J. Comp. Int. Sci. (2017) 8(1):01-10
http://epacis.net/jcis/PDF_JCIS/JCIS-0116.pdf
jcis@epacis.net

Pan-american Association of Computational Interdisciplinary Sciences

A Parallel Cellular Automaton Applied to Traffic Simulation With Different Driving Style

Marcelo Zamith[a1], Juliana M. N. S. Zamith[a], Regina Célia P. Leal-Toledo [b], Esteban Clua[b]

[a]Universidade Federal Rural do Rio de Janeiro
[b]Universidade Federal Flumimense

Received on October 31, 2016 / accepted on January, 2017

Abstract

The traffic movement and the demand by public or private transport are problems of present days. All cities in the world suffer with traffic jammed, air and noise pollution, affecting our life quality. With this in mind, simulation systems have arose as solution to understand traffic phenomenas and help us to design vehicular traffic efficiently. These simulation systems are based on mathematic models so that they can describe the traffic behavior. Among mathematic models employed, Cellular Automaton (CA) are able to mimic the basic characteristics of vehicular traffic. CA models, that take into account different driving style, demand more computation between two consecutive simulation time steps and also require more neighbors to calculate the correct velocity of each vehicle, which implies more memory access. This work proposes a new parallel CA model which includes new policies of acceleration and distance perception based on continuous probability function to describe the unpredictable behavior of driving styles. This model is presented in parallel platform approach in order to evaluate its performance in execution of the proposed model in simulating traffic big cities. The model herein illustrates qualitative and quantitative results similar to those suggested by the literature. Besides, we conducted tests in order to evaluate the parallel execute of the proposed model.

Keywords: Cellular Automaton, Traffic Simulation, Parallel Platform

1. Introduction

The traffic movement and the demand by public or private transport are problems of present days. All cities in the world suffer with traffic jammed, air and noise pollution, affecting our life quality. Thus, the correctly understanding of vehicular traffic on road network allows to find out how to minimize traffic jammed effects.

[1]E-mail Corresponding Author: mzamith@ufrrj.br

doi:10.6062/jcis.2017.08.01.0116

Indeed, traffic prediction and simulations tools become useful to help us to understand traffic dynamic. Furthermore, some researches have been developed to deal this kind of problem [1]. Among these work, some researchers have proposed mathematical models in order to describe traffic flows. These models are classified in macroscopic and microscopic. The former describes the traffic as fluid. The latter makes arise a global behavior from local interaction. Among these models, Cellular Automaton (CA) is able to describe traffic flow from a simple set of rules [2].

Besides the mathematical model, another problem with traffic prediction and simulations tools is in the fact that traffic flow of big cities demands a high power processing and huge quantity of memory achieved in computers cluster. Traffic simulation of cities may reach in a millions order. For example, New York City has $2,107,321$ vehicles registered in 2015 [3] (buses, truck, car, etc..), moving through $6,300$ miles in streets in the city [4]. This example requires a lattice of $20,277,734$ cells in case of CA model, considering discretization with cells of 0.5m.

In this paper, we propose a CA model applied for traffic simulation that considers several driving styles that extends previous works [5, 6]. Our extension consists in distance perception of ahead vehicle without anticipation policy, as proposed by authors in previous works. Instead of computing several intermediaries time steps as presented in [5] or using more data from neighborhood and more neighborhoods to calculate the anticipation policy [6], we proposed herein a higher space discretization and velocity of ahead vehicle so as to define the distance perception, adapting the acceleration policy proposed in [6]. Furthermore, the proposed model is optimized to take the advantages of parallel machines.

In this work some related work is reviewed in Section 2, following we brief describing the variables related to traffic simulation based on cellular automaton and the base model as well as its extension in Section 3. Section 4 discusses tests and results and Section 5 presents the conclusion and future work.

2. Related Work

The first CA model applied for traffic simulation is named NaSch's model and it was introduced bt Nagel and Schreckenberg [2]. Next models are based on NaSch's model and try to improve certain characteristics of traffic, such as flow-density relation or the synchronized flow region that is also known as meta-stability region [7, 8].

The CA models that improve flow-density relation consider the continuous movement of drivers, where the ahead vehicle is still moving, at the

next time step, in the same velocity of previous time step. These models require computing of several intermediaries time steps [9, 10, 5]. Another anticipation policy accesses several data from more neighbors in order to guarantee the correct velocity definition [6].

