An Approach for Traffic Forecast with GPU Computing & Cellular Automata Model

Marcelo Zamith
Mark Joselli
Jose Ricardo Silva Junior
Regina Célia P. Leal-Toledo
Esteban Clua
MediaLab, IC-UFF

Eduardo Soluri
Nullpointer Tecnologia

Abstract

Traffic forecast has been of practical interest for modern society, mainly in minimizing of jammed traffic effects due to the saturation of roads, as well as predictable the impact of road interventions. In this way, a family of computational methods that represent basic traffic characteristics is based on Cellular Automata (CA). Moreover, the simulation of traffic system for greater roads lattice requires a high computational cost. Therefore, this work presents an approach of sophisticate CA models applied for traffic simulation on GPU.

Keywords:: An approach for freeway traffic forecast

Author's Contact:

{mzamith, jricardo, mjoselli, leal, esteban}@ic.uff.br
esoluri@nullpointer.com.br

1 Introduction

Transportation is a necessity of modern society. However, vehicle fleet increases faster than roads, not to mention environment pollution and deteriorating urban areas due to traffic disorderly growth. Thus, a correct manager of traffic system can minimizing these effects. The traffic saturation on the roads, the jammed traffic and environment pollution has been studied since 1950s. Louis A. Pipes [Pipes 1953] was one of first researchers in studying the traffic motion in which describes mathematically the vehicular motion.

Many researches try to describe mathematic models for understating traffic flows its bottlenecks. These models are classified as macroscopic or microscopic. Among the microscopic models, Cellular Automata (CA) has been adopted with good results, due to the fact that it creates similar behavior features of traffic vehicular motion, such as the relation of flow-density and the metastable phase [Wahle et al. 2001], [Hafstein et al. 2004], [Nagatani 2009] and [Spyropoulou 2007]. Furthermore, CA models are widely adopted to describe complex system through simple rules set. CA models are time explicit, i.e., for defining state of current time, CA models use all informations of previous one, they are parallelizable models.

The model proposed herein improves previous CA models, which includes in a new manner anticipation and acceleration policies. Indeed, these models present some difficultly for adjusting the velocities of vehicles which consider the continuous movement of ahead vehicle. The proposed model herein has also the capacity to predict the movement of head vehicle.

2 The Traffic-Flow Theory

The behavior of traffic is analyzed in according to three variables: density (ρ), velocity (v) and flow (J). The first (ρ) is the number of vehicles per unit length on a highway at some time (Eq. 1), where n is the number of vehicles and L represents the full length of the highway's segment under observation.

$$\rho = \frac{n}{L} \tag{1}$$

The average velocity, i.e., the averaged sum of velocities, is obtained with the following (Eq.2)

$$\overline{v} = \frac{\sum_{i=1}^{n} v_i}{n} \tag{2}$$

Finally, the flow is defined as the number of vehicles (m) that pass through a specific point of the highway in a given time interval (T), as denoted in Eq. 3.

$$J = \frac{m}{T} \tag{3}$$

Moreover, the variables which describe vehicular motion data can be obtained by two ways: either by an instantaneous photograph of a specific section with length L or by observing the flow from a fixed location during an interval time T. For the first case Eq. 1 and 2 describe the density and average velocity considering a continuous section of the freeway as a photograph taken at a given instant of time. The second approach, fixed location, Eq. 3 considers the counting strategy during a gap time. Finally, it is possible to build a relation among these variables, as described in the following Eq. 4 (considering a stationary traffic state).

