Fast simulation kernel for urban traffic

B. Plateau B.Ycart

April 12, 1997

LMC/IMAG, BP 53, 38041 Grenoble Cedex 9 France

Abstract

We propose a simplified model for urban traffic. Our goal is to achieve faster than real-time simulation over a realistic size model of a city network. The model can be viewed as a Markovian system, its simulation uses a harmonisation procedure.

1 Introduction

Our goal is to achieve faster than real-time simulation over a realistic size model of a city. We do not view the proposed method as the only kernel algorithm of HIPERTRANS, but rather as an extension of the main simulator, capable of certain type of real time predictions. Possible predictions bear on the traffic load and possible congestions after a change of parameters (accident, flow increase, diversion), and also estimates of mean travel time between two points of the city, depending on the current traffic situation. What this method cannot predict is the travelling time of a tagged vehicle or the variance of the travelling time along a route. When these values are required another simulation technique, using regular event lists (*cf. [12]*) should be used. The restrictions come from the type of model used for urban traffic, but this type of model is however *dynamic*, robust, guarantees fast response time and offers an easy parallelisation.

This document is organised as follows. In section 2 the model will be described. Its basic object is the notion of *section*, seen as a portion of street in one direction, between two consecutive crossings. The current state which is simulated is an array of number of vehicles of different types into the different sections. The elementary events consist of transfers of one vehicle from a section to another one, connected to the first at a crossing. The input parameters of the model consist mainly of the geography of the network, the mean traffic flows into sections and the destination probabilities.

In section 3, the algorithm is described. We start with a few basic ideas to be reminded in programming in order to minimise the running time. Some ideas about the possible data structure are given. The main loop is described informally, and the successive random choices to be made at each iteration are specified. Then we sketch the perspectives of parallelisation, which turn out to be quite reasonable. The section ends with the description of a demonstration prototype, that will visualise the execution of the algorithm over a simplified network of a Manhattan-like city map.

Section 4 gives the potentialities of the algorithm, and underlines its main limitations. Section 5 is devoted to the mathematical background. The model can be viewed as a particular case of a Markovian Petri net, and the simulation algorithm is based on a harmonisation technique.

2 The model

2.1 Street sections

The basic notion in our model is that of a *section.* It is meant as a portion of a street between two consecutive crossings, in one direction. Portions of a one-way axis are counted as one section, whereas along two-way axes sections are doubled. The geography of the network specifies the interconnections of sections inside the network. Vehicles can arrive into a section coming from some others, and they leave the section to get into different ones. Roundabouts are treated as particular sections. Entrances and exits to the network are also particular sections, with unlimited capacities. They serve as counters for numbers of vehicles entering and leaving the network. Each section ends at an intersection of one of the following three types (each with their exiting conditions).

- Roundabout: limitative conditions come from possible congestion and priority rules.
- Regular crossing: same types of conditions.
- Traffic light: it is green or red according to the current value of the time counter. The periodicity is determined by traffic control input.

Into each section, vehicles of different types may enter. The number of types is not fixed at this point but we mainly think of two types, heavy and light vehicles. Such a distinction is widely accepted in traffic models. Vehicles stay inside the section for a certain time, after which they leave the section to get into another one (except for exits where they accumulate). The occupation of the section, or *total load* is measured as a function of the numbers of vehicles of each class. As an example of such a total load function, if two types of vehicles are considered (heavy and light) one could propose to add the number of light vehicles (cars, motorbikes) plus twice the number of heavy ones (trucks, buses). Except for entrances and exits, each section has a limited capacity which is the maximal value allowed for its total load.

2.2 Configurations and events

The current state of the system which is being simulated is reduced to the number of vehicles of each type inside each section at a given instant. It could be viewed as a double array of integers indexed by the sections and by the vehicle types. This array is called a *configuration*. The current configuration changes each time one vehicle leaves a section to enter another (transfer). This is our basic event, to be simulated repeatedly in the main loop of the algorithm. A vehicle transfer is decomposed into four decisions to be taken successively. These decisions are taken according to the answers to the following four questions.

1. *Is exit possible?* The answer depends on the type of end point of the section under consideration. In the case of a traffic light, it depends on the value of the time counter. If it is a regular crossing or a roundabout with priority rules, it will depend on the total load of neighbouring sections.

