SYSC5104 METHODOLOGIES FOR DISCRETE EVENT MODELLING AND

SIMULATION (FALL 2020)

# Assignment 2: Cellular automata simulation of traffic including cars and bicycles

## By Maaz Jamal

## ID # 101127351

# Introduction:

In this assignment we implement a cellular automaton to simulate the flow of traffic. The Paper implements many cases for the road conditions, comprising of turns, narrow roads, conflict areas and intersections. This assignment implements the narrow road condition outlined in the paper.

# Paper Description:

In this paper, the cellular automata simulations are used to define tracks and vehicles, including cars and bicycles and interactions between them. The paper creates a spatial model to represent the road. The spatial model is composed of following constructs:

* The most basic representation is one-dimensional cellular-automata space called a *track.* The *track* is composed of numbered cells indicating direction and size to indicate the allowed vehicle types.
* *Turns* are spatial constructs to indicate a change in direction of the vehicle.
* *Conflict Construct* are points where more than one vehicle contend for access to a location.
* *Track relationship* defines the geographical relationship between two close tracks i.e. are they joined or do they have space between them.
* *Divergence* defines two tracks that diverge after travel in one direction after a certain distance.

Vehicles:

The vehicles move along a 1-D cellular spatial automata and do not change tracks. For example, if there a turn on the road, there will be two tracks, one that goes straight and the other that has a turn, superimposed on each other. The vehicle motion is based on Nagel–Schreckenberg rules.

The Nagel–Schreckenberg rules were developed to simulate traffic flow using cellular automata. In its original form the Nagel–Schreckenberg rules comprise of:

* *Acceleration* :
* *Deceleration:*
* *Randomization* :
* *Motion:* Each vehicle is advanced units across the track

Each of these rules is applied to the cells before the cell updates in order listed. The paper sets the maximum speed of bikes to 2 and of vehicles to 3. The paper modifies the deceleration parameter of the original Nagel–Schreckenberg rules by defining custom speed limits to the cars so that they reduce their speed gradually instead of sudden change implied from the original paper.

Each of the different construct impose different speed limits on the cars. However, no speed limit changes are applied on the bicycles. The road width is defined as narrow or wide. Narrow road mean that the car and bicycle both share a narrow road, and the car has to adjust its speed based on its proximity to the bicycle. In wide road the car does not have to adjust its speed based on its proximity to the bicycle.

The limits imposed by the bicycle for the narrow road condition on the cars are show in table 1.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Distance | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Speed Limits | No limit | 2 | 2 | 2 | 1 | 1 | 1 |

Table 1. Speed limits imposed on car due to Vehicles. [1]

The paper implements and test two scenarios:

* Road Stretch with closed boundary conditions
* Road with left turn.

This assignment will replicate the first scenario and show the simulations for the different conditions.

## Implementation Changes:

Changes were made to the implementation as described in the paper to fit the constraints of cd++ tool. The first change made was in the implementation of the Nagel–Schreckenberg rules. The rules are all executed before the time step can happen. However, attempting to implement each change per time step can result in the simulation not proceeding. The circumvent this. The rules are modified so that each rule can only occur if it is possible. The rules then become in simple form.

* *Acceleration* :
* *Deceleration:*

The probability of reducing speed is applied to the acceleration and motion rules. In this way the model is able to simulate the rules correctly.

The second change is to the speed limits imposed on the vehicle by the bike. In the paper each vehicle cell is equivalent to two bike cells. As shown in fig 1.

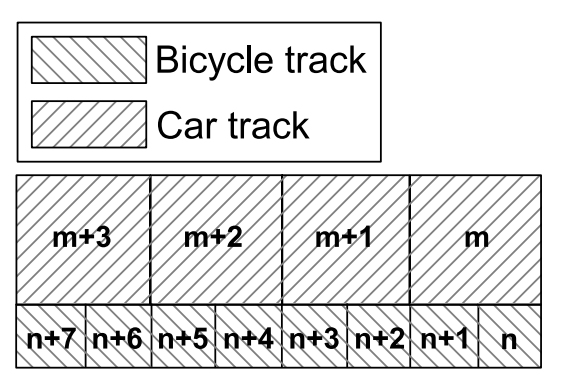


Fig 1. Car and Bike tracks. [1]

This is not easy to implement in CD++ tool as described in the paper. The simpler solution to this problem was to double number of cells and double the speed of the car, while leaving the velocity of the bicycles unchanged. The new possible speeds of the car then become 0,2,4 and 6 respectively. This also modifies the limit imposed by the bicycles on the car. The new limits are shown in table 2.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Distance | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Speed Limit | No limit | 4 | 4 | 4 | 4 | 4 | 4 | 2 | 2 | 2 | 2 | 2 | 2 |

Table 2. Updated limits on Cars due to bicycles.

With these changes the model can be implemented.

# Model Specifications:

The model formal specifications for the model are described below:

## General Rules:

In this section we will list the general rules that were used to implement the model in CD++ . All the other rules are variation of the these rules.

