ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Lecture with Computer Exercises:
Modelling and Simulating Social Systems with MATLAB

Project Report

Zurich traffic simulation Sihlstrasse/Uraniastrasse

Filip Meier & Sebastian Honegger

Zurich
December 2015

Agreement for free-download

We hereby agree to make our source code for this project freely available for download from the web pages of the SOMS chair. Furthermore, we assure that all source code is written by ourselves and is not violating any copyright restrictions.

Filip Meier

Sebastian Honegger

Contents

1 Abstract **4**

2 Individual contributions **4**

3 Introduction and Motivations **4**

4 Fundamental Questions **5**

5 Description of the Model **5**
- 5.1 Nagel-Schreckenberg-model 5
- 5.2 Chowdhury-Schadschneider-Modell 8

6 Implementation **10**
- 6.1 Main simulation 10
- 6.2 Chowdhury-Schadschneider-Model-implementation 11

7 Simulation Results and Discussion **12**

8 Summary and Outlook **12**

1 Abstract

The following report shows a simulation for a specific street in Zurich (Uraniastrasse/Sihlstrasse) which should be changed in the future to a pedestrian area (Sihlstrasse). The simulation is implemented with MATLAB and we used the *Nagel-Schreckenber-model* as the basic simulation model. To implement the traffic lights for city traffic we used finally the *Chowdhury-Schadschneider-Model*. The results for the traffic jam in the neighbourhood is much higher than today. We get traffic jam value which are up to 6 times higher than today.

2 Individual contributions

In our project, both of us were active and contributed in the development of the simulation. We worked together at the program and at the report.

3 Introduction and Motivations

The city of Zurich planing to change one specific part of the Sihlstrasse in a pedestrian area. The idea is to make this area more comfortable for the visitors of the city center and it should be also an upgrade for the restaurants and shops around this area.

Figure 1: right: situation plan which shows the change of the tracks. left: illustraion of the pedestrian area at sihlstrasse.[1]

The change will have a big impact for the traffic because there will be one lane less than before. Sihlstrasse (from west to east) and Uraniastrasse (from east to west) is one of the most travelled road in the city center. It is the only alternative road to the highway (Westumfarung). If they decide to built a pedestrian area in the Sihlstrasse they will lose one track from west to east[1]. We know want to analyse the impact to the traffic jam and the impact on the neighbourhood streets.

4 Fundamental Questions

With our simulation, we want to answered the following questions:

1. Are the streets still large enough to manage the traffic jam peaks on working days?
2. What is the impact on the neigbourhood streets?

5 Description of the Model

5.1 Nagel-Schreckenberg-model

Our model is based on the prototype of cellular automata model which is called *Nagel-Schreckenber-model*. It was developed by Kai Nagel and Michael Schreckenberg in 1992. The basic idea was to split the streets in cells, which contain only one car. Therefore we can identify on cell with the typical required space for one car in a traffic jam. Generally this length is around 7.5 m, which correspond approximately the length of the car and the average distance to the car in front in a traffic jam. Figure (2) shows a typical *Nagel-Schreckenberg-model* set-up, one cell (approx. 7.5 m)

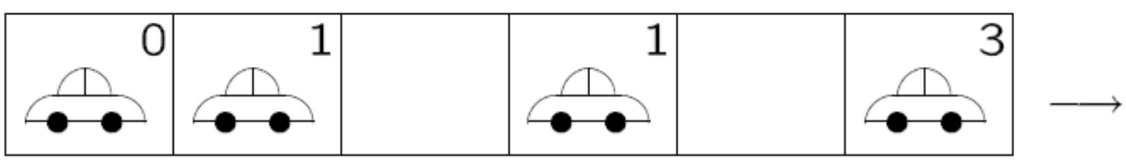


Figure 2: this figure illustrate the typical *Nagel-Schreckenberg-model* configuration: one cell (approx. 7.5 m) which can include exactly one car.[2]

which include one car. The number in the upper right corner of the cell (which include a car) representing the actual velocity v_n of the car n. The velocity is a discrete value an we assume that the car n can have the velocities $v_n = 0, 1 \ldots, v_{\max}$. Every car has the same $v_{\max}$ which has the same effect like a speed limit.
With this properties one have a good description of the state of the street at time t. The next step is to define the development in time. So we have to define the state of

the street at the time $t+1$. To simulate this time step in the *Nagel-Schreckenberg-model* we have to define for steps, which we have to apply for each car n:

1. **Acceleration**

   If $v_n < v_{\max}$ at time t, the car n will accelerate is velocity about one unit:

   $$v_n \rightarrow v_n' = \min(v_n + 1, v_{\max}) \tag{1}$$

   v_n' represents the new velocity at time $t+1$.

2. **Slow down**

   We define d_n as the number of empty cells in front of the car n until to the next car $n+1$. So if d_n is smaller than v_n', the car n has to slow down to the velocity d_n:

   $$v_n' \rightarrow v_n'' = \min(v_n', d_n) \tag{2}$$

3. **Randomization**

   If $v_n'' > 0$, the velocity of car n will be randomly with the probability p reduced about one unit:

   $$v_n'' \rightarrow v_n''' = \begin{cases} \max(v_n'' - 1, 0) & \text{with probability} \quad p \\ v_n'' & \text{with probability} \quad 1-p \end{cases} \tag{3}$$

4. **Drive**

   The car n drives with the new velocity $v_n(t+1) = v_n'''$ about $v_n(t+1)$ cells:

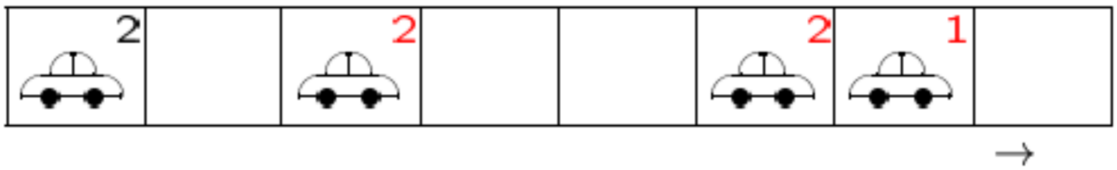
   $$x_n(t+1) = x_n(t) + v_n(t+1) \tag{4}$$

One have to apply every step simultaneous to every car. So we can not simulate the real situation, that the car in front can move as well simultaneous to the car behind. One can see that just step (2) has a interaction between cars and with step (3) the simulations has a stochastic dynamic. Therefore the *Nagel-Schreckenberg-model* is called a stochastic cellular automata.

