ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Lecture with Computer Exercises:

Modelling and Simulating Social Systems with MATLAB

Project Report

Ticket inspection behaviour and its influence on fare evasion

Lukas Bischofberger & Jens Ammann

Zürich,

December 12

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of Originality

This sheet must be signed and enclosed with every piece of written work submitted at ETH.

I hereby declare that the written work I have submitted entitled

Ticket inspection behaviour and its influence on fare evasion

is original work which I alone have authored and which is written in my own words.*

Author(s)

Last name	First name
Ammann	Jens
Bischofberger	Lukas

Supervising lecturer

Last name	First name

With the signature I declare that I have been informed regarding normal academic citation rules and that I have read and understood the information on 'Citation etiquette' (http://www.ethz.ch/students/exams/plagiarism_s_en.pdf). The citation conventions usual to the discipline in questic here have been respected.

The above written work may be tested electronically for plagiarism.

L. Bischofberger

J. A

10.12.12
Place and date

Signature

*Co-authored work: The signatures of all authors are required. Each signature attests to the originality of the entire piece of written work in its final form.

Print form

Agreement for free-download

We hereby agree to make our source code of 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.

Lukas Bischofberger

Jens Ammann

ABSTRACT

An agent based model, that represents the interplay of fare inspection and evasion behaviour, was successfully implemented on the tram net of Zürich. The main two assumptions of model are that the probability of fare evasion decreases exponentially with the continuous series of previous uninspected legal trips. Once a passenger is fare dodging he continues until he is caught. The passenger movement turned out to be most realistic when using the shortest path algorithm, as this leads to the expected distribution with higher concentration in the centre of the net. Furthermore it was found the most efficient inspector behaviour is random movement on the net, since this ensures a spatial and temporal even distribution on the net. In contrast matching the inspector distribution to the passenger distribution on the net doesn't show any influence on the fare evasion rate. The simulation was then set to the inspection rate and dodging fraction measured in the net of Zürich. Under these input parameter, the passenger in average start dodging after a continuous series of 278 uninspected legal days in the public transportation net.

TABLE OF CONTENTS

1 INTRODUCTION

Public transportation is of outstanding importance for your society since it ensures the mobility for a brought clientele, independent of social state and physical condition. In order to preserve public transportation collecting fare revenue is essential to provide service, especially in a tough fiscal environment with limited resources. In the public transportation net of Zürich (VBZ) for example just about one per cent of the passengers are evade their tickets [1]. This might appear small but considering the annual 310 million trips in the VBZ net it sums up to 3.8 million unpaid trips a year. This is fairly low, comparing for example to a survey from the San Francisco Municipal Transportation Agency that found a 9.5 per cent fare evasion rate [2]. Even though the vast majority pays the suitable fare, those who do not pay frustrate other customers and reduce the financial resources available to operate comprehensive and reliable transit.

The payment systems for public transportation vary from application. In the proof-of-payment system the passenger are supposed to provide a recipe of payment trough their trip. This system requires additional controlling personal but it may reduce dwell time, potentially resulting in higher ridership and fewer vehicles to maintain service levels. But a drawback of this system is the potential of fare evasion. Although the alternative fare collection systems with operator fare verification and gate station entrance do fully guarantee compliance, it requires an expensive set of facilities.

In Switzerland the prove-of-payment system is dominating. It requires a sophisticated controlling system that holds the evasion rate low. The main strategy to face this problem is random fare inspection and expiations for those who do not have an appropriate ticket. The profit of the fines is not sufficient to cover the personal costs for these inspections but since the inspections are undeclared it forces the passenger into buying tickets.

In Zürich the inspectors are working with an electrical data support. They register every fare dodger in a database. According to VBZ this data base is used to detect the areas with high fare evasion rate and therefor enables the inspection to reinforce their work at the appropriate districts. If the inspectors are not present in a certain area, fare dodging increases instantly [1]. This leads to the assumption that the inspector behaviour influences fare dodging.