### Acceleration:

rule : {if(random < 0.9,111,114)} 1 { cellPos(0)=0 and (0,0)=-1 and ((0,-1)=1 and (0,1)=-1 )}

This rule implements acceleration from 1 to 2. This is done by an empty cell represented by -1. This cells checks from its neighborhood of cells the cell behind if it is occupied. If it is occupied and the next cell is unoccupied. Then the current cell sets its value to 114 as a flag for this is the next position of previous cell. The if condition implements the cases for random slowing down of the vehicle.

rule : {if(random < 0.9,101,100)} 1 { cellPos(0)=0 and (0,0)=0 and ((0,1)=-1 )}

This rule for condition where the current speed of vehicle is 0 is modelled. The rule checks the current value and determines if the next cell is free. If the next cell is free. The cell changes value to flag 101. This is a special case as in this case the occupied cell changes value as opposed to all other cases where an unoccupied cell changes value.

### Slowing Down:

rule : {102} 1 { cellPos(0)=0 and (0,0)=-1 and (((0,-1)=1 or (0,-1)=2 ) and ((0,1)>-1 and (0,1)<3 ))}

This rule implements slowing a vehicle down from 1 or 2 to 0 in case the next cell to the future occupied cell is occupied. The flag 102 representing a 0 is set in the unoccupied cell.

### Default Advance Cells:

rule : {110} 1 { cellPos(0)=0 and (0,0)=-1 and ((0,-1)=1)}

This rule implement advancing of the cell if no other condition is satisfied. This case represents movement of 1 cell per time step. The next cell is unoccupied cell is set to flag 110, which represent 1 from this case.

### Clearing the previous cell:

rule : {-1} 0 { cellPos(0)=0 and (((0,0)=1 ) and (0,1)=110) }

This rule clears the previous occupied cell and sets it to -1 to represent unoccupied. This case is for the default advance condition. Once the next time step occurs after default advance cell. This rule set the previous cell to -1 without updating the time step. All the condition have unique flags to allow for special conditions to clear each case. It also help in debugging the model.

### SET NEW CELL VALUES:

rule : { 0} 0 { cellPos(0)=0 and ((0,0)=100 or (0,0)=102)}

This rule without updating the timestep. Sets the flag values to the appropriate state values. In this case the flags represent occupied 0 and therefore the cell value is updated to 0.

# Simulation Results:

In this section, we will discuss some of the simulations carried out and the results of the simulations.

## Bicycle Slowing and Acceleration:

In this simulation as shown in fig 2. there are three bicycles at the close to each other and they speed up one by one as the cell become unoccupied in front of them until all three reach the maximum speed. The bicycles randomly slow down as well.



Starting close together



Accelerating Away.



Moving at max speed.

Fig 2. Bicycle simulation showing.

The simulation can be found in “BikeSlowAccelerateRandomSpeed.webm” file and the values are found in the “BikeSlowAccelerateRandomSpeed.val” file.

## Cars Slowing and Accelerating:

Starting close together



Car 1,2 are moving at 6 and 4 units per time step while car 3 is still stationary





All 3 cars are moving with two at max velocity and one yet to accelerate to that velocity



All cars moving at max velocity

Fig 3. Simulation showing cars slowing and accelerating

In this simulation three cars can be seen in the stopped in place. The car that has no obstructions accelerates and move to new cell. Followed by the second car and finally the third. In the end all three cars are moving at constant maximum speed.

The simulation can be found in “CarSlowStopAccelerate.webm” file and the values are found in the “CarSlowStopAccelerate.val” file.

## Car and Stationary Bike Interaction:



One Car accelerated to max, second car is slowed down further due to closer proximity

One car left the bike, second car is slowed down





Both Car reach max speed. 

Car loop back and are slowed down again by bicycles 



Fig 4. Simulation showing interaction between car and stationary bicycle.

In this simulation two bicycles are stationary, represented by the grey blocks and the cars pass near them. The car slow down as they get closer to the bikes. Once they cross the bicycles the cars accelerate to top speed and then loop back and are slowed down again by the bicycles.

The simulation can be found in “CarbikeSlowDown.webm” file and the values are found in the “CarbikeSlowDown.val” file.

## Car and Bike Interaction:



Car and bicycle have same speed so do not cross each other

Car slow down as they approach a bicycle. Bicycle in top row



Bike randomly slows down allowing car to move ahead and accelerate to max speed.



Fig 5. Simulation showing moving bicycle and three cars.

In this simulation there is one moving bicycle and three moving cars. The cars are behind thee bicycle so match their speed with the bicycle and do not cross it. The bicycle randomly slows down allowing a car to leave and accelerate. The other two cars are still moving behind the bicycle.

The simulation can be found in “CarAndBike.webm” file and the values are found in the “CarAndBike.val” file.

## Alternative Method of Implementation:

This assignment implements the model using 2-d Cell-Devs model. However, this assignment can also be implemented as a 3-d Cell-Devs. In this method , One 2-D layer will represent the road while the second hidden 2-D layer can be used to store more information for each cell. Information such as the bicycles region of influence. This may simplify the rules needed to implement the model and may reduce the chances for errors.

# Conclusion:

In this assignment modified Nagel–Schreckenberg rules were implemented in cd++. The rules were modified to allow simulation with the tools. Simulation results were obtained for different scenarios. The different scenarios mimic the flow of traffic and can show congestions, shockwave congestion jams and car behaviour around bicycles.

# References:

[1] J. Vasic and H. J. Ruskin, “Cellular automata simulation of traffic including cars and bicycles,” Physica A: Statistical Mechanics and its Applications, vol. 391, no. 8, pp. 2720–2729, Apr. 2012, doi: 10.1016/j.physa.2011.12.018.