Figure (3) shows a complete time step of the *Nagel-Schreckenberg-model*. In this case in step 4 we have three car which are able to hang behind but just one will to it in the next step (probability $p = \frac{1}{3}$). And as one can see the speed limit in this example is $v_{\max} = 2$.

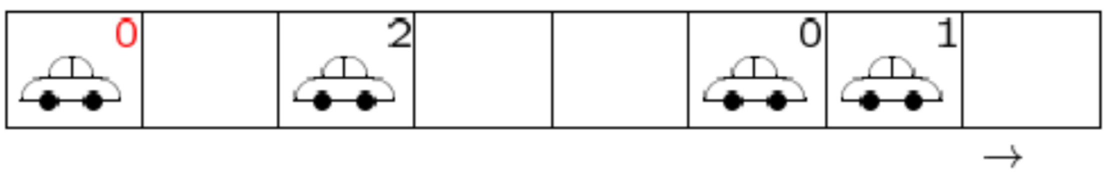
1. **start configuration**

2. **acceleration** ($v_{\max} = 2$)



3. **slow down**

4. **Randomization** ($p = \frac{1}{3}$)



5. **drive** (= configuration at $t + 1$)

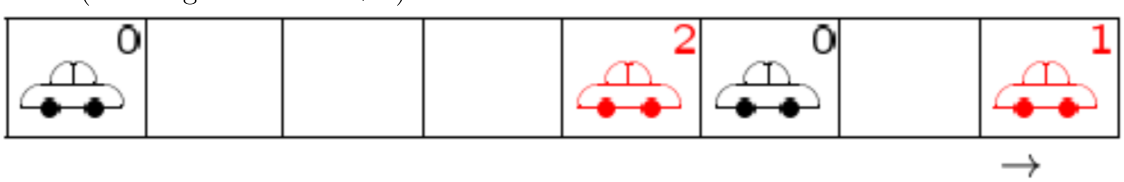


Figure 3: Shows on complete time step of the *Nagel-Schreckenberg-model* with acceleration, slow down, hang behind probability and drive.

1. The first step in figure 3 shows that all cars want to accelerate as soon as possible to maximum speed limit.

2. the 'slow down' step (figure 3) prevent car accidents. But as mention before it does not include the movement of the car in front at the same time.

3. The step Randomization simulates different effects. It tries to modelling for example natural fluctuations in driving style. A normal car drives never with a total constant speed it always fluctuate between $v_{\max}$ und $v_{\max} - 1$. This function generates an asymmetry for the acceleration and the slow down mode. In detail it will generate a stronger slow down and sometimes the speed stays constant instead to accelerate. If we have a high density of cars, the above described effects could have a traffic jam as result.

4. The last step is just the motion of the cars with the new speed which is compute with the steps 1 to 3.

We already defined the length of one cell (7.5 m) but we also have to define the correct time scale for one time step $t \to t+1$. Therefore we need a value for the average speed, one can calculate it in the following way:

$$v_{\text{aver}} = (1-p)v_{\max} + p(v_{\max} - 1) = v_{\max} - p \tag{5}$$

We now identify this speed with 50 km/h. For a model with $v_{\max} = 5$ and $p = 0.5$ we get

$$\frac{7.5\,\text{m}}{\text{cells}} \cdot \frac{4.5\,\text{cells}}{\text{timestep}} \cdot \frac{3.6\,\text{s}}{50\,\text{m}} \approx 2.5\,\frac{\text{s}}{\text{timestep}} \tag{6}$$

5.2 Chowdhury-Schadschneider-Modell

The dynamic of a real city traffic is characterized through the interaction of two time scales, the driving time between two traffic lights and the time of the green phase of each traffic light. Chowdhury and Schadschneider have modify the *Nagel-Schreckenberg-model* with a algorithm for traffic lights to simulate the traffic of a city. The dynamic is define by the following steps. d_n define as before the gap to the next car and s_n defines the distance to the next traffic light:

1. **Acceleration**

If $v_n < v_{\max}$ at time t, the car n will accelerate is velocity about one unit:

$$v_n \to v'_n = \min(v_n + 1, v_{\max}) \tag{7}$$

v'_n represents the new velocity at time $t + 1$.