The aim of this paper is to simulate the interplay between the inspector and fare dodger with an agent based model. Several inspector behaviours are implemented and their influence on fare evasion is investigated.

2 MATERIAL AND METHODS

In the simulation two different kinds of agents represent passengers and inspectors. They are both logically moving on the transportation net, which is implemented as graph. The agent and passenger move independently but the behaviour of both, inspector and passenger, is well defined. For each passenger a unique commuting way is defined, which they are absolving repeatedly. For the movement of the inspectors several behaviours with different scale of regional focus are implemented.

2.1 NET

As the net properties might have an influence on the simulation, a real existing transportation net was used. Thus the simulation was done on the tram net of Zürich. The connection of the stations and the amount of lines are represented in a graph (Fig. 1). The stations that are not on a node were neglected.

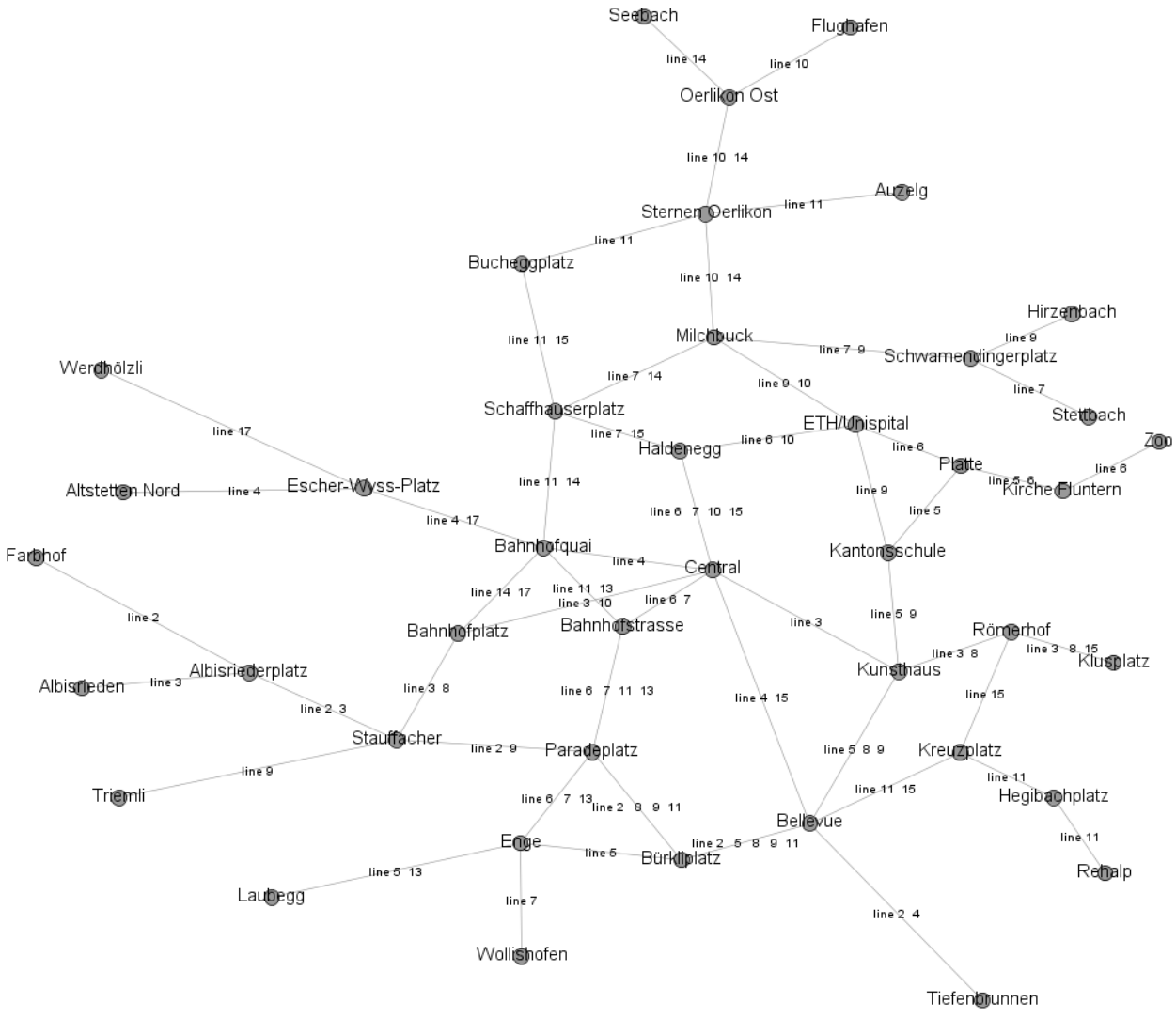

FIG.: 1: THE TRAMWAY OF ZÜRICH REPRESENTED AS GRAPH.

2.2 MOVEMENT OF THE PASSENGER

As most of the time public transportation is used for the way to work, the representation of the passenger movement is exclusively based on the commuting way. Therefor a unique closed way on the net was assigned to each passenger, which is then followed by the passengers repeatedly during the simulation. This closed way of going to work and back home is in this work from now on referred as one commuting way. Two different methods were applied to determine the commuting way.

2.2.1 RANDOM STEP

In the first method, we wanted to control the length of the commuting way. Therefore the commuting way was composed of a Gaussian distributed amount of random steps on the net. From a statistic of the government, the mean commuting time and the corresponding standard deviation was calculated and then implemented [1]. For the random steps on the net, the following rules were imposed:

- The starting point is randomly selected.
- The line is changed with a defined probability.
- If a node of the network with just one edge is reached the commuting is stopped.
- If the end of a line is reached, the commuting way is proceed with another line.
- After the passenger has reached their specific amount of stations, they take the same way back home.