2. *Is exit performed at the right frequency?* Here comes a crucial notion of our model, that of *exit rate*. The exit rate is the number of vehicles of a given type that come out of the section per time unit, as a function of the numbers of vehicles of each type which are present in this section. This function could also be called the throughput. The shape of this function is classically referred to as the "fundamental diagram" by traffic engineers. Based on actual measurements, they view it as a dome shaped curve which is interpreted as follows. When the concentration of vehicles is small, there is no particular limitation and the throughput is roughly proportional to the concentration. Therefore the first part of the curve is a straight line, the slope of which is proportional to the speed limit and the number of lanes. Due to limited capacity, the curve reaches a maximum for a certain concentration and decreases to a very low value when the maximal total load is reached (traffic jam). The decision to make an actual transfer is taken in the algorithm with a probability which is proportional to the current value of the exit rate.

3. *Which direction is chosen by the vehicle?* For each type of vehicle the possible directions are associated to probabilities of choosing them. One can imagine for instance that 80% of the traffic goes straight on, 15% turns right and 5% turns left. These probabilities may be different from one type to another (buses follow precise routes and cannot go to certain sections, main axes are preferably followed by heavy vehicles and so on).

4. *Is transfer to that direction possible?* It is not possible if the chosen section is at its maximal capacity.

After the last question has been answered positively, the transfer becomes effective, the vehicle type counter of the exit section is decremented, that of the next section is incremented.

2.3 Input parameters

We summarise here the different parameters, required as input by our model. These parameters can be provided by the user, or statistically estimated from real observations (see [4], [8]).

- Geography of the network: connection graph, types of connections (regular crossings, roundabouts, traffic lights), sizes (maximal capacities).
- Traffic control: Status of traffic lights depending on clock, priority rules, diverted axes.
- Initial state : numbers of vehicles of each type present in the sections.
- Entrance flows: average number of vehicles entering per time unit at each entrance of the network.
- Section exit rates: obtained by time sampling and direct counting. The estimation of fundamental diagrams is a routine operation for traffic engineers.
- Destination probabilities: this may be the only non-standard input of the model. The proportion of vehicles of a given type going to each direction when leaving a given section can be statistically estimated without major problem, but sampling and data collection may require a rather costly effort. These destination probabilities can also be estimated by adjustment between the real situation and simulation.

3 The algorithm

3.1 General principles

Our main objective is to achieve faster than real-time simulation over a very large network. This implies that the main loop should be made as fast as possible by

- suppressing expensive computations. This implies that functions (probabilities, exit rates) should be discretised and tabulated before entering the main loop.
- Using a powerful and fast random number generator (see [6]). We propose to use, in its assembly language version, the generator ULTRA, created by Marsaglia and Zaman [9], and proposed as shareware on the web.

In our view, the speed up in computing time should not be achieved at the expense of flexibility. For instance we believe that most variables or functions (maximal capacities, destination probabilities, exit rates, traffic lights...) should be different for each section, even though it is to be expected that many sections of the same type will have in common some of these characteristics. If the program is to be used as a predicting tool, the user needs to be able to change in real time the values of the parameters of a given section in order to observe the effects of his change. As an example, one might want to modify the destination probabilities of some consecutive sections to observe the effect of a diversion; or set some of them to zero in order to predict traffic jams caused by an accident; or change the scheduling of traffic lights in order to estimate the benefits of synchronisation.

3.2 Data structure

In an object oriented language, the sections can be represented by a class including the following functions or variables.

- Maximal capacity.
- Exit permission (depending on time counter (traffic lights) or total load of neighbours (priorities).
- Exit rates (for different types of vehicles, depending on their concentrations).
- Destination probabilities (probabilities of choosing each possible destination for different types of vehicles).

Apart from that, the data structure should be kept as simple as possible, in order to optimise memory. Some (few) global functions are needed, such as the formula returning the total load of a section as a function of the numbers of vehicles of different types.

3.3 The main loop

As already said, the main loop will consist of simulating the transfer of one vehicle from one section to another. It should be designed to run as fast as possible. Ideally, it should contain only calls to the random number generator, calls to tabulated functions, tests and integer incrementations.

We shall describe now the successive steps to be taken at each iteration of the main loop.

1. *Choose one section*, with a probability proportional to it maximal cumulated exit rate (sum of maximal exit rates over all types of vehicles). This is the most delicate step as far as running time minimisation is concerned. Since there are

very many sections, the choice cannot be made by the inversion method. One has to use a mixed decomposition-rejection method: sections with similar exit rates should be grouped together. Firstly, a group is chosen with an appropriate probability. Then the choice of one section inside the group is made uniformly, with rejection adjustments.

2. *Check if exiting that section is possible* (traffic light or priority). (If not, skip the rest of the iteration.)

3. *Choose one type of vehicle*, with a probability proportional to its maximal exit rate.