2. **Slow down because of cars or traffic lights**

   1. case: The traffic light is red:

   $$v'_n \to \min(v_n, d_n, s_n) \tag{8}$$

   2. case: The traffic light is green:
      a. The traffic light get red in the next time step:

      $$v'_n \to \min(v_n, d_n, s_n) \tag{9}$$

      b. The traffic light is not getting red:

      $$v'_n \to \min(v_n, d_n) \tag{10}$$

3. **Randomization**

   If $v''_n > 0$, the velocity of car n will be randomly with the probability p reduced about one unit:

   $$v''_n \to v'''_n = \begin{cases} \max(v''_n - 1, 0) & \text{with probability} \quad p \\ v''_n & \text{with probability} \quad 1 - p \end{cases} \tag{11}$$

4. **Drive**

   The car n drives with the new velocity $v_n(t + 1) = v'''_n$ about $v_n(t + 1)$ cells:

   $$x_n(t + 1) = x_n(t) + v_n(t + 1) \tag{12}$$

The simulations is exactly like the *Nagel-Schreckenberg-model* just the interaction with the traffic lights in step 2 is new. If we have a red traffic light the has to stop before. Therefore the velocity v_n of car n should not be higher then the distance s_n to the next traffic light. So the car will be affected by the traffic light if $s_n < v_n$.
If the traffic light is green we can simulate a kind of orange phase. If the traffic light gets red in the next step, the cars will slow down immediately. If the traffic light not get red the simulation will just look on the distance to the next car.

6 Implementation

6.1 Main simulation

In the main script we define all global parameter: time steps and $v_{\max}$. Then we have to define the streets for each direction through an array. One cell represents around 7.5 m and can be occupied only by one car. We then define the location of the cells where the traffic lights are placed. Therefore we used a second array which has the information of the location for the different traffic lights and the current status (green or red). We measured the whole street and the locations of the traffic lights in google maps.
For convenience we neglected several small cross streets around the area of sihlstrasse and uraniastrasse.
To simulate one hour we call different functions in a loop. The important ones are the following:

1. We apply the *Chowdhury-Schadschneider-Model* for the current street status.
2. We add cars on the beginning of the street. Therefore we use an algorithm which works like a traffic light.
3. We take the cars away from the street at the end of it. We also use an algorithm which works like a traffic light.
4. To keep the actual status of the street, we add it in a matrix.

In figure 4 you will see an visualization of this process in form of matrix. Figure 4 shows the current status with two lanes in each direction. The more redish cars are the fast ones and the more yellow cars are the slow ones and the black cars have no movement at the moment. The upper figure has time axis from top to bottom and the street from right to left. And the one above has time axis from top to bottom and the street from left to right.

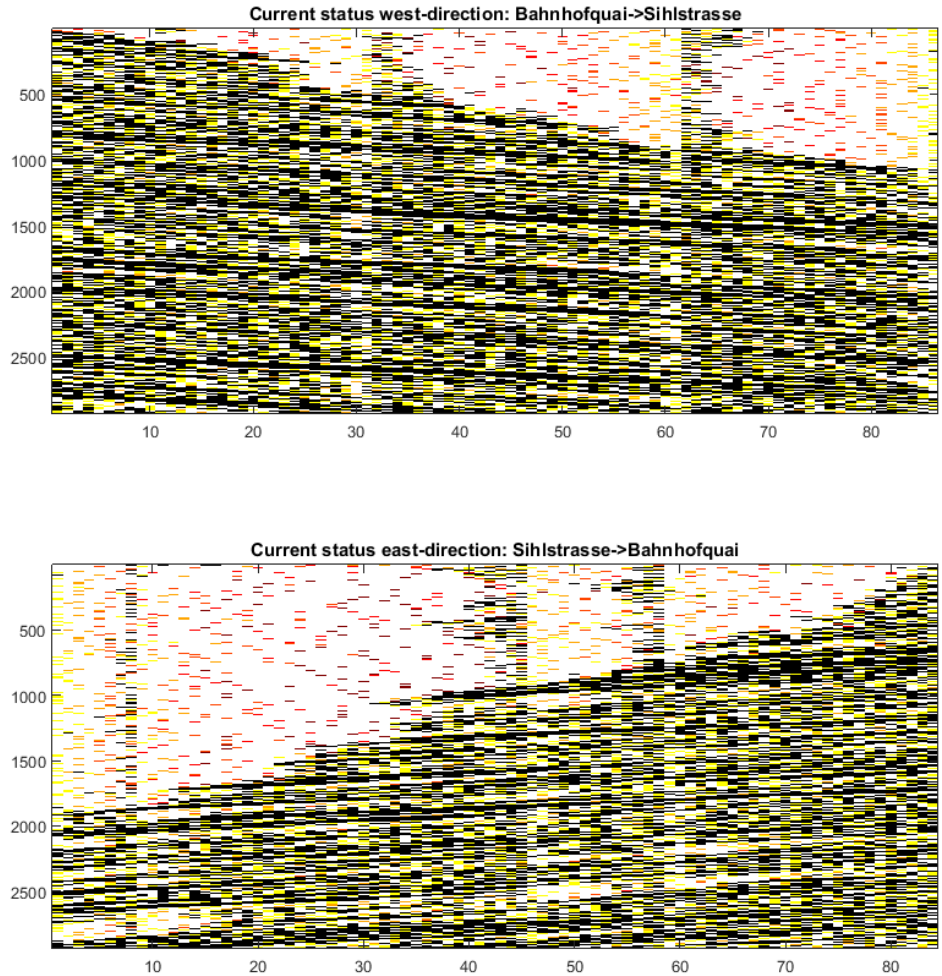


Figure 4: Visualization over a long duration (1h) of the two lane street (current status) in both directions around 6pm. The timesteps from top to the bottom are 2.5 s. The x-axis represents the streets.

6.2 Chowdhury-Schadschneider-Model-implementation

For the different steps in the *Nagel-Schreckenberg-model* we defined one function for each step, figure 1 for example shows the acceleration function.

