# Driver Behaviour Algorithms for the Cellular Automata-Based Mathematical Model of Traffic Flows

*Antonina* Chechina[*], *Natalia* Churbanova, and *Marina* Trapeznikova

Keldysh Institute of Applied Mathematics RAS, RU-125047, Moscow, Russia

**Abstract.** The work is devoted to the development of new algorithms describing the complex interaction of drivers for the parallel numerical implementation of the traffic flow model developed by the authors based on the cellular automata theory. As is known from observations, the behaviour of drivers largely determines how difficult it will be to pass a section with a bottleneck (a narrowing, a motionless obstacle, etc.) or even without it at the same flow values. In such conditions, cars can move in a synchronized flow at a more or less constant speed or go into "stop-and-go" mode. The authors have developed a set of algorithms for a large number of different road situations, which is featured in this article.

## 1 Introduction

Traffic flow modelling is an essential field of research nowadays, as traffic management problems require solving on everyday basis. Now, having high-performance supercomputers at hand, one can use developed traffic models and algorithms to create software for real time routing and infrastructural tasks that require immediate attention, as well as for long term road and city development tasks.

The model considered in this article was developed by the authors a few years ago and since then is being constantly modified and refined, mainly by building realistic algorithms of driver interactions in various situations and on different road elements (see, for example, [1]).

The main focus of this work are new algorithms for the model. The algorithm of driving around a wide obstacle (i.e., road maintenance works on a multilane highway) has been developed. The algorithm for boundary conditions exchange for the correct representation of a traffic jam spread across the network in case of parallel computations has been developed.

## 2 Cellular automata (CA) model for traffic flows

Traffic flow modelling allows for various approaches, though all models can be divided into two big groups: macroscopic models and microscopic models. As a rule, macro models are based on the similarity between a vehicle flow and a compressible fluid flow, so traffic is described by hydrodynamic equations and only average characteristics (flow density and velocity) are considered. Micro models on the other hand consider each vehicle separately. Usually the motion equations are written out, but it's not the only way of constructing a microscopic mathematical model of traffic flows. The approach based on cellular automata (CA) realizes this possibility in order to avoid dealing with differential equations and to ensure more flexibility in depicting drivers' interactions.

The cellular automata (CA) approach has a long and successful history of usage in various fields of research, including medical, biological and social sciences. It was introduced into the traffic field by K. Nagel and M. Schreckenberg in 1992 [2] and, since then, is widely used for modelling vehicular traffic all over the world [3-9].

The basics of the CA traffic flow model are as follows: the computational domain (the road, see Fig. 1) is divided into equal cells, each cell is 7.5 m long and one lane wide (the amount of space a single car takes up in the maximum density case – i.e., when being in a traffic jam). The distance in the model is measured in cells, a vehicle speed – in the number of cells per time step, time is measured in steps, time step is equal to 1 second.

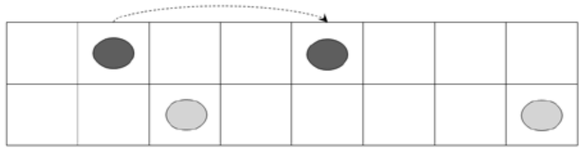

**Fig. 1.** The computational domain in the CA model.

Every time step the cell state update takes place. First, drivers make decision about changing lanes; second – vehicles move along the road by the rules of one lane traffic from Nagel-Schreckenberg model. The block-scheme for vehicular movement along the road is presented in Fig. 2.