$$J = \rho\overline{v} \tag{4}$$

Based on previous equations, the analysis of traffic flow is accomplished by constructing the corresponding fundamental diagram. This shows the relation of flow and density. Therefore, the theoretical model is illustrated in Fig. 1 as long as a fundamental diagram for real data is presented in Fig. 2. Upon inspection of the fundamental diagram (Figs. 1 and 2) it is possible to identify three well defined phases, as Kerner et al [Kerner et al. 2002] discussed. The first one represents the free-flow and corresponds to the region of low to medium density and weak interaction between vehicles, any velocity fluctuation only reduces the distance among vehicles without affecting average velocity. As illustrated by Fig. 1, the vehicles move almost at the highways speed limit, and the flow increases linearly with density level, within the interval of $0 \leq \rho \leq c_1$. The second phase represents medium and high density and is also known as the synchronized flow. The flow should behave either free or jammed, i.e., the phase $c_1 < \rho < c_2$ shows a flow that is not defined only by density, but by the interaction among the vehicles. Considering the fact that each vehicle moves close to next one,when just one vehicle decreases its velocity, the previous vehicles should reduce their velocity, and there is a reducing at highway average velocity. At next time instants, the vehicles begin to increase their velocity and highway average velocity increased again. In this phase, little jammed flow formed and dissolved. This phase is named the metastable phase. The last phase represents the jammed flow, where an increase in density causes a decrease in the flow ($\rho > c_2$).

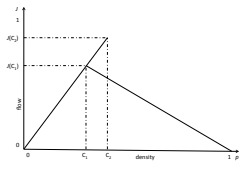

Figure 1: *Theoretical fundamental diagram*

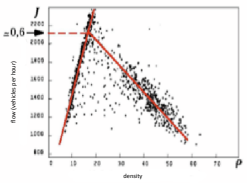

Figure 2: *Real data fundamental diagram*

3 Cellular Automata

The Cellular Automata (CA) is model mathematic widely employed for describing complex system, such as traffic, crowd simulation, etc. Its concept was presented by J. Von Neumann[Neu]. A theoretical systematic description of CA was developed by Wolfram [Wolfram 1984]. It is a robust and powerful mathematical model for simulating the dynamics systems, typically used for physics, biology, ecology, fluid flow, crowd behavior, among others [Takai et al. 1995], [Ling et al. 2004] and [Gaylord, Richard J. and D'Andria, Louis J. 1998].

CA models applied to traffic problems can describe basic proprieties of the traffic. They are used for representing the vehicular movement across the lattice, which represents the highway. Nagel and Schreckenberg (NaSch) [Nagel and Schreckenberg 1992] proposed the successful model that has been widely employed, which became the basis for several developments and is referred to NaSch's model. Several others models were proposed based on NaSch's models, which create families of models.

A well known family of these solutions is named slow-to-start. This model tries to mimic the driver inertia in returning his velocity whenever he stops. It was introduced by Takayasu and Takayasu (T2) [Takayasu and Takayasu 1993]. Several others works were proposed [Benjamin et al. 1996], [Tian 2009], [Barlovic et al. 1998] and [Nishinari et al. 2004]. Despite depicting metastable region, this family model should work with more than one probability with different range of values. At last, metastable region relies on its initial condition: For free flow, the vehicles must be distributed uniformly by the CA lattice. On the other hand, jammed flow is represented by vehicles in jammed state on the CA lattice.

Another family of CA applied for traffic are related to those that include anticipation feature. These models resemble the continuity of drivers movement in the next instant time. The concept adopted in this approach is that every driver considers that his ahead vehicle will move with the same velocity as in the previous time instant. The disadvantage of this model family is given by the unpredictable driver's behavior due to correct velocity definition. Emmerich et al. [Emmerich and Rank 1997] suggests that each vehicle may adjust its velocity considering the movement of the vehicle ahead at the next time instant, but this strategy depends on flow and the results show that the flow becomes jammed prematurely, besides not reproducing metastable phase. Another strategy considers a percentage of velocity of the ahead vehicle as proposed by Larraga et al. [Larraga et al. 2004], including a safety distance among vehicles. Break light models is a subset of this family, which adopts a signalize strategy [Knospe et al. 2000]. Other strategy adopted by anticipation model was proposed by Zamith et al. [Zamith et al. 2010]. where a recursive procedure is added in order to correctly define each vehicle velocity. Although these models reproduce basic propriety of traffic, such as good flow-density relation and metastable region, these are done through one of following features: signalization of previous vehicle, recursive procedure for correct velocity definition and counter flow velocity adjustment. Several of them need to setup many parameters and others require a high computational power with complex data structure.