For this function we just use the minima function of matlab. Input is the current velocity v_n of the car and the maximum velocity $v_{\max}$.
The most difficult function is the slow down function were we have to implement the two slow down reasons: the next traffic light or the next car. For the distance to the next car d_n we defined first $d_n = 0$ and then we implemented a loop over the length of the whole street to check where the next car is located.
For the traffic lights we made in a similar way. First of all we have to check which traffic light is the next one in front of the car. Then we can analyse the traffic light: is it currently green or red and is it going to be red in the next step. If it is green we can ignore it, therefore we set s_n , which is the distance to the next traffic light, to a high value. Like this it will not influence the car in the next time steps. If it is red we have to compute the exact value for s_n.
Finally we get the values for d_n and s_n. Now we just can define the minima of v_n, d_n and s_n as mention in equation 9. This will be the value of our output of the slow down function.
For the probability function we used simply the function rand from matlab, which define randomly a number between 0 and 1 and then we just can define a probability p and an if command.

7 Simulation Results and Discussion

We simulated first of all the current status of the street. The city of zurich [3] sent us data of the average traffic for 24 hours. With our simulation we now can determine the traffic jam ahead the first traffic light. Figure 5 shows the traffic jam in cars per unit. As you can see at the typically traffic jam hours (morning time and evening time) we got peak-values of traffic jam.
Figure 7 shows the new situation were we will have just one lane in east direction. The traffic jam is much higher than before, on figure 6 one can compare it. The extension is very long, it is in some hours 6 times higher. This will have huge affects for the neigbourhood.

8 Summary and Outlook

In the project of the city of zurich [1] the project manager wrote that they have to make some accompanying measures for the neighbourhood area. They do not give explicit suggestion how they look like. Our simulations shows that if they cancel one lane in direction east it will have a huge impact on the traffic jam. One have to handle a car volume which is 6 times higher than before. In our opinion they have

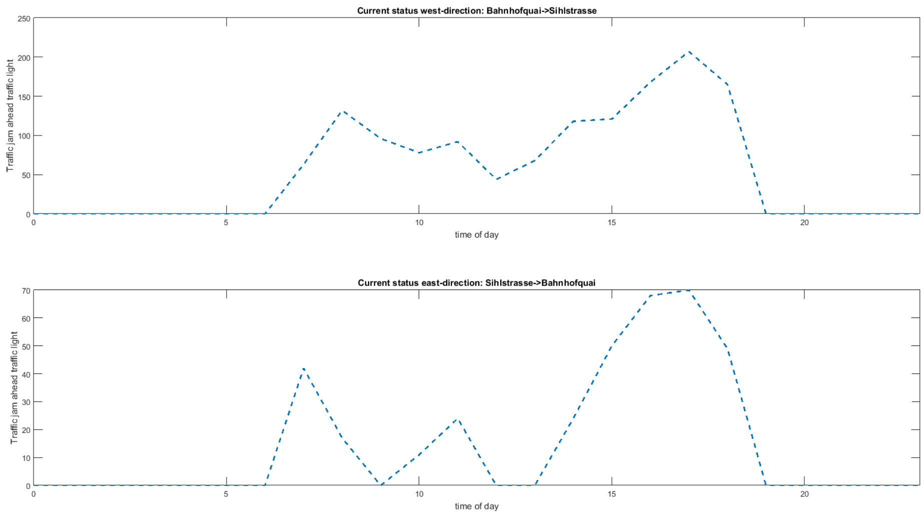


Figure 5: This figure shows the traffic jam on a full 24h day for the two lane street (current status). One can see the traffic jam peaks in the morning hours and the evening hours.

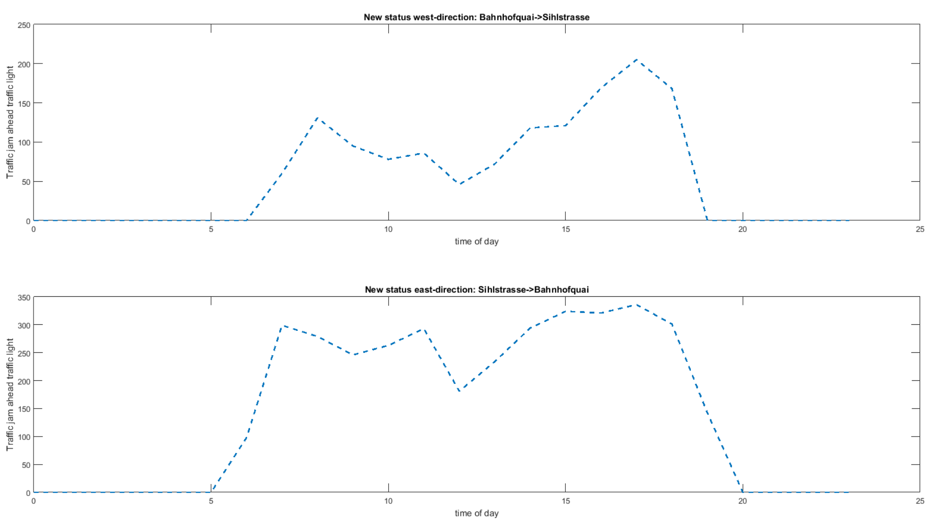


Figure 6: This figure shows the traffic jam on a full 24h day for the one and two lane street (future status). One can see the traffic jam peaks in the morning hours and the evening hours.