Hoekstra et al. [11] presented and discussed about CAs in multi-core clusters architecture which uses the cluster processing power by low cost. The challenge discussed in this work was the level of communication in parallel machine which uses the memory hierarchy of one machine to provide the communication among the cores. The bottlenecks are network bus in case of former and memory bus in case of the latter.

In literature, we find works that achieved more power processing based on distributed memory. Among them, Dattilo et al. [12] proposed a distributed CA model to simulate landslide which achieved a good performance on cluster such as Beowulf clusters. Oliverio et al. [13] proposed another approach which just uses shared memory based on OpenMP compilation directives in CA models to simulate lava flows. To simulate large scale domains, Bandman [14] proposed an approach that uses shared memory. In this case, the author uses several different CAs which each one is computed in one machine. Each CA domain is partitioned in fragments, using domain decomposition method, and each fragment is assigned to a core. Following, all CAs interacts one another so that they can simulate very large scale domain.

3. Parallel Cellular Automaton Model

We proposed herein an optimization CA model for traffic simulation to take advantage of parallel machines. So that we can mimic very large domains, such as big cities, and provide a computing of the simulation in acceptable time.

Traditionally, CA models applied in traffic simulation work with space and time discretization, where space represents the highway, streets or avenue, the spaces where vehicles move. Time defines the steps which the simulation evolves. With this in mind, each cell can represent a space of 7.5m or 1.5m, considering that a vehicle occupies just one cell in former approach or five cells in the latter. Time discretization is given in time steps of one second [2, 5, 6]. These discretization represent quite well the flow traffic in highways, however, they do not describe correctly the traffic movement in big cities, due to the low speed limit of road network and the jammed flow in these roads or even the interaction with pedestrians, that require a very refinement discretization (cells of 0.5m) [15]. Hence, to simulate traffic

in cities, it is necessary to work with more refinement discretization. In this case, we used cells of 0.5m and each vehicle occupies fifteen cells. Buses and small trucks are represented by thirty cells.

The proposed model considers that all drivers accelerate their vehicles to the maximum speed limit or allowed by vehicle flow. Eventually, any driver may not be this behavior and he is going to keep its velocity by subsequents times steps (driving style). In according to driving style, each driver reduces his acceleration desire as close as he is to the ahead vehicle or traffic lights.

Algorithms 1 and 2 describe our model, the former defines the correct velocity of each vehicle while the latter updates them on the lattice. The variables related to a vehicle is its velocity (v_i^t), distance (d_i^t) and length (l_i). t is the current time step, i is i^{th}-vehicle and his ahead vehicle is given by $i-1^{th}$-vehicle. The speed limit is given by v_{max}. $Beta$ PDF models driving styles [16].

Line 4 in Algorithm 1 describes the acceleration policy and lines 6 to 10 model the distance perception of ahead vehicle. Our solution is explicit in time and pretends to ensure the individual correct velocity definition. Moreover, our CA models herein requires few data from ahead vehicle, reducing the memory access in parallel machines that improves the performance of simulation during the computing. i^{th} vehicle increases its velocity $\Delta v \times \alpha$ in each time step ($\alpha \in [0:1]$), $\Delta V = 12$ and it represents a quantity of cells that one driver can accelerate, α is close to 0 makes i^{th} vehicle maintain its velocity, as long as i^{th} vehicle can accelerate up to 12 cells in one time step if α is close to 1. Line 6 in Algorithm 1 calculates the distance between two consecutive vehicles which is used in line 8 to reduce the driver acceleration desire. Braking rule is given by line 10 and line 11 defines the correct velocity.

The parameter d_s is close to zero when a driver is moving faster and close to the ahead vehicle, while cautious driver takes more time steps to accelerate its vehicle and also keeps far to the next vehicle, in this case d_s is close to D, where $D = 6$. d_s is defined in according to β (line 3 of Algorithm 1).