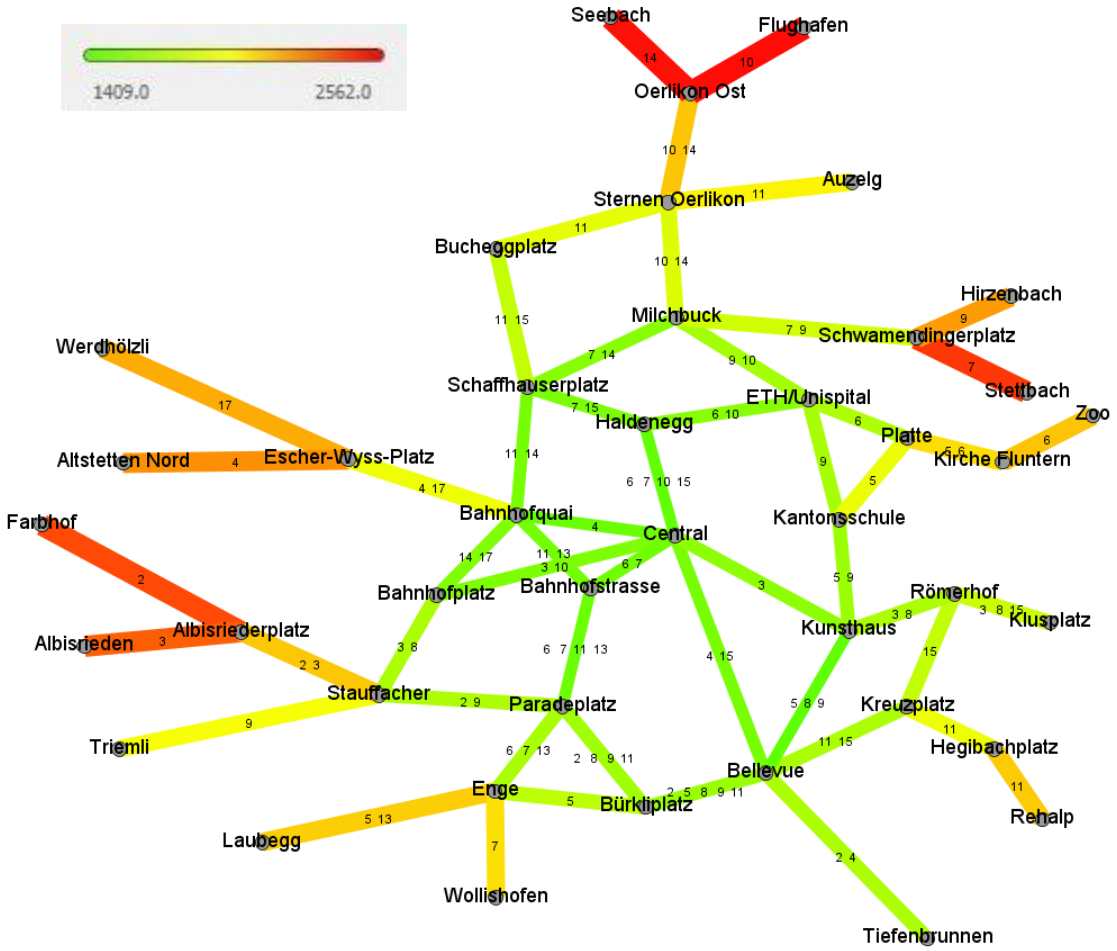

FIG. 2: THE NET SHOWS THE PASSENGER DISTRIBUTION OF THE RANDOM MOVEMENT. THE COUNTING WAS MADE IN A SIMULATION WITH 5088 PASSENGER AND 20 TIME STEPS.

2.2.2 SHORTEST PATH

The aim of the second method is to generate more realistic movements on the net. In the first method, passenger could go in circles and might end up at the same station as they started since the steps are chosen randomly. This does not represent the reality, as people tend to take the shortest way to work. In order to implement communing ways that are as short as possible, two randomly chosen nodes were connected by the shortest path, which was determined by the Bellman-Ford method [3]. To keep the way even more realistic, passengers only change the tram if they need to. If the same tram continues to the headed station, they stay in the tram. As we can see in Figure 3 this leads to a far more realistic distribution on the public transportation network. In the centre of the city the trams are more stuffed and at the end of a line not many people are still using it, this coincides far better with reality.