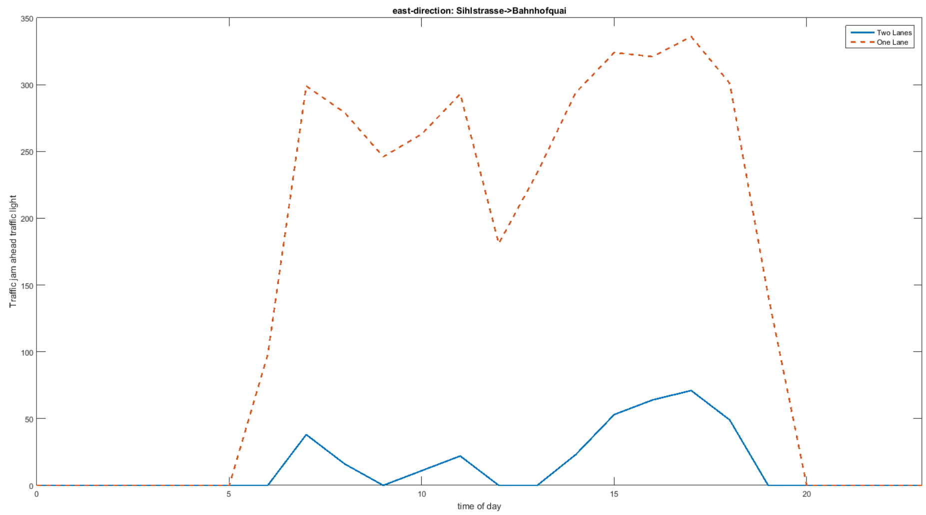


Figure 7: This figure shows the traffic jam on a full 24h day for the one and two lane street (future and current status) of the Sihlstrasse. One can see that the traffic jam is much higher with just one lane in this direction.

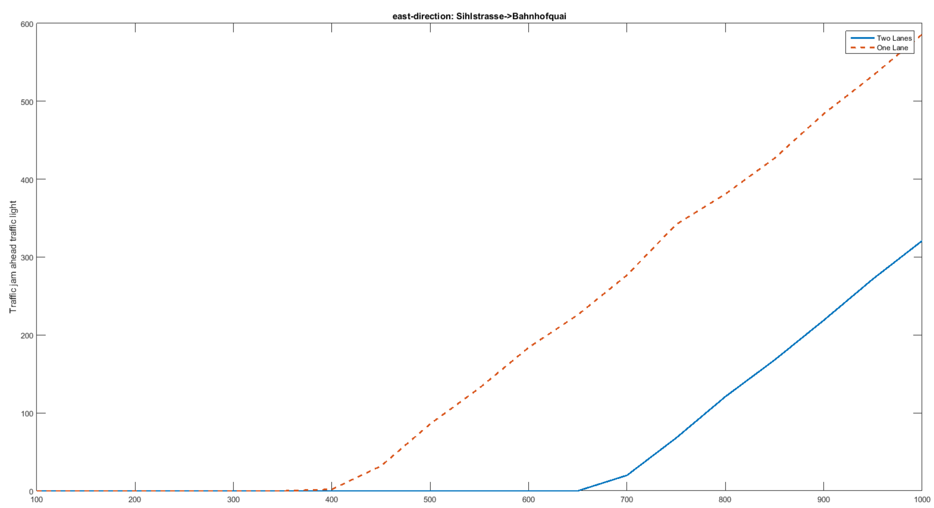


Figure 8: This figure shows the traffic jam from 100 cars until 1000 cars per hour.

to change a lot in this neighbourhood areas.
But to improve our model we should try to analyse what is the best set-up for the red and green cycle of the traffic lights. We think like this we can minimize the traffic jam.
Our simulation is very flexible, we can change all parameters very easily, therefore we can use it for different streets or different cities.

References

[1] Neue Verkehrsorganisation Uraniastrasse (project booklet), *Tiefbauamt Stadt Zuerich,* http://www.stadt-zuerich.ch/

[2] Physik des Straenverkehrs, *Andreas Schadschneider,* http://www.thp.uni-koeln.de

[3] Tiefbauamt Stadt Zrich, *Roland Frei,* Projektleiter Infrastruktur + Raum, roland.frei@zuerich.ch

[4] Tiefbauamt Stadt Zrich, *Gian Doenier,* Chef Verkehrssteuerung / Stv. L VM, gian.doenier@zuerich.ch

Matlab Code

```


%Input Bahnhofquai, Werte gem ss Z hlstelle und Extrem Wert
IBQ =
    [154,116,91,71,84,184,483,745,817,782,762,773,729,754,804,807,855,892,8
%Input Sihlstrasse, Werte gem ss Z hlstelle
ISS =
    [115,74,52,46,51,128,508,719,698,656,691,706,607,652,704,732,746,751,72

n = length(IBQ);



%x_modus == place cars on 1 or 2 lanes
e_modus = 2;
w_modus = 2;
modus = [w_modus, e_modus];


%simulation mit der n tigen Anzahl Autos starten
%West_direction = [Input_Count_Bahnhofquai,
    Warteschlange_Bahnhofquai, Output_Count_Sihlstrasse,
    car_counter_w(1,1), car_counter_w(2,1)];
%East_direction = [Input_Total, Warteschlange_Total,
    Output_Count_Bahnhofquai, car_counter_o(1,1),
    car_counter_o(2,1)];
WD = [];
ED = [];
for i = 1:n

    [West_direction, East_direction] = test(IBQ(i), ISS(i),
        modus);
    %             Input_Count,           Warteschlange
    WD = [WD; West_direction(1), West_direction(2)];
    ED = [ED; East_direction(1), East_direction(2)];
```

```
end


results=figure;

subplot(2,1,1)
plot( 0:length(IBQ)-1, WD(:,2)','--','linewidth',2 );

title('Current status west-direction: Bahnhofquai-> Sihlstrasse')
xlabel('time of day')
ylabel('Traffic jam ahead traffic light')

subplot(2,1,2)
plot( 0:length(ISS)-1, ED(:,2)','--','linewidth',2);
title('Current status east-direction: Sihlstrasse-> Bahnhofquai')
xlabel('time of day')
ylabel('Traffic jam ahead traffic light')

hold off
```

```
function [West_direction, East_direction] = test(x_w, x_o, modus)

%clear Workspace
%   clear
%   x_w = 1000;
%   x_o = 1000;
%

%   modus = [2,2];

%globale Parameter
steps = 1470; %Number of timesteps in the simulation
free_road = 255; %value for empty street
v_max = 5; %maximal speed in the model
rho = 0.3; % hang_behind Parameter