Input: (Lattice, vehicles attributes)
Result: (New vehicles velocity definition)
1 **for** *For each vehicle do in parallel* **do**
2 $\alpha = Beta(a, b)$
3 $\beta = 1 - Beta(a, b)$
4 $v_i^t = min\left(v_i^t + (\Delta v \times \alpha), V_{max} - (3 \times \beta)\right)$
5 $d_i^t = x_i^{t-1} - x_{i+1}^{t-1} - l_{i+1}$
6 $t_h = \frac{d_i^t}{v_i^t - (v_{i+1}^{t-1} + \Delta v \times \alpha)}$
7 **if** $t_h \leq (h \times \beta)$ **then**
8 $d_s = D \times \beta$
9 **end**
10 $d_i^t = max(d_i^t - d_s, 0)$
11 $v_i^t = min(v_i^t, d_i^t)$
12 **end**
13 Synchronize all vehicles

Algorithm 1: Vehicles velocity definition.

Input: (Lattice, vehicles attributes)
Result: (Updated Lattice, New vehicles position)
1 **for** *For each vehicle do in parallel* **do**
2 $x_i^t = x_i^{t-1} + v_i^t$
3 **end**
4 Synchronize all vehicles

Algorithm 2: Update vehicles.

We adopted a specific workload balance approach which divides the vehicles among the threads, that are moving on the part of lattice that was defined in the begining of simulation. It means, instead of processing each point of the latticle, a list of vehicles is computed. Our proposed parallel model adopts the communication in the share memory hierarchy among the cores that belong in the same machine.

Our CA model is composed by two stages: *i*)velocity definition of all vehicles at the current time step (Algorithm 1) and, *ii*) update stage (Algorithm 2) is responsible for updating each vehicles in the latticle. Our model requires only three synchronize points: the first is to guarantee that vehicles velocity definition precedes update vehicles position (line 13 of Algorithm 1). The second guarantees data communication consistence (lines 5 and 6 of Algorithm 2) among processes. The last synchronize point is at the end of update stage, so that the velocity definition can access consistent data in the next time step. We adopted barrier as synchronize object in the three points due to all threads should wait one another before going to the next point (pthread API).

4. Tests and Results

In this section, we describe and discuss two set of tests: the first test is related to the quantitative and qualitative results of simulation and, the second one presents performance and speedups obtained by parallel approach. We setup the same configuration for both set of tests: periodic boundary condition (inflow is equal to outflow), 30.000 cells, 28.800 simulation time steps and the first 14.400 time transient time steps, which means 15 km in 4 hours, considering cells of 0.5m. The global traffic occupation is in a

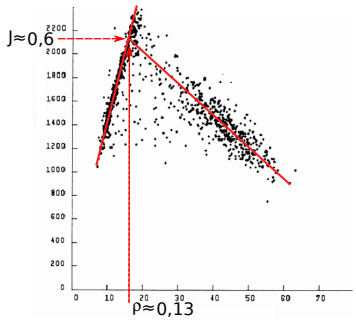
range from 0.01 (20 vehicles) to 0.95 (1,900 vehicles). The maximum speed value allowed is 75 cells per unit time (135 km/h), considering the fact that each cell represents a space of 0.5m and a vehicle has 7.5m of length. In the first test we used parallel version with 8 threads, Figure 1 and 2 display the results.

Figure 1 presents the accuracy of our method during traffic simulation. On one hand, Figure 1(a) shows a real data [17]. On the other hand, Figure 1(b) presents result of a simulation. Our model resembles quite well the same traffic dynamics empirical observed by Fred et al. [17].

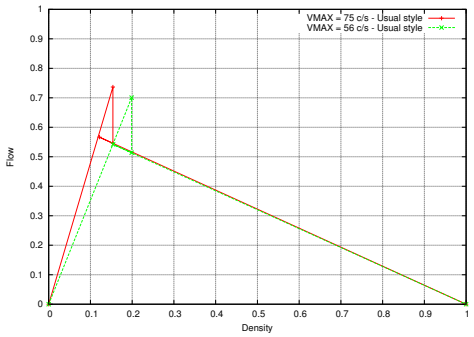
The free flow is sustained while vehicles moves in desired speed and they have enough space among them, not being influenced by any driving style. With increasing vehicles on the highway, the free space among vehicles decreases up to these vehicles cannot move on desired velocity. In this case, any velocity fluctuation of a vehicle makes other vehicles reduce their velocity, making arise slow-to-start phenomena (a vehicle takes much more time to accelerate than brake). Considering the little space among vehicles, the maximum flow is achieved when all vehicles move close to one another at the maximum speed limit.