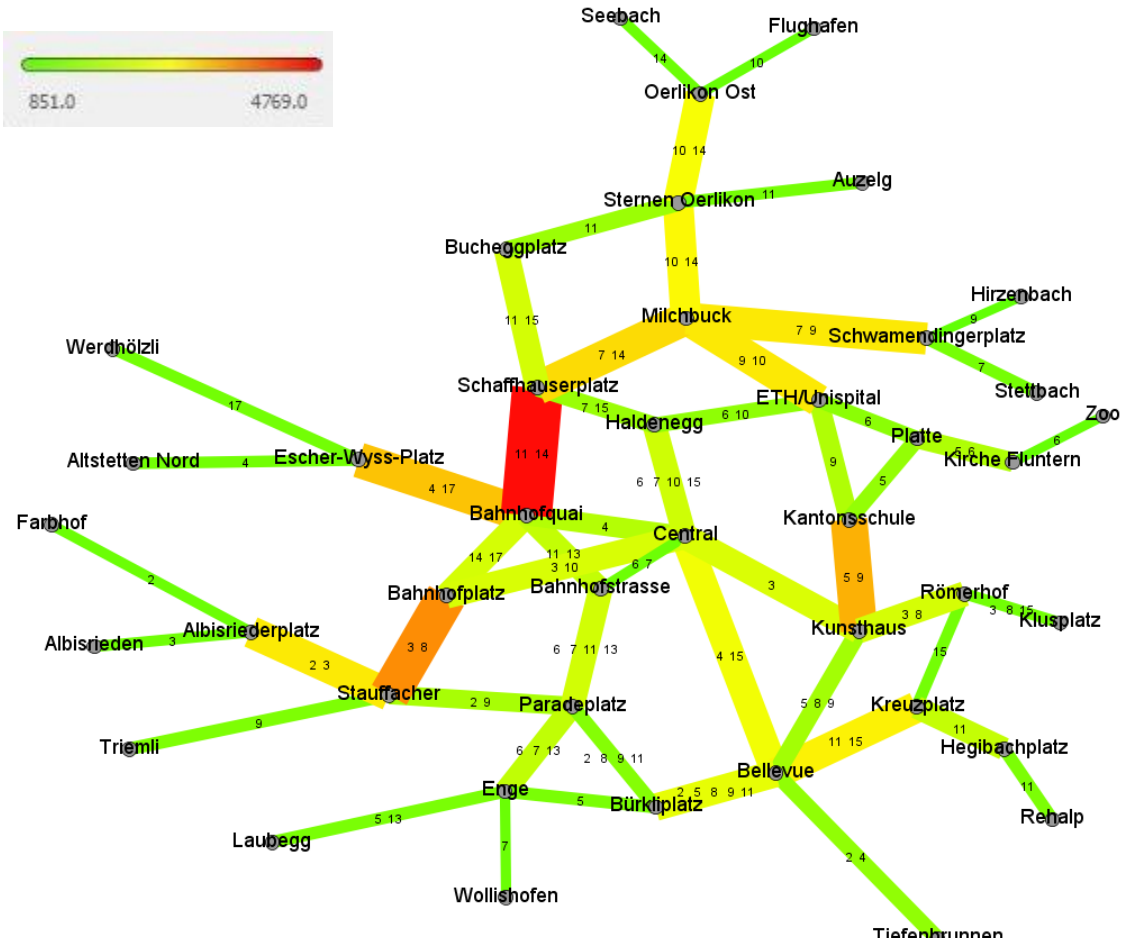

FIG. 2: THE NET SHOWS THE PASSENGER DISTRIBUTION OF THE SHORTEST PATH MOVEMENT. THE COUNTING WAS MADE IN A SIMULATION WITH 5088 PASSENGER AND 20 TIME STEPS.

2.3 MOVEMENT OF INSPECTORS

The movement of the inspectors should be implemented in a way that they can focus on a certain area in the net. Three different models with different degree of regional focusing were implemented. The **station** behaviour, which is most locally focused, the inspectors shuttle between two stations. In the **line** behaviour the inspectors are restricted to a certain line. And

the last and least locally focused behaviour consist of **random move** in the net with a certain probability of changing the line.

Since we did not define the spatial distribution of our graph, it might be problematic to talk about local focusing. Nevertheless this method enables the inspectors to focus on a subset of passengers.

2.4 INTERPLAY BETWEEN INSPECTORS AND PASSENGER

We let the simulation run for about 500 to 1000 iterations. In each iteration step the passengers and inspectors move one station further. So in one step a passenger can either start his commuting way or he continues his way based on the path described above. If he arrives at his destinations he takes the same way back and then starts again. Because the commuting ways do not have equal length one passenger may just start his way, whereas another one is already on the way back. This leads to a constant number of passengers who start their random walks. This is essential for the simulation, because only at the starting point it is decided whether the passengers take a ticket or not. The tickets are valid throughout the whole day (for one commuting way and back home). Once a passenger is inspected, he is not inspected in the same tram between two stations again. This constraint must be made because the inspectors behave randomly and if they unfortunately take the same tram each passenger would is controlled twice. If a passenger doesn't take a ticket, he doesn't have one for the entire day unless he is inspected (in reality he would have to buy a ticket besides paying a fee). So if a passenger is caught doing fare dodging, he gets a ticket for the rest of the commuting way and thus doesn't count as fare dodger until the end of the trip.