[*] Corresponding author: chechina.antonina@yandex.ru

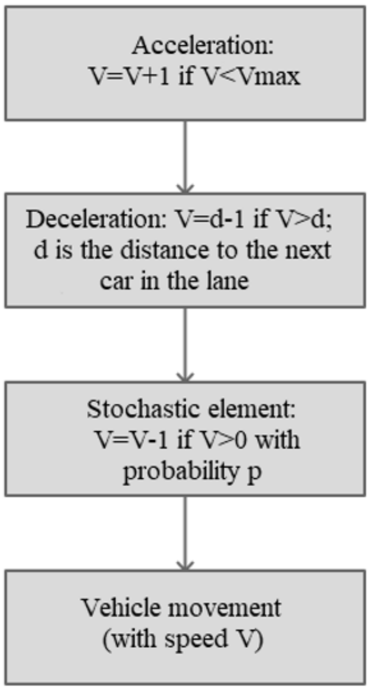
Acceleration:
V=V+1 if V<Vmax
Deceleration: V=d-1 if V>d;
d is the distance to the next car in the lane
Stochastic element:
V=V-1 if V>0 with probability p
Vehicle movement
(with speed V)

**Fig. 2.** Nagel-Schreckenberg model rules for movement along a road.

Here $V$ is the speed of a vehicle, $V_{max}$ is maximal speed of a vehicle, $d$ is the distance to the next car in the lane, and $p$ is the probability of additional speed decrease by a driver.

Basically, most complicated algorithms in the model are connected to driver behaviour. The decision making regarding a lane change depends on various factors. For each of the following situations the sequence of operations before reaching the decision will be different:
• closer/farther the car is from the intersection;
• narrowing/widening of a road;
• driver's goal (a turn, a certain exit, etc);
• presence of obstacles in a given and/or an adjacent lane;
• presence of a congestion in a given/adjacent lane;
• presence/absence of road markings prohibiting lane change;
• on-ramp or an entrance from a secondary road to a main road;
• crossing a non-signalized/signalized intersection;
• U-turn;
etc.

The authors have developed a set of algorithms for a large number of different road situations (see [10]).

## 3 Numerical realization

The model is realized as a program package written using C/C++, Glut library for visualizing the computation along with it and MPI library for boundary data exchange during parallel calculations on high-performance supercomputer systems. The program has a module structure, each module answers for modeling traffic on a certain road element (an intersection, a strait road fragment, a U-turn, etc.). From these elements a road network is constructed. Each element is computed on a separate CPU.

## 4 The algorithm for changing lanes for driving around a wide obstacle

One of the new algorithms developed for the CA model is the algorithm of drivers' decision making regarding changing lanes in presence of a wide obstacle that blocks several lanes on a multilane highway. This situation occurs fairly often when part of the road is closed for repairs. Lanes can also be blocked by slow-moving heavy duty vehicles that clean the road. The main difference between this case and the one where drivers need to move around small obstacles (car/cars that stopped due to an accident or breakage) is that the width and the position of the obstacle should be taken into consideration while choosing from which side to by-pass an obstacle to –the left one or the right one.

The algorithm of this decision making is shown in Fig. 3.

The Boolean type variable that determines if the lane change happens for the certain car at the current time step is at first set to *true* value if the step number is even (for moving to the right) or odd (for moving to the left). Then, it is determined whether the lane change would be advantageous to the driver (that is, if it results in getting into a faster and/or freer lane). For that, the speed and the distance to the next car in the current as well as in the neighbouring lane are checked and compared. If the lane change is indeed advantageous, the value of the variable stays *true*, if not – it is reset to *false*. Next, the presence of an obstacle is checked a few cells ahead, both in the current lane and in the target one, and the width of the obstacle is evaluated. To do so, the next free cell in the sequence of occupied ones is searched across the road in the direction of a supposed lane change and in the opposite one. These distances are compared to each other. If the obstacle is there, but not in the target lane, or it's there in the current lane and in the target one, but the distance to the first unoccupied cell is shorter in the desired lane change direction at the given time step, the lane change variable is set to *true*. After that, the safety is checked and lane change is cancelled if it can result in a road accident. Finally, if lane change variable is *true* after the safety check, the randomization rule is applied: the random value is received (*rand()* function in C/C++) and if it is greater than the given probability, lane change takes place, if not – it is cancelled.