4 GPU Computing

Nowadays, GPUs are presented as solution for high performance computing. The architecture massively parallel and circuits dedicated mathematics calculus together with low cost becomes a good solution for many simulation problems, either academic researches or industry. For instance, an old GPU as nVidia 8800 ultra [NVIDIA 2006] takes a measured 384 GFLOPS/s whereas 35.3 GFLOPS/s for the 2.6 Ghz dual-core Intel Xeon 5150 [NVIDIA 2008].

Although the GPUs' high performance computation, they requires a different programming paradigm in relation to CPUs. A good manages of memory and optimized algorithms can improve the speedup in according to CPU. On the other hand, weak manages memory of GPU memory hierarchy or algorithms with conditional instructions make GPU performance low.

Indeed, CUDA enables inexpensive multi-threaded GPUs programming. One of the advantages of programming in CUDA instead of conventional Shader language programming is that it allows one to work with familiar programming concepts while developing software that can run on a GPU, avoiding the performance overhead of graphics layer APIs by compiling the software directly to the hardware [Dobbs 2008].

CUDA has several advantages over traditional general purpose computation on GPUs, such as: scattered reads - code can read from arbitrary addresses in memory; shared memory - CUDA exposes a fast shared memory region which can be shared among different threads; faster downloads and read-backs to and from the GPU and full support for integer and bitwise operations, including integer texture lookups [NVIDIA 2012]. Besides all this memory facilities of GPU, CUDA does not cater memory access by CPU. This constraint requires the CPU to previously copy the data from its memory to the GPU.

The GPU used for running CUDA is based on an unified architecture, i.e., there is a set of stream processors for vertex and pixel programming. Due to the fact that CUDA organizes this set of stream processors in a set of multiprocessor cores, it is possible to run multiple threads concurrently. The threads are arranged in blocks, which have their own shared memory space for sharing among their threads, as well as each multiprocessor can process one block. Therefore, the level of parallelism is enhanced through proper arrangement of the GPU model, blocks and threads per blocks. The block set is named grid.

CUDA enables application programming in an extension of the C language (C for CUDA). In fact, a CUDA program is a set of C functions, named kernels, that can be invoked by the host, and are executed on the device's n instance of the kernel in parallel. In CUDA architecture, CPU is defined as host a GPU as a device.

When a kernel starts running based on the execution configuration and according to the function arguments, the host continues to the next line of code, after the kernel launches. At this point, both the CUDA device and host are simultaneously running their separate programs. Nevertheless, it is possible to artificially synchronize both host and kernel programs.

A typical structure of a CUDA program is composed by the steps illustrated in Fig.3 [HBeonLabs 2009]:

1. copy of data from the main host memory to the GPU memory;
2. the host invokes the kernel;
3. the kernel runs concurrently in different threads in the device;
4. the results stored in GPU memory are copied back to the CPU.

Not all problems can be solved efficiently in CUDA. Problems which are computation-intensive and that involve processing large data sets using the same code (Single Instruction Multiple Data paradigm) are natural candidates to be parallelized on GPUs. Developing an appropriate parallel algorithm that suits the CUDA programming model may be very difficult. Besides, there are two main challenges in modeling an efficient CUDA program: breaking the problem appropriately into many sub-problems that can run concurrently in several independent threads and dealing appropriately with CUDA memory hierarchy, so that no overhead is introduced due to an inappropriate choice of data distribution, order of access and communication strategies.

5 Cellular Automata Model and GPU

CA models applied to traffic problems can describe basic proprieties of the traffic. They are used for representing the vehicular movement across the lattice, which represents the highway. Nagel and Schreckenberg [Nagel and Schreckenberg 1992] proposed the successful model that has been widely employed, became the basis for several developments and is referred to as NaSch model.