2.5 FARE EVASION BEHAVIOR

One of the most difficult part of our simulation was to implement the fare evasion behaviour. In reality the decision of the passenger, whether to buy a ticket or not is a highly complex process and varies strongly from individual to individual. This demands huge simplification. First it is assumed that all passengers act in the same way. Further it is assumed that passengers are motivated to do fare dodging by not being inspected while having a ticket. Furthermore we assumed that once the passengers started dodging fare they continue until they are caught. So in the implementation the passenger can be in two states, either in the legal state or in the fare dodging state. The probability of a legal to the dodging transition increases with the previous continuous series of uninspected trips, completed with appropriate fare. Or putting it the other way, the probability of the passenger taking a ticket exponentially decays (equation 1).

$$P_{legal \rightarrow dodging}(S) = 1 - \exp(-\lambda S) \quad (1)$$

For λ<<1 this equation can be linearized (2). Using the linearized form saves computational time, since exponential equations are demanding computational time.

$$P_{legal \rightarrow dodging}(S) = \lambda S \quad (2)$$

For the opposite transition, from dodging to legal, the passenger must be caught. In this case the S value is set to zero again.

2.6 INPUT PARAMETERS

Since we wanted to implement a simulation that represents the situation in Zürich, appropriate input parameters were researched. The amount of passenger was selected to be 53 per vehicle. This number is given by a survey of the SBB [4]. There are 48 edges in our net. That makes 96 vehicles running constantly, as there is two ways for each edge. Thus an ensemble of 5088 passenger is most realistic. The number of inspectors was selected so that the checking probability in the simulation was one per cent. This is consistent to the checking probability in public transportation of Zürich [1].

In order to guarantee the full transparency of the result, the input parameters of every simulation presented in this paper is given in the form: (number of iteration steps, number of passenger, number of inspectors, changing probability of inspectors, behaviour of passenger, behaviour of inspectors, lambda, starting step for averaging)

2.7 ANALYTICAL ESTIMATIONS AND STATISTIC VALUE

In order to review the output values of our simulation, analytic estimation were made and compared to the statistic measurements of the simulation.

Probability of being controlled by an inspector

$$p_a(check) = \frac{I}{V} \text{ (if V >> I)} \quad (3)$$

Controlling measure

$$p_s(check) = \frac{C}{TN} \quad (4)$$

Expectation of number of steps in the legal group

$$E = \sum_{x=1}^{M} [\prod_{h=1}^{x-1} 1 - P_{l \rightarrow d}(h)] P_{l \rightarrow d}(x)\, x \quad (5)$$

Variance of number of steps in the legal group

$$V = \sum_{x=1}^{M} [\prod_{h=1}^{x-1} 1 - P_{l \rightarrow d}(h)] P_{l \rightarrow d}(x)(\mu - x)^2 \quad (6)$$

Variable

I amount of inspectors

V	amount of vehicles
C	amount of controls
N	amount of time steps
$P_{l\to d}(h)$	Probability of a passenger starting fare dodging after a series of h legal tours without being inspected
M	Number of terms for approximation. For the linear probability the condition N < 1/lambda must be fulfilled.

3 RESULTS

3.1 COMMUTING WAY

The statistic about commuting way [5] analysed showed a fairly Gaussian distribution (Fig. 4). The mean time for the commuting way was determined to be 20.69 min and corresponding sigma 9.84 min. The travelling time was converted to an amount of station assuming an average traveling time between two stations of 1.5 min. This might seem too long, but it should be reminded, that not all stations are considered in our net and thus the station frequency must be lower in the simulation than in reality. These assumptions lead the conclusion that our time to work must be Gaussian distributed with a mean of 4.5 stations and a standard deviation of 2.1 stations.