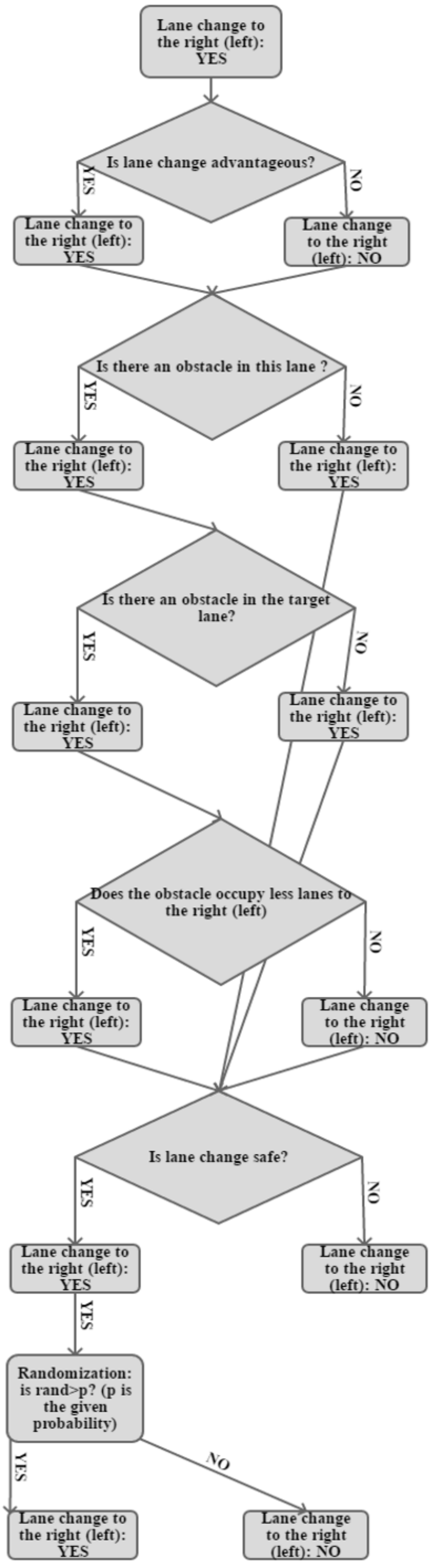
Lane change to the right (left): YES
Is lane change advantageous?
YES
NO
Lane change to the right (left): YES
Lane change to the right (left): NO
Is there an obstacle in this lane ?
YES
NO
Lane change to the right (left): YES
Lane change to the right (left): YES
Is there an obstacle in the target lane?
YES
NO
Lane change to the right (left): YES
Lane change to the right (left): YES
Does the obstacle occupy less lanes to the right (left)
YES
NO
Lane change to the right (left): YES
Lane change to the right (left): NO
Is lane change safe?
YES
NO
Lane change to the right (left): YES
Lane change to the right (left): NO
YES
Randomization: is rand>p? (p is the given probability)
YES
NO
Lane change to the right (left): YES
Lane change to the right (left): NO

**Fig. 3.** Block-scheme of the algorithm of driving around a wide obstacle.

The algorithm is realized as a module for the program package developed by the authors. The output is shown in Fig. 4.

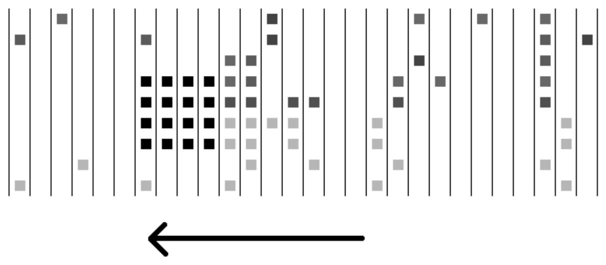

**Fig. 4.** Modelling driving around a wide obstacle.

Figure 4 represents the 9-lane highway road. The movement direction is shown with the arrow (right to left). The traffic here is right-hand side. A cluster of 4x4 black squares represents a wide obstacle. The cars are depicted by squares of different shades of gray. Darker cars enter this road fragment by first 5 lanes (slower ones), lighter cars use faster lanes - from 6 to 9.

Here we can see that darker cars choose to move to the right while lighter ones change lanes to the left, this way ensuring that obstacle by-passing takes less steps, which is the goal of the algorithm and reproduces real life driver behaviour.