4. *Compute the actual exit rate for that type.*

5. *Decide to make or not the transfer* with a probability equal to the ratio between the actual exit rate and the maximal one. (If not, skip the rest of the iteration.)

6. *Choose one destination* according to the destination probabilities for the chosen type.

7. *Check whether the chosen destination is possible* (if its total load is below maximal capacity). (If not, skip the rest of the iteration.)

8. *Transfer one vehicle* (update the two counters for the chosen vehicle type, of the exit section and the destination).

9. *Increment the time scale* by a constant, which is the inverse of the sum of all maximal exit rates over all sections (pre-calculated of course).

3.4 Parallelisation

If k processors are available, it is a natural idea to divide the city network into k districts of equal importance, each one being treated by one of the processors. However in such a partition, a sizable number of sections will start in one of the districts and lead to one of its neighbours. Any transfer from such a section will require a synchronisation of time scales of the two corresponding processors. All these synchronisations will result in an important loss of computing time, thus jeopardising the benefits of parallelisation.

One possible way of avoiding synchronisations between neighboring districts is to isolate the districts, considering that a section which leads outside a district is an exit, whereas a section coming from another district will be treated as an entrance. Of course, one has to make sure that the numbers of vehicles of each type coming out of a given district correspond, at least on average, to the numbers of vehicles of the same type entering a neighbour through the same section. Thus, one could treat the different districts as independent networks provided the flows of vehicles of each type are coherent.

For a given set of parameters, the average flows of vehicles out of a given section are unknown and have to be estimated over the whole network.

In order to do this one can imagine a preliminary stage in which the algorithm would run sequentially over a limited number of iterations, in order to estimate first the exchange flows between districts. Then the main simulation would take these estimates as inputs for independent simulations of each district by a different processor. When a change of parameters (i.e. traffic conditions) occurs, the independent simulations exchange their flow values (output value to update input values).

3.5 Demonstration version

In order to evaluate the performances of the algorithm, we propose to implement it, with graphic visualisation of sections loads, over a simple imaginary city network. This network will be based on a square grid of M by N blocks, with regularly spaced diagonals (figure 1). At each intersection on the boundary of the grid, there is an

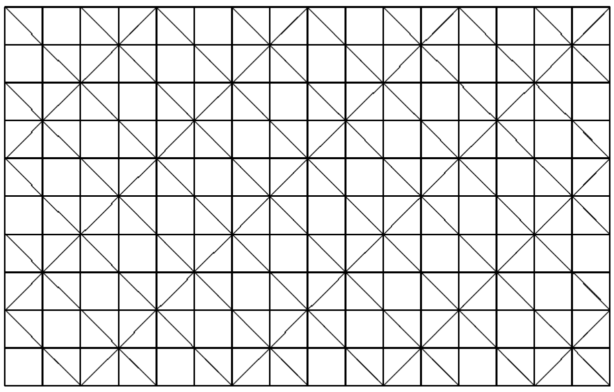

Figure 1: City map of 16×10 blocks

entrance and an exit. Inside the grid, there are crossings of 4, 6 and 8 sections. Crossings of 4 sections will be endowed with traffic lights, crossings of 6 and 8 sections are roundabouts. The boundaries are considered as priority axes, and crossings on the boundary are regular, with priority to boundary sections. The number of elements of the network depend on the dimensions N and M. There are $2(N + M + 2)$ entrances and exits and $6MN - (N + M)/2$ sections, including roundabouts.

It is important to be able to parametrise the city by its dimensions. This will permit to evaluate the evolution of performances as the size of the network increases. Thus we shall be able to check that the algorithm can actually be used on a real city. A rough evaluation on the traffic network of Grenoble ($400,000$ inhabitants) leads us to an estimate of $50,000$ sections. This corresponds to a 100×100 grid. To reach the size of the biggest capitals, one has to be able to deal with networks of about $1,000,000$

sections, which corresponds to a 500×500 grid. In order to achieve real time on such a network, it seems that the running speed should be at least $10,000$ iterations of the main loop per second CPU.

4 Expected performances

4.1 Capabilities

We view our algorithm as the kernel of a prediction and decision aid simulation program. The main feature of this program would be to visualise in faster than real-time the evolution of traffic under any set of parameters. All what concerns traffic flows (mean number of vehicles in a given part of the network, congestions, mean travel time from one point to another) can be simulated with our algorithm. Ideally, one could imagine that an interface would allow the decision maker to interact at any time with the simulation to change at his convenience the parameters (direction probabilities, exit rates...) at any section. The program would soon indicate the consequences of the modification.