In NaSch model a single-lane highway is considered. The road is defined as a lattice composed by cells. Each cell represents a road segment of 7.5m (Δx) and the time evolution scale is measured

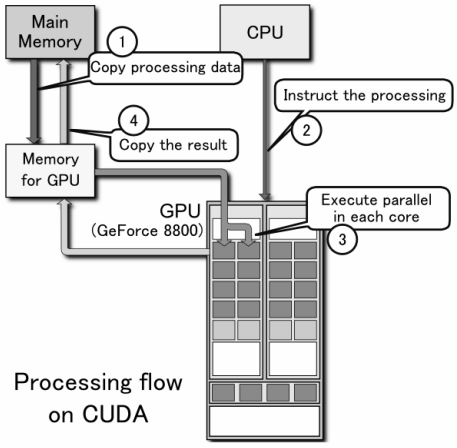

Figure 3: *Processing Flow on CUDA*

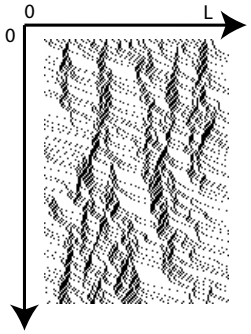

Figure 4: *Space vs. time plot of jam*

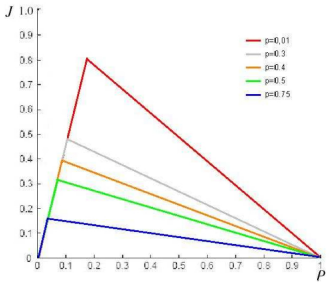

Figure 5: *NaSch model result*

in seconds, where the time interval is represented by Δt. A position in the lattice and the time are given by x and t, respectively, where $t \in \mathbf{N}$, $x_i^t \in \mathbf{Z}$. The highway is considered to have periodic boundary conditions (closed circuit), i.e., the position x is the same as position $x + L$ where L is the length of the circuit. Likewise, the ith vehicle is the same as the $(i + N)$th vehicle, where N is the number of vehicles. Moreover, the variable (v_i^t) denotes the velocity of the ith vehicle at time instant t, measured in cells per units of time and x_i^t denotes the spatial position. The distance between two vehicles, head to head, in a time instant is represented by d_i^t, and the maximum speed allowed is given by v_{max}.

Finally, the probabilistic feature of the model is given by p and p_m, where the first is chosen from an uniform distribution and the second is an initial parameter which plays the role of a cutoff. Both contribute to mimic the drivers' tendency to preserve their velocities at the next time instant $t + 1$.

NaSch model presents basic and simple rules, where all drivers attempt to drive in the maximum velocity allowed by the flow of vehicles or the road's speed limit; a driver shall keep the vehicle's velocity or, without any apparent reason and with a small probability, simply reduce it. A subsequent adjustment of each vehicle's speed considers its distance to the one immediately ahead, following the rules:

$$\begin{aligned} \text{Acceleration}: \quad & v_i^t = min[v_i^{t-1} + 1, v_{max}] \\ \text{Distance}: \quad & d_i^t = x_i^{t-1} - x_{i+1}^{t-1} - 1 \\ \text{Adjust}: \quad & v_i^t = min[v_i^t, d_i^t] \\ \text{Probability } (p \leq p_m): \quad & v_i^t = v_i^{t-1} - 1 \\ \text{Update}: \quad & x_i^t = x_i^{t-1} + v_i^t \end{aligned} \tag{5}$$

Moreover, the data structure of the model is also simple, and it requires only $O(n)$ memory space, where n is the number of cells.

NaSch model is able to reproduce properties of real traffic flow, like the density-flow relation and spatio-temporal evolution of jams, shown in Fig. 4. However, it indicates that the traffic becomes jammed prematurely. Real data denote the traffic becomes jammed for $J \approx 0.6$, as illustrated by Fig. 2. Fig. 5 illustrates the effects of different values of p_m, indicating in which density value the free flow behavior becomes jammed. If a probabilistic drivers' behavior ($p = 0.30$) is introduced, the flow becomes jammed for $J \approx 0.5$. NaSch does not give rise to the metastable phase.