The maximum speed limit and driving style define in which density the traffic flow would become free, jammed or synchronized. Figure 1(b) shows the influence of maximum speed limit in free, synchronized or jammed flow. Red solid line represents the simulation based on maximum speed limit of 135km/h, as long as green dashed line is simulation where vehicles were limited in 100km/h. The former guarantees free flow up to 13% occupation of road (approximately) and in the latter the free flow is sustained up tp 17% (approximately). In fact, our model resembles this dynamic traffic as described in theory [18].

Figure 2 depicts the influence of driving style on flow-density diagram. As theory would predict, slow drivers (unhurried - $Beta(6, 5)$) arise great empty spaces among vehicles. In doing so, empty spaces among vehicles reduces the free flow prematurely. On the other hand, drivers who have style to move close to one another improves the flow-density relation due to vehicles are as close as possible one another (Hurried drivers - $Beta(8, 2)$). The average driving style is modeled by Usual Style is modeled by $Beta(4, 2)$. As any others CA models, our proposed model does not consider crashes among vehicles. In general, CA models focus on vehicular movement and the interaction one another.



(a) - Real data adapted from Fred et al. [17].



(b) - Two v_{max}.

Figure 1 - Flow-density diagram - Real data and simulated.

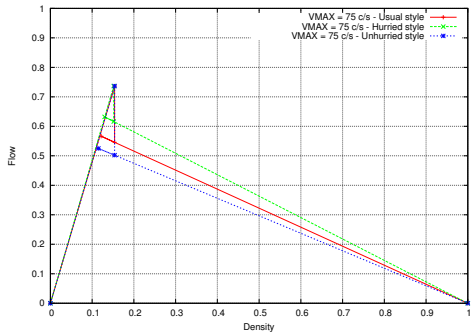
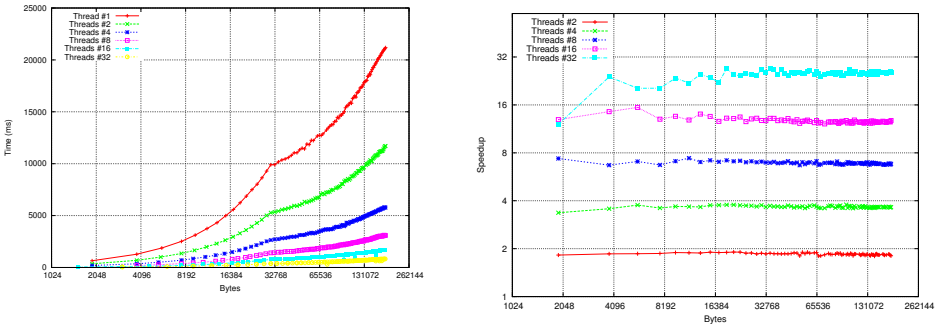


Figure 2 - The influence driving style on flow-density diagram.

The environment is composed by machines which have two quad-core processors (Intel Xeon 5355 Clovertown). Each core has one private L1 cache (64KB) and share one L2 cache (8MB) with another core on the same processor. All cores of a same machine have a uniform access to a 16GB main memory module. CentOS 5.3 is the operating system with kernel is 2.6.18.

We adopted pThread API to manage and create the threads and we used **posix_memalign** C API to improve caches access. This API allows us to allocate memory aligned with cache line, avoiding data fragmentation in memory. We also used **pthread_setaffinity_np** C API, so that we can



(a) - Simulation time.

(b) - Speedup achieved.

Figure 3 - Performance obtained with simulations.

make the same thread be re-scheduling in the same core, taking to advantage of data fetch in cache L1[19].

We conducted the performance tests in order to determinate how our parallel CA model behaves in parallel machine, defining the limit where the parallel execution does not take the advantage of machine. Thus, we executed tests with 1, 2, 4, 8, 16 and 32 threads and we divided vehicles among the threads, guaranteeing, at least, one vehicle per thread. As the machine works in hyper-thread mode, 16 and 32 threads share physical cores.

Figure 3(a) shows the elapsed time in processing simulation and Figure 3(b) presents the speedup achieved with multi-threads. In both diagram, x-axis represents the time of the amount of vehicles computed in bytes, as Figure 3(a) illustrates. Figure 3(b) shows the machine achieved linear speedup to all instances, even-if computing 32 threads, where 4 threads shared the same core. In fact, our model is efficient due to our data structure that takes the advantage of linescaches and has fine grain.

