasing the maximum speed that the first car in the road line can have, and can be activate with the proper action described in Sec. 3.

In this experiment the initial state of the system is choose to be the "congested", as explained above. Then the traffic light is activated, and it is deactivated only when the speed of all cars in the system fall to zero, like Fig. 4 shows. The goal is to understand which are the conditions that allow the models to recover a stable state with all the cars having maximum speed. For these experiments the values of α_l and α_o will be set to 1.0, i.e. one of the most stable conditions.

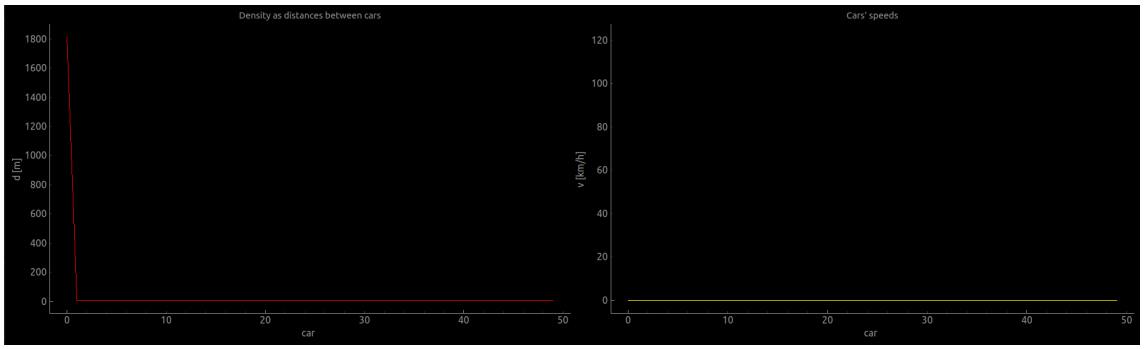

Figure 4: These graphs represent the state of the system when the traffic has been activated since some time, and also the starting state when the traffic light is deactivated.

Clearly the FTL model has to be slightly modified since the speed of the first car is dependent only on the speed of its leader. Because of this property, when the traffic light is activated FTL model is not able to start any car in the system as all the cars have speed $v_n = 0\text{km/h}\ \forall n \in [1, N]$. A simple modification to solve this issue could be adding one more regime of adaptation similar to *regime 1* of the Optimal Speed model, i.e. a free motion regime. This new model will be called "modified Follow-the-Leader" (mFTL) model in the next.

Using this configuration above, mFTL model recovers completely the initial state at the first round of the circle, while Optimal Speed model keep a zone of partial congestion in a well localized zone that slowly disappear. CA model on the other hand keep a well localized zone on deep congestion where the cars stops for a while.

5 Conclusions

The experiments shown are only a part of the possible test one can perform to compare those models. It is clear the strong difference between continuous models, namely FTL/mFTL and Optimal Speed, w.r.t. the discrete one, CA model. The similarity of FTL/mFTL and Optimal Speed is derived from the similar mathematical formulation of the acceleration function, despite they comes from different assumptions of modelization. The linear density in populating the road line appears to be a fundamental quantity to discover states of congestion on that road line for any model. Studies of perturbations show that the most resilient model is Optimal Speed followed by FTL, on the other hand CA model seem to be highly unstable when a perturbation of any kind occurs. The latter can be caused by the fact that CA model is intrinsically stochastic having intrinsic perturbations.

Traffic light simulations seem to show realistic situations and underline more differences between models. Without any modification FTL could not be able to reproduce a traffic light, because of the dependence to the leader speed in acceleration. The

first car here could stop in the middle of nowhere only because its leader, that is placed far away, stops itself. This last is not a realistic situation. CA also shows some surreal behaviours when traffic light is deactivated: many cars gain speed quickly not allowing their followers to exit the congested state. The latter might be caused by some mis-tuning in the parameters.

Other future works and research could concern modelling a system of cars with different times of adaptation, or modelling reactivity modifying the safe distance functions. In other works there was put another parameter in the continuous model to modify the influence of the leader speed in the acceleration computations, e.g. change regime for $d_n > d_s$ to $\dot{v}_n(t) = \alpha_l \cdot [(1 + \boldsymbol{\epsilon}) \cdot v_{n-1}(t) - v_n(t)]$. Adding another model or some modification to this code is very simple and straightforward.

References

[1] V. I. Shvetsov. Mathematical modeling of traffic flows. *Automation and Remote Control*, 64(11):1651–1689, Nov 2003.

[2] Elliott W. Montroll Robert E. Chandler, Robert Herman. Traffic dynamics: Studies in car following. *Operations Research 6(2)*, pages 165–184, 1958.

[3] G. F. Newell. Nonlinear effects in the dynamics of car following. *Operations Research 9(2)*, pages 209–229, 1961.

[4] Kai Nagel and Michael Schreckenberg. A cellular automaton model for freeway traffic. *Journal de Physique I*, 2:2221, 12 1992.