5.1 GPU Implementation Approach

GPU implementation is divided in two kernels: The first one is responsible for correct velocity definition (lines 1 to 4) of model 5. The second kernel just updates vehicles position. The algorithm 1 is pseudo code of model, where vehicles are processed in parallel independently of highway discretization. kernel 1 input parameter are: current vehicle state, probability p_m and speed limit of highway (v_max). At the end of algorithm 1, all vehicles have defined

```
   Input: (p_m, current state, v_max)
   Result: new velocity value
1  for all vehicles do in parallel do
2      copy from global memory to shared memory vehicle
       position (x_i)
3      synchronize threads
4      v_i^t = min[v_i^{t-1} + 1, v_max]
5      d_i^t = x_i^{t-1} - x_{i+1}^{t-1} - 1
6      if p ≤ p_m then
7          v_i^t = max[v_i^{t-1} - 1, 0]
8      end
9  end
```

Algorithm 1: Algorithm of kernel 1.

the correct velocity in order to not hit to next vehicle. Thus, kernel 2 described by algorithm 2 has input the correct velocity definition and output is the new state of each vehicle, i.e., the new position on the highway. For each time step simulation two kernels synchronized are invoked.

```
   Input: (new velocity value defined in kernel 1)
   Result: new state - position update
1  for para todos os veculos do
2      x_i^t = x_i^{t-1} + v_i^t;
3  end
```

Algorithm 2: Algorithm of kernel 2.

6 Data Structure Adopted

Data structure proposed in this work is composed by two arrays in global memory and an array in shared memory. In global memory is stored a highway discretization (first array) and a list of vehicles (second array). The first array stored a integer value which represents the index of vehicle in second one. The list of vehicles array stored the vehicles' attributes (position and velocity). The array in shared memory is integer array and it is used to store vehicle position. Therefore, each thread in algorithm 1 copies its vehicles position from global to shared memory. Immediately after all threads are synchronized for velocity definition step which accesses shared memory in order to get vehicle position, as illustrated by Fig. 6. At last, empty cell is numbered with -1 values.

7 Conclusion

This work depicts a concept of Cellular Automata applied for traffic simulation forecast. Moreover, discusses an approach for correct

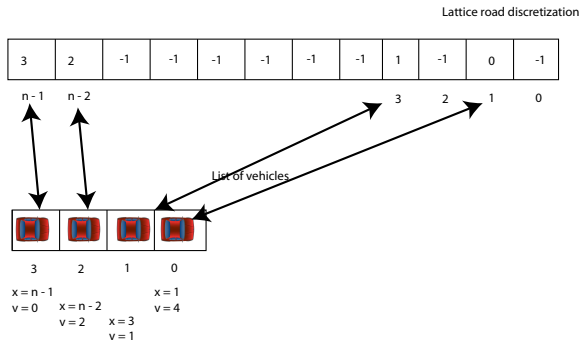

Figure 6: *Data structure in global memory*

use of GPU memory in order to maximize the use of shared memory. Finally, forecast in real time is important for traffic manages due to the fact that unexpected problems arise from vehicles interaction in real world, such as accidents. For this cases it is important to predict how long jammed flow takes s as well as length of it. Even-if problem solved in real world, how long jammed flow takes in order to dissolves. For all these necessity, high performance computing as GPUs shows very efficient.

References

BARLOVIC, R., SANTEN, L., SCHADSCHNEIDER, A., AND SCHRECKENBERG, M. 1998. Metastable states in cellular automata for traffic flow. *The European Physical Journal B 5*, 3, 793–800.

BENJAMIN, S., JOHNSON, N., AND HUI, P. 1996. Cellular Automata Models of Traffic flow Along a Highway Containing a Junction. *Journal of Physics A 5*, 3119–3127.

DOBBS, D. 2008. Cuda, supercomputing for the masses: Part 1. *Avalible at: http://www.ati.com/developer/techpapers.html*, 1–7.

EMMERICH, H., AND RANK, E. 1997. An Improved Cellular Automaton Model for Traffic Flow Simulation. *Journal of Physics A 234*, 676–686.