The possibility of adding new types of vehicles could be used to study for instance the mean duration of projected routes for buses, or to investigate the efficiency of alternative routes on congested axes. In order to study the travel time of a particular route, one can take advantage of the data structure, and create a new type of vehicles that can only run on the route (by fixing at each crossing one probability to be equal to 1, the others being 0).

4.2 Limitations

Firstly, it is quite obvious that the main objects of interest in our model are flows of vehicles, and not the particular behaviour of one vehicle. One could imagine the following experiment. Assume one type of vehicles is separated into two types, one of them containing only one vehicle, the parameters remaining unchanged. One could easily keep track of that vehicle in the simulation and observe its trajectory in the network. That trajectory (being Markovian) will appear very irregular and unrealistic. Even though the model and the algorithm have been described in terms of individual transfers, these transfers are not meant as individual decisions by automobilists, but rather as a discretisation of the mean vehicle flow. Remember however, that the individual travel time on a prescribed route can be simulated by creating a new type of vehicles, as indicated previously.

The main limitation comes from the necessary approximations that have been made in the model. In the proposed algorithm, the speed up of the main loop has been achieved at the expense of several first order approximations. By "first order approximation", we mean replacing a probability distribution which is unknown by another probability distribution with the same (estimated) averages. The two main first order approximations concern exit rates and the incrementation of the time scale. Assuming that the process is Markovian amounts to assuming that the intervals of time between two transfers are exponentially distributed. This is probably not the case in reality. Also, as already pointed out, the time scale is incremented by integer steps instead of by exponential steps as it should be. The model is robust, *i.e.* first order quantities, such as mean numbers of vehicles in portions of the network, mean travel times, do not depend on the changes of distributions that have been made. However, the same is not true for higher order quantities such as variances or probability distributions. As an example, we believe that the mean travel time from one point to another can be predicted by our model, but the variance of the same quantity, or its quantiles cannot.

5 Mathematical background

5.1 Markovian Petri nets

Markovian Petri nets (*cf.* [1, 3]) can be seen as a very general model for synchronised queuing systems (*cf.* [10, 11, 13]). A Petri net is composed of a finite number N of *places* and a finite number T of *transitions*. In our model, places are street sections, and transitions correspond to vehicle transfers. Each place can contain a certain number of *tokens*, also called *marks*. Tokens are vehicles. If different types of vehicles are considered, one can either introduce a so called *colored Petri net*, or simply multiply places by types, i.e. a place contains the set of vehicles of a fixed type for a given section. The *marking* of the Petri net is the integer valued vector of all numbers of marks in the sections. It was called *configuration* in section 2.

The evolution of the marking is determined by the *triggering* of transitions. When transition τ is triggered, it brings marks to certain places and removes some to others. The number (positive null or negative) of marks added to (or substracted from) place n when transition τ is triggered is denoted by $c_{n\tau}$. The matrix

$$C = (c_{n\tau}) \,,\; n = 1, \ldots, N \,,\; \tau = 1, \ldots, T$$

is called the *incidence matrix* of the net. Its τ-th column, which describes the effect of transition τ on the marking, will be denoted by C_τ. In our case, a transition removes one mark from a single place and brings it to another. So that the columns of the incidence matrix contain one entry equal to -1, one equal to 1, the rest being null. Entrances and exits are particular transitions which modify only one place.

A Petri net is *timed* when a transition triggering process has been specified. Denote by Z_t the marking vector at time t. In a Markovian Petri net, the process $\{Z_t \,;\, t \geq 0\}$ is a (non homogeneous) Markov jump process with values in the set of possible markings. The jumps correspond to transitions triggering. If transition τ is triggered, the current marking changes from z to $z + C_\tau$. The rate of that jump, denoted by $\lambda_\tau(z, t)$ depends on the marking and possibly on the time scale. In our case, the transition rates for a vehicle transfer depend on the number of vehicles inside the section (through exit rates), on the times scale (in case of traffic lights), on the total load of neighbouring sections (in case of priorities), and on the total load of the destination. More precisely, the transition rate of a vehicle of a certain type from section o (origin) to section d (destination) is the product of the following quantities.

- The exit rate of section o for the considered type.
- The indicator function (one or zero) of the traffic light (if any).
- The exit probability as a function of total load of neighbouring sections in case of priority rules.
- The destination probability of section d from section o, for the considered type.
- The indicator function of the origin section (zero if it is at its maximum capacity, else one).