```

```
output_w_h = 600;
output_o_h = 600;
w_modus = modus(1);
e_modus = modus(2);



%Datenstruktur der Simulation initialisieren
%Richtung Osten
[traffic_light_o, M_o, map_size_o] = simulation_sihlstrasse(
    v_max, free_road, e_modus );
trafficlight_t_o=15;
redlight_t_o=4;

%Richtung Westen
[traffic_light_w, M_w, map_size_w] =
    simulation_uraniastrasse( v_max, free_road, w_modus );
trafficlight_t_w=15;
redlight_t_w=4;

%change_traffic_light
%function to change the above traffic lights
%input step=i
%tbd... sollte eine Funktion zuweisen
%change_traffic_light = change_traffic_light_test(
    traffic_light, i);

%Data structure for simulation steps M_o and M_w
% two rows represents a timestep (one map update)
% the first (map_size)-colums represents the street, the
    last two stands
% for the car_counter on the two streets
%same for west direction


%T Debuggingdatenstruktur
%T enth lt den Zustand der einzelnen traffic_lights in der
    row-i zum Zeitschritt i
T_o = [];
```

```
T_w = [];

%east direction
%Anzahl Autos in der Warteschlange
Warteschlange_Sihlstrasse = 0;
Warteschlange_Annagasse = 0;
Warteschlange_out_Bahnhofquai = 0;
%Anzahl der generierten Autos
Input_Count_Sihlstrasse = 0;
Input_Count_Annagasse = 0;
%Anzahl gel schter Autos
Output_Count_Bahnhofquai = 0;

%west direction
%Anzahl Autos in der Warteschlange
Warteschlange_Bahnhofquai = 0;
Warteschlange_out_Sihlstrasse = 0;
%Anzahl der generierten Autos
Input_Count_Bahnhofquai = 0;
%Anzahl gel schter Autos
Output_Count_Sihlstrasse = 0;



%run the simulation for east direction
for i = 1:steps
%% west direction
    %Update the Map
    New_Map = map_update(w_modus, M_w(end-(w_modus-1):end,:), map_size_w, free_road, v_max, rho, traffic_light_w);
    %Zufluss zum System zu steuern
    [New_Map, Warteschlange_Bahnhofquai, Input_Count_Bahnhofquai] = add_cars(w_modus, New_Map, i, Warteschlange_Bahnhofquai, Input_Count_Bahnhofquai, free_road, x_w, steps);
    %[New_Map, Warteschlange_Bahnhofquai, Input_Count_Bahnhofquai] = input_bahnhofquai(New_Map, i, Warteschlange_Bahnhofquai, Input_Count_Bahnhofquai, free_road);
```

```
    %Abfl sse
    [New_Map, Warteschlange_out_Sihlstrasse, Output_Count_Sihlstrasse] = delete_cars(w_modus, New_Map, i, Warteschlange_out_Sihlstrasse, Output_Count_Sihlstrasse, free_road, output_w_h, steps);
    %count all cars on the street
    car_counter_w = [0;0];
    for street = 1:w_modus
        for k = New_Map(street,:)
            if ( k ~= free_road )
                car_counter_w(street) = 1 + car_counter_w(street);
            end
        end
    end
    %f gt aktuellen car_counter der Datenstruktur hinzu
    if ( w_modus == 1 ), car_counter_w = car_counter_w(1); end;
    New_Map = [New_Map, car_counter_w];
    %f gt aktuelle Map der Datenstruktur M hinzu
    M_w = [M_w;New_Map];
    %Debugin Plot: jeden Step grafisch anzeigen
     M_plot = New_Map;
     visualization_test( M_plot );
    %Debuggingdatenstruktur mit aktuellem Zustand der traffic_lights f llen
    T_w = [T_w; traffic_light_w(2,:)];

%% east direction
    %Update the Map
    New_Map = map_update(e_modus, M_o(end-(e_modus-1):end,:), map_size_o, free_road, v_max, rho, traffic_light_o);

    %Zufluss zum System zu steuern
    [New_Map, Warteschlange_Sihlstrasse, Input_Count_Sihlstrasse] = add_cars(e_modus, New_Map, i, Warteschlange_Sihlstrasse, Input_Count_Sihlstrasse, free_road, x_o, steps);
```

```
    %[New_Map, Warteschlange_Annagasse, Input_Count_Annagasse] = input_annagasse(New_Map, i, Warteschlange_Annagasse, Input_Count_Annagasse, free_road);

    %Abfl sse
    [New_Map, Warteschlange_out_Bahnhofquai, Output_Count_Bahnhofquai] = delete_cars(e_modus, New_Map, i, Warteschlange_out_Bahnhofquai, Output_Count_Bahnhofquai, free_road, output_o_h, steps);

    %Debuging etc
    %count all cars on the street
    car_counter_o = [0;0];
    for street = 1:e_modus
        for k = New_Map(street,:)
            if ( k ~= free_road )
                car_counter_o(street) = 1 + car_counter_o(street);
            end
        end
    end
    %f gt aktuellen car_counter der Datenstruktur hinzu
    if ( e_modus == 1 ), car_counter_o = car_counter_o(1); end;
    New_Map = [New_Map, car_counter_o];
    %f gt aktuelle Map der Datenstruktur M hinzu
    M_o = [M_o;New_Map];

    %Debugin Plot: jeden Step grafisch anzeigen
    %M_plot = New_Map;
    %visualization_test( M_plot );
    %Debuggingdatenstruktur mit aktuellem Zustand der traffic_lights f llen
    T_o = [T_o; traffic_light_o(2,:)];
```

```
    %change traffic_lights for next timestep
    traffic_light_w = change_traffic_light_uraniastrasse(traffic_light_w, i, redlight_t_w, trafficlight_t_w);
    traffic_light_o = change_traffic_light_sihlstrasse(traffic_light_o, i, redlight_t_o, trafficlight_t_o);
end

%% Ausgabe erstellen
%Matrix bearbeiten
%Feld 1 und car_counter entfernen
%M_plot = Map(:,1:(end-1))
M_w = M_w(:,2:(end-1));
M_o = M_o(:,2:(end-1));

subplot(2,1,1)
rgb_Matrix_w = visualization_test( fliplr(M_w) );
title('Current status west-direction: Bahnhofquai-> Sihlstrasse')

subplot(2,1,2)
rgb_Matrix_o = visualization_test( M_o );
title('Current status east-direction: Sihlstrasse-> Bahnhofquai')
hold off




%%Output Daten
% Input_Count == x (2*15 Autos die am Anfang generiert werden, werden nicht
% mitgez hlt
West_direction = [Input_Count_Bahnhofquai, Warteschlange_Bahnhofquai, Output_Count_Sihlstrasse, car_counter_w'];

%Habe den Input Annagasse momentan gestrichen... den k nnen wir am Freitag
%wieder aktivieren
```

```
Input_Total = Input_Count_Sihlstrasse + Input_Count_Annagasse;
Warteschlange_Total = Warteschlange_Sihlstrasse + Warteschlange_Annagasse;

East_direction = [Input_Total, Warteschlange_Total, Output_Count_Bahnhofquai, car_counter_o'];


end
```

```
function v = acceleration( v_n, v_max )
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here

v=min(v_n+1,v_max); % acceleration to new speed



end
```

```
function v = slow_down(Map, map_size, free_road, traffic_light, v_n, position)
%UNTITLED3 Summary of this function goes here
%   Detailed explanation goes here

% Distance to next car == d_n
d_n = 0;
for k = position+1:map_size
    if( Map(k) == free_road )
        d_n = 1 + d_n;
    else
        break
    end
end

% ineffizient... kann oben mit d_n kombiniert werden
% lasse es aber momentan hier, da m glicherweise next_car und
% next_traffic-light bekannt sein m ssen
```

```
% Distance to next traffic_light = s_n
% Lichtsignale von hinten nach vorne durchgehen

traffic_light_number = 0;
s_n = 1000;

%             t = 1,2,3,4
%       traffic_light = [    20, 40, 60, 80 ;
%                             0,  1,  0,  1   ]

for t = 1:length(traffic_light)
    if ( position <= traffic_light(1,t) )
        traffic_light_number = t;
        break
    end
end

% trifft zu, wenn Auto nach dem letzten Traffic Light ist
% dann "unendlich weit"==1000 fahren
if ( traffic_light_number == 0 )
    s_n = 1000;
else
    position_traffic_light = traffic_light(1, traffic_light_number);
    status_traffic_light = traffic_light(2, traffic_light_number);
    %wenn ==1, dann gr n
    if ( status_traffic_light == 1 )
        s_n = 1000;
    %n chstes Lichtsignal ist rot
    else
        s_n = position_traffic_light - position;
    end
end



v=min([v_n,d_n,s_n]);