GAYLORD, RICHARD J., AND D'ANDRIA, LOUIS J. 1998. *Simulating Society: A Mathematica Toolkit for Modeling Socioeconomic Behavior*, 1 ed. Springer, June.

HAFSTEIN, S. F., CHROBOK, R., POTTMEIER, A., AND SCHRECKENBERG, M. 2004. A high-resolution cellular automata traffic simulation model with application in a freeway traffic information system. *Computer-Aided Civil and Infrastructure Engineering 19*, 338–350.

HBEONLABS. 2009. Cuda– compute unified device architecture. *Avalible at: http://www.hbeonlabs.com/detailcuda.htm*.

KERNER, B. S., KLENOV, S. L., AND WOLF, E., D. 2002. Cellular automata approach to three-phase traffic theory. *Journal of Physics A: Mathematical and General 35*, 47.

KNOSPE, W., SANTEN, L., SCHADSCHNEIDER, A., AND SCHRECKENBERG, M. 2000. Towards a realistic microscopic description of highway traffic. *Journal of Physics A: Mathematical and General 33*, 48.

LARRAGA, M. E., DEL RIO, J. A., AND SCHADSCHNEIDER, A. 2004. New kind of phase separation in a ca traffic model with anticipation. *Journal of Physics A: Mathematical and General 37*, 12, 3769–3781.

LING, C., XIAOHUA, X., YIXIN, C., AND PING, H. 2004. A novel ant clustering algorithm based on cellular automata. In *Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, IEEE Computer Society, Washington, DC, USA, IAT '04, 148–154.

NAGATANI, T. 2009. Traffic states and fundamental diagram in cellular automaton model of vehicular traffic controlled by signals. *Physica A: Statistical Mechanics and its Applications 8*, 388, 1673–1681.

NAGEL, K., AND SCHRECKENBERG, M. 1992. A cellular automaton model for freeway traffic. *Journal de Physique I 2* (Dec.), 2221–2229.

The general and logical theory of automata. University of Illinois Press, Urbana.

NISHINARI, K., FUKUI, M., AND SCHADSCHNEIDER, A. 2004. A stochastic cellular automaton model for traffic flow with multiple metastable states. *Journal of Physics A. 37*, 3101-3110.

NVIDIA. 2006. Geforce 8800 gpu architecture overview. tb-02787-001_v0.9. Technical report, NVIDIA.

NVIDIA. 2008. Nvidia - cuda compute unified device architecture. Programming guide, NVIDIA.

NVIDIA. 2012. Nvidia - cuda compute unified device architecture. *Programming guide, NVIDIA*.

PIPES, L. A. 1953. An operational analysis of traffic dynamics. *Journal of Applied Physics 24*, 274–281.

SPYROPOULOU, I. 2007. Modelling a signal controlled traffic stream using cellular automata. *Transportation Research Part C: Emerging Technologies 15*, 175–190.

TAKAI, Y., ECCHU, K., AND TAKAI, N. K. 1995. A cellular automaton model of particle motions and its applications. *The Visual Computer 11*, 5, 240–252.

TAKAYASU, M., AND TAKAYASU, H. 1993. 1/f Noise in a Traffic Model. *Factral 1 5*, 860–866.

TIAN, R. 2009. The mathematical solution of a cellular automaton model which simulates traffic flow with a slow-to-start effect. *Discrete Applied Mathematics 157*, 13, 2904–2917.

WAHLE, J., NEUBERT, L., ESSER, J., AND SCHRECKENBERG, M. 2001. A cellular automaton traffic flow model for online simulation of traffic. *Parallel Computing 27*, 5, 719–735.

WOLFRAM, S. 1984. Computation theory of cellular automata. *Communications in Mathematical Physics 96*, 15-57.

ZAMITH, M., LEAL-TOLEDO, R. C. P., KISCHINHEVSKY, M., CLUA, E. W. G., BRANDÃO, D. N., MONTENEGRO, A. A., AND LIMA, E. B. 2010. A probabilistic cellular automata model for highway traffic simulation. *Procedia CS 1*, 1, 337–345.