5.2 Harmonised Markov chain

The dependence in time of the transition rates in our model (traffic lights) is very limited. One can consider, as a first approximation, that the model is a (continuous time) homogeneous Markov jump process. Necessary adjustments to the non homogeneous case will easily be made. This Markov process (*cf.* [2, 5]) takes its values in the set of all configurations. If z and z' are two configurations, the transition rate from z to z' is denoted by $\lambda_{z,z'}$. It is non null only if the two configuration differ by exactly one vehicle transfer. The rate at which configuration z may change is denoted by λ_z. It is the sum of all $\lambda_{z,z'}$'s. It turns out that the maximal possible value for λ_z is finite. When this is the case, the Markov process is said to be *harmonisable.* We shall denote by ν the maximal possible value of λ_z. This quantity ν is the sum over all sections of their maximal cumulated exit rates, as defined in section 3.

The (discrete time) *harmonised Markov chain* associated to the (continuous time) model has transition probability

$$\frac{\lambda_{z,z'}}{\nu} \, ,$$

between distinct configurations z and z', and $1 - \lambda_z/\nu$ from z to z itself.

5.3 Simulation and time scale

It is a well known fact that a continuous time Markov jump process can be simulated by its harmonised Markov chain, with a Poissonian time scale whose intensity is equal to the harmonisation coefficient ν (*cf.* [7, 15]). Thus the general simulation algorithm is the following

```
$t \longleftarrow 0$
Initialise $X$
Repeat
    $z \longleftarrow Z$          $\{$*current configuration*$\}$
    choose $z'$ with probability $\lambda_{z,z'}/\nu$
    $X \longleftarrow z'$          $\{$*next configuration*$\}$
    $t \longleftarrow t - \log(Random)/\nu$
Until (simulation stops)
```

In our case, the line

```
    choose $z'$ with probability $\lambda_{z,z'}/\nu$
```

is decomposed into successive random choices as indicated in section 3.

The time scale is incremented at each step by an exponential random variable with parameter ν, i.e. with expectation $1/\nu$.

```
    $t \longleftarrow t - \log(Random)/\nu$
```

(Here *Random* designates the generator, which returns a random real, uniformly distributed on the interval $[0, 1]$.)

The network we wish to simulate is large and the constant $1/\nu$ (inverse of a sum over all sections) is small. The number of iterations will be very large, so that, using the law of large numbers, the time scale will be roughly proportional to the number of iterations multiplied by $1/\nu$. This observation permits to suppress the random incrementation, which is replaced by a constant incrementation. Thus the time scale merely becomes a counter of the number of iterations. This induces a quite sizable gain in computer time.

Another advantage is the following. Since the CPU time of each iteration is approximately constant, the actual running time is also proportional to real time. Thus the simulation gives an accelerated picture of the actual evolution of the model. This is important in particular if the program is to be used as a visual prediction tool, as was explained in section 4.

References

[1] F. Baccelli, G. Cohen, G. Olsder, and J.P. Quadrat. *Synchronization and Linearity: an algebra for discrete event systems.* Wiley, Chichester, 1992.

[2] A.T. Barucha-Reid. *Elements of the theory of Markov processes and their Applications.* McGraw-Hill, London, 1960.

[3] G.W. Brams. *Réseaux de Petri: théorie et pratique.* Masson, Paris, 1983.

[4] E. Casetta. *Estimation of trip matrices from traffic counts and survey data : a generalised least squares approach estimator.* Transportation Research, Series B, 21(2), 1987.

[5] E. Çinlar. *Introduction to stochastic processes.* Prentice Hall, New York, 1975.

[6] G.S. Fishman. *Monte Carlo concepts algorithms and applications.* Springer-Verlag, New York, 1996.

[7] J. Keilson. *Markov chain models - rarity and exponentiality.* Number 28 in Applied Mathematical Sciences. Springer-Verlag, New York, 1979.

[8] H. Mahmassani, R. Herman. *Dynamic user equilibrium departure time and route choice on idealized traffic arterials.* Transportation Science, 18:362-384,1984.

[9] G. Marsaglia and A. Zaman. A new class of random number generators. *Ann. Appl. Probab.*, 1:462–480, 1991.

[10] T.G. Robertazzi. *Computer networks and systems: queuing theory and performance evaluation.* Springer-Verlag, New York, 1990.

[11] R.Y. Rubinstein. *Monte-Carlo optimization, simulation and sensitivity of queuing networks.* Wiley, New York, 1986.

[12] P. L. Toint. *Transportation modelling and emerging technologies.* Faculté des sciences de Namur, Report 93-23.

[13] J. Walrand. *Introduction to Queuing Networks.* Prentice-Hall, New York, 1989.

[14] K. Watkins. *Discrete event simulation in C.* McGraw-Hill, London, 1993.

[15] B. Ycart. *Simulation de modèles markoviens.* Cours de DESS polycopié, Grenoble 1997.