7. Conclusion

We proposed new parallel CA model which is mimicking basics dynamic of traffic and also are able to model vehicles interacting in slow velocities, that are necessary in cities, taking into account several driving styles and vehicles (buses, trucks, etc...). Moreover, the model is composed by few rules and requires few neighborhood data. These two elements combined with a correct cache memories alignment allowed us to take the advantage

of hyper-thread machines, achieving linear speedups with high quantity of CPU threads, as illustrated by results.

This work opens some question that we will try to answer in future work: *i*) reducing or minimizing the communication step among cluster machine and, *ii*)interaction with another CA model responsible for pedestrian movement.

References

[1] L. A. Pipes. An operational analysis of traffic dynamics. *Journal of Applied Physics*, 24:274–281, 1953.

[2] K. Nagel and M. Schreckenberg. A cellular automaton model for freeway traffic. *Journal de Physique I*, 2:2221–2229, December 1992.

[3] NYS Vehicle Registrations on File 2015. Available at `https://dmv.ny.gov/statistic/2015reginforce-web.pdf`, 2015.

[4] The City of New York. Available at `http://www.nyc.gov/html/dot/html/infrastructure/infrastructure.shtml`, 2016.

[5] Marcelo Zamith, Regina Célia P Leal-Toledo, and Esteban Clua. A new approach of cellular automata applied for traffic simulation. In *Proceedings of the third Conference of Computational Interdisciplinary Sciences - CCIS*, pages 243–254, 2014.

[6] Marcelo Zamith, Regina Célia P Leal-Toledo, Esteban Clua, Elson M Toledo, and Guilherme VP de Magalhães. A new stochastic cellular automata model for traffic flow simulation with drivers behavior prediction. *Journal of Computational Science*, 9:51–56, 2015.

[7] M. Takayasu and H. Takayasu. 1/f Noise in a Traffic Model. *Factal 1*, 5:860–866, 1993.

[8] A. Kuang. Effect of slow-to-start in the extended bml model with four-directional traffic. *Physics letters. A*, 378:1455–1460, 2014.

[9] M E Larraga, J A del Rio, and A Schadschneider. New kind of phase separation in a ca traffic model with anticipation. *Journal of Physics A: Mathematical and General*, 37(12):3769–3781, 2004.

[10] M. E. Larraga and L. Alvarez-Icaza. Cellular automaton model for traffic flow based on safe driving policies and human reactions. *Physica A: Statistical Mechanics and its Applications*, 389:5425–5438, 2010.

[11] A.G. Hoekstra, J. Kroc, and P.M.A. Sloot. *Simulating Complex Systems by Cellular Automata*. Understanding Complex Systems. Springer Berlin Heidelberg, 2010.

[12] Giuseppe Dattilo and Giandomenico Spezzano. Simulation of a cellular landslide model with camelot on high performance computers. *Parallel Comput.*, 29(10):1403–1418, October 2003.

[13] Marco Oliverio, William Spataro, Donato DAmbrosio, Rocco Rongo, Giuseppe Spingola, and Giuseppe A. Trunfio. Openmp parallelization of the sciara cellular automata lava flow model: performance analysis on shared-memory computers. *Procedia Computer Science*, 4:271 – 280, 2011.

[14] O. Bandman. Implementation of large-scale cellular automata models on multi-core computers and clusters. In *High Performance Computing and Simulation (HPCS), 2013 International Conference on*, pages 304–310, July 2013.

[15] Carsten Burstedde, Kai Klauck, Andreas Schadschneider, and Johannes Zittartz. Simulation of pedestrian dynamics using a two-dimensional cellular automaton. *Physica A: Statistical Mechanics and its Applications*, 295(3):507–525, 2001.

[16] M.C. Jones. Kumaraswamys distribution: A beta-type distribution with some tractability advantages. *Statistical Methodology*, 6(1):70 – 81, 2009.

[17] L. H. Fred, L. A. Brian, and A. G. Margot. Empirical analysis of freeway flow-density relationships. *Transportation Research Part A: General*, 20(3):197 – 210, 1986.

[18] C. F. Daganzo. *Fundamentals of transportation and traffic operations*. Pergamon Press, ed. limitada edition, 1997.

[19] Juliana M. N. Silva, Cristina Boeres, Lcia Maria de A. Drummond, and Artur Alves Pessoa. Memory aware load balance strategy on a parallel branch-and-bound application. *Concurrency and Computation: Practice and Experience*, 27(5):1122–1144, 2015.