## 5 The boundary conditions exchange algorithm for parallel computations

As computations for each road element are carried out on different processors and drivers don't "know" anything about traffic conditions on the next element, some probing needs to take place before a car might exit the current road element in order to see if there's the possibility to enter the next one. For that, flag variables are set to indicate the situation, and they are exchanged before actual boundary condition exchange takes place.

Block-scheme for the algorithm of car exiting a road element is shown in Fig. 5.

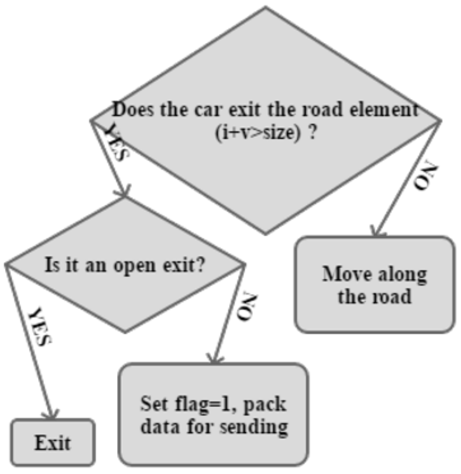
Does the car exit the road element (i+v>size) ?
YES
NO
Is it an open exit?
YES
NO
Move along the road
Exit
Set flag=1, pack data for sending

**Fig. 5.** Block-scheme of the algorithm of exiting a road element

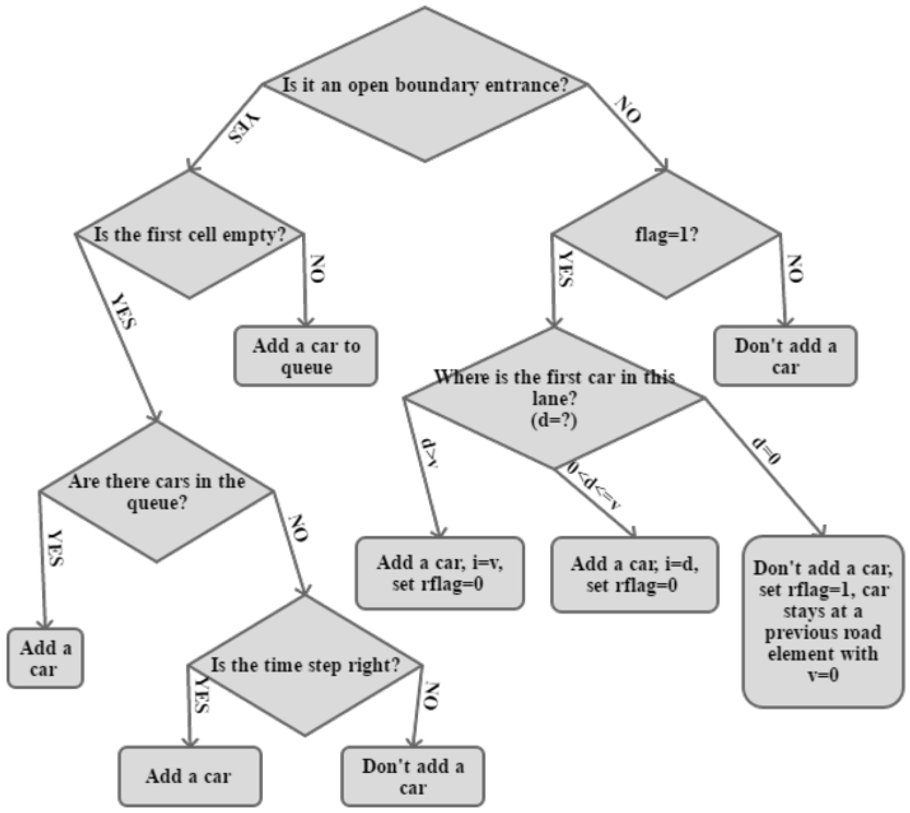
Is it an open boundary entrance?
YES
NO
Is the first cell empty?
NO
Add a car to queue
YES
Are there cars in the queue?
YES
Add a car
NO
Is the time step right?
YES
Add a car
NO
Don't add a car
flag=1?
YES
NO
Don't add a car
Where is the first car in this lane? (d=?)
d>v
Add a car, i=v, set rflag=0
0<d<=v
Add a car, i=d, set rflag=0
d=0
Don't add a car, set rflag=1, car stays at a previous road element with v=0