```

```
end
```

```
function v = hang_behind( v_n, rho )
%UNTITLED4 Summary of this function goes here
%   Detailed explanation goes here

t=rand;
if t<rho
    v=max(v_n-1,0);
else
    v=v_n;
end

end
```

```
function New_Map = map_update( modus, Map, map_size, free_road, v_max, rho, traffic_light )

%create return datastructure
New_Map = ones(modus,map_size)*free_road;

for street = 1:modus
    for i= 1:map_size
        %for each field that contains a car, do..
        if ( Map(street,i)~= free_road )

            %get car speed
            v_n = Map(street,i);

            %call the acceleration function
            %increases the speed
            v_n = acceleration(v_n,v_max);

            %call the slow_down function function
            v_n = slow_down(Map(street,:), map_size, free_road, traffic_light, v_n, i);

            v_n = hang_behind(v_n, rho);

            %callculate the position on the map, depending
```

```
                  on the new speed
                position = i + v_n;
                %write the new speed into the callculated position
                New_Map(street, position) = v_n;
            end
        end
    end

end
```

```
function [New_Map, Warteschlange, Input_Count] = add_cars(modus, New_Map, i, Warteschlange, Input_Count, free_road, x, steps)
%adds x-many cars in steps (==time steps)

%Soll-Wert der generierten Autos im Schritt i
y_i = i ./ steps * x;
differenz = floor( y_i - Input_Count );
if ( differenz > 0 )
    Input_Count = Input_Count + differenz;
    Warteschlange = Warteschlange + differenz;
end

abc = 1;
if ( modus == 2)
    %p street 1: 1/2
    %p street 2: 1/2
    t=rand;
    %strasse 2 dann 1
    abc = 2:-1:1;
    if ( t < 0.5 )
        %strasse 1 dann 2
        abc = 1:2;
    end
end

for street = abc
    if ( Warteschlange ~= 0 && New_Map(street,1) == free_road )
```

```
            New_Map(street,1) = 1;
            Warteschlange = Warteschlange - 1;
        end
end


end
```

```
function New_Map = add_one_car(New_Map, free_road)
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here

if ( New_Map(1) == free_road )
    New_Map(1) = 0;
end



end
```

```
function traffic_light = change_traffic_light_sihlstrasse(traffic_light, step, redlight_t, trafficlight_t )
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here

%rot alle 0,1
%gr n alle 2,3,4
mod_step = mod(step,trafficlight_t);
for t_l = 1:length(traffic_light)
    if ( mod_step <= redlight_t )
        traffic_light(2,:) = 0;
    else
        traffic_light(2,:) = 1;
    end
end
end
```

```
function traffic_light = change_traffic_light_uraniastrasse(traffic_light, step, redlight_t, trafficlight_t  )
%UNTITLED Summary of this function goes here
```

```
%   Detailed explanation goes here

%rot alle 0,1
%gr n alle 2,3,4
mod_step = mod(step, trafficlight_t);
for t_l = 1:length(traffic_light)
    if ( mod_step <= redlight_t)
        traffic_light(2,:) = 0;
    else
        traffic_light(2,:) = 1;
    end
end
end
```

```
function [New_Map, Output_Count_Bahnhofquai] = output_bahnhofquai(modus, New_Map, Output_Count_Bahnhofquai, i, free_road)

%790 per / 1470 times steps

if ( mod(i,4) == 0 || mod(i,37) == 0)
    if ( New_Map(1,end) ~= free_road )
        Output_Count_Bahnhofquai = Output_Count_Bahnhofquai + 1;
        New_Map(1,end) = free_road;
    end
    if ( modus==2 && New_Map(2,end) ~= free_road )
        Output_Count_Bahnhofquai = Output_Count_Bahnhofquai + 1;
        New_Map(2,end) = free_road;
    end
end

end
```

```
function [New_Map, Output_Count_Sihlstrasse] = output_sihlstrasse(modus, New_Map, Output_Count_Sihlstrasse, i, free_road)

%1300 per / 1470 times steps
```

```
if ( mod(i,2) == 0 )
    if ( New_Map(1,end) ~= free_road )
        Output_Count_Sihlstrasse = Output_Count_Sihlstrasse + 1;
        New_Map(1,end) = free_road;
    end
    if ( modus==2 && New_Map(2,end) ~= free_road )
        Output_Count_Sihlstrasse = Output_Count_Sihlstrasse + 1;
        New_Map(2,end) = free_road;
    end
end

end
```

```
function [tl, map, map_size, car_count ] = simulation_sihlstrasse_new( v_max, free_road )
%Datenstruktur der Simulation initialisieren

%traffic_light
%traffic_light ist eine 2xN Matrix f r N Lichtsignale. %0=red, 1=green
tl = [    1, 9, 46, 59;
          0,  0,  0, 0 ];

%Map generieren
%Parameter
map_size = 85;
%free_road = 255; %value for empty street
car_count = 30; %initial car count
%v_max = 5; %maximal speed in the model
%rho = 0.3; % hang_behind Parameter


%Werte aus Google-Maps passend zur map_size 85
%Sihlstrasse Werte in m/Position
%A1 Position: 0/1
%A2 Position 71/9
%St. Annagasse: 173/23
```

```
%Abzweigung: 248/33
%A3  Position 344/46
%A4 Position: 440/59
%A5 Position: 636/85


map = ones(2,map_size)*free_road;

%Optional: place cars on the street
for street = 1:1
    for i = 1:car_count
        while (1)
            position=floor(rand*(map_size));
            if ( position ~= 0 && map(street,position) == free_road )
                map(street,position)=round(rand*v_max);
                break
            end
        end
    end
end
```

```
function [tl, map, map_size] = simulation_sihlstrasse_one_line( v_max, free_road, modus )
%Datenstruktur der Simulation initialisieren

%traffic_light
%traffic_light ist eine 2xN Matrix f r N Lichtsignale. %0=red, 1=green
tl = [    1, 29, 59;
          0,  0,  0];

%Map generieren
%Parameter
map_size = 85;
%free_road = 255; %value for empty street
car_count = 15; %initial car count
%v_max = 5; %maximal speed in the model
%rho = 0.3; % hang_behind Parameter

```

```
%Werte aus Google-Maps passend zur map_size 85
%Sihlstrasse Werte in m/Position
%A1 Position: 0/1
%A2 Position 71/9
%St. Annagasse: 173/23
%Abzweigung: 248/33
%A3  Position 344/46
%A4 Position: 440/59
%A5 Position: 636/85


map = ones(modus,map_size)*free_road;

%Optional: place cars on the street
for street = 1:modus
    for i = 1:car_count
        while (1)
            position=floor(rand*(map_size));
            if ( position ~= 0 && map(street,position) == free_road )
                map(street,position)=round(rand*v_max);
                break
            end
        end
    end
end
if (modus==2)
    map = [map,[car_count;car_count]];
else
    map = [map,car_count];
end
```

```
function [tl, map, map_size] = simulation_uraniastrasse( v_max, free_road, modus )
%Datenstruktur der Simulation initialisieren

%traffic_light
%traffic_light ist eine 2xN Matrix f r N Lichtsignale. %0=red, 1=green
```

```
tl = [    1, 26, 56;
          0,  0,  0];

%Map generieren
%Parameter
map_size = 87;
%free_road = 255; %value for empty street
car_count = 15; %initial car count
%v_max = 5; %maximal speed in the model
%rho = 0.3; % hang_behind Parameter


%Werte aus Google-Maps passend zur map_size 87
%Uraniastrasse Werte in m/Position
%A5 Position: 0/1
%A4 Position 210/26
%A6 Position: 420/56


map = ones(modus,map_size)*free_road;

%Optional: place cars on the street
for street = 1:modus
    for i = 1:car_count
        while (1)
            position=floor(rand*(map_size));
            if ( position ~= 0 && map(street,position) ==
               free_road )
                map(street,position)=round(rand*v_max);
                break
            end
        end
    end
end

if (modus==2)
    map = [map,[car_count;car_count]];
else
    map = [map,car_count];
```

```
end
```