**Fig. 6.** Block-scheme of the algorithm of adding a car to a road element.

When a vehicle is reaching the end of the road on an element, we check if it's going to leave it at the current time step. If the number of cell *i* exceeds the size of the road after adding the current speed value, the car is about to exit. If it's a free exit with an open boundary, the vehicle just leaves the computational domain. If the road element has a neighbouring one, the data about the exiting car (its current speed, ID number, desired destination, maximal speed, the ability of the driver to become cooperative) is packed for sending and the flag variable that indicates that the there's data for receiving is set to 1 (*flag*=1).

When cars are added to the system, it happens differently whether it is a free boundary entrance or not (see Fig. 6). If it is free, we check if the first cell is empty. If it is, a car can be added, if not – it is added to the queue to be put on the road as soon as the first cell is empty. If a road element is connected to its neighbour, we check if there's data packed for transmission on the processor that is accountable for modelling a neighbouring road fragment. If said data exists, we check if there is a possibility of adding a car according to the position of its target cell. If adding is possible, we do so, setting the return flag variable to 0 (*rflag*=0) indicating that the car was added and thus can be removed from the previous element. If adding a car was not successful, the return flag is set to 1 and the car stays on the previous road element with its speed set to zero. In order to avoid high deceleration values, the situation is constantly checked for congestions at boundaries to slow down cars that approach the entrances to congested road elements.

To test the algorithm, the computations on different road networks were carried out. The results show it works correctly.

## 6 Conclusions

New algorithms for the CA model developed by authors earlier were presented in the article. The algorithms were realized as a part of the program package developed by authors and test problems were solved to confirm their adequate representation of real life traffic.

The developed program package for high performance computing systems can be used for traffic modelling on city road networks, taking into consideration different driving strategies and evaluating the efficiency of measures taken in order to decrease traffic jams. It can be used as a basis for Intelligent Transport System of a metropolis.

The calculations were performed with the use of supercomputers installed in the Collective Usage Centre of KIAM RAS [11].

## References

1. A. Chechina, N. Churbanova, M. Trapeznikova, A. Ermakov, M. German, Int. J. of Eng. & Techn., **7**(2.28), 225-227 (2018)
2. K. Nagel, M. Schreckenberg, J. Phys. I France, **2**, 2221–2229 (1992)
3. S. Maerivoet, B. De Moor, Physics Reports, **419**(1), 1-64 (2005)

4. M. E. Lárraga, L. Alvarez-Icaza, Physica A: Statistical Mechanics and its Applications, **389**(23), 5425-5438 (2010)
5. B. Kerner, S. Klenov, G. Hermanns, M. Schreckenberg, Physica A: Statistical Mechanics and its Applications, **392**(18), 4083-4105 (2013)
6. H. Jiang, Zh. Zhang, Q. Huang, P. Xie, Safety Science, **82**, 182-189 (2016)
7. A.P. Buslaev, A.G. Tatashev, M.V. Yashina, Int. J. of Eng. & Techn., **7**(2.28), 351-356 (2018)
8. Kerner B., Klenov S., Schreckenberg M. Phys. Rev. E., **84** (2011)
9. J. Vasic, J. Heather, H.J. Ruskin, Physica A: Statistical Mechanics and its Applications, **391**(8), 2720-2729 (2012)
10. N.G. Churbanova, A.A. Chechina, M.A. Trapeznikova, P.A. Sokolov, Mathematica Montisnigri, **XLVI**, 72-90 (2019)
11. KIAM – The official site of Keldysh Institute of Applied Mathematics, https://www.kiam.ru/MVS/resourses/