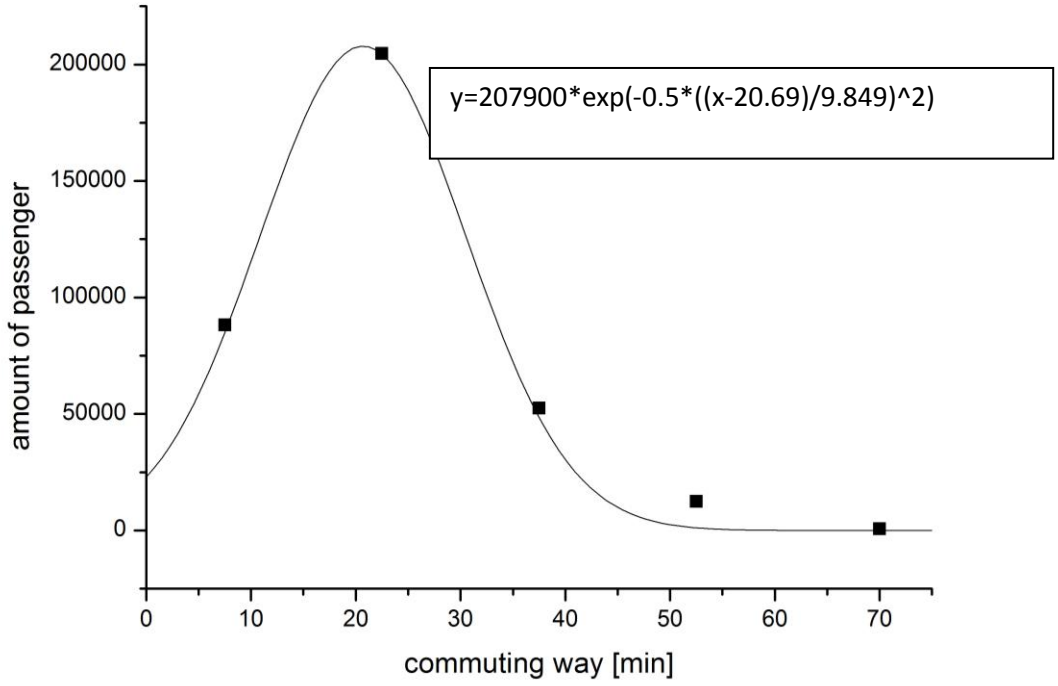

FIG. 4: THE TIME OF THE COMMUTING WAY WAS PLOTTED VS. THE AMOUNT OF PASSENGER ACCORDING TO A SURVEY OF THE GOVERNMENT.

3.2 DISTRIBUTION ON THE NET

The implementations of the commuting way lead two different distributions. The amount of passenger passing the ways were counted during a simulation with 5088 passenger and 20 steps and then plotted onto the net. In the random step behaviour the passenger density is higher at the outer edges (Fig. 2), whereas for the shortest path implementation the passenger density is higher at central edges (Fig. 3). In the random case, the passengers are more likely to end up in the outer region and then, they have to go the same way back. This cases higher density in the outer edges. In the second case the passenger move on shortest of two randomly chosen nodes. This forces them to pass through central located edges and thus leads to higher density in the central edges.

3.3 EQUILIBRIUM OF SYSTEM

In order to get good statistics the equilibrium of the system had to be examined. Since we started our simulation with all passengers being in the legal state, the simulation had to run for a certain amount of steps to reach equilibrium. The relaxation time was found to be dependent on the input parameter. First the relaxation of the different inspector behaviour was invested. One can see that the relaxation time is slightly longer for the random inspectors than for the other two (Fig. 1). Furthermore the simulation of the random behaviour is more unstable than the others. This can be ascribed to the fact, that there are two random dynamics implemented instead of just one: the random movement of the passenger and the inspectors. The behaviour "random", "line" and "station" represent inspector strategies with decreasing order in local focusing. The graph shows that the local focus leads to higher dodging rates. The parameter lambda, which is a degree for the fare dodging probability, also influences the relaxation time: As smaller the fare dodging probability as longer the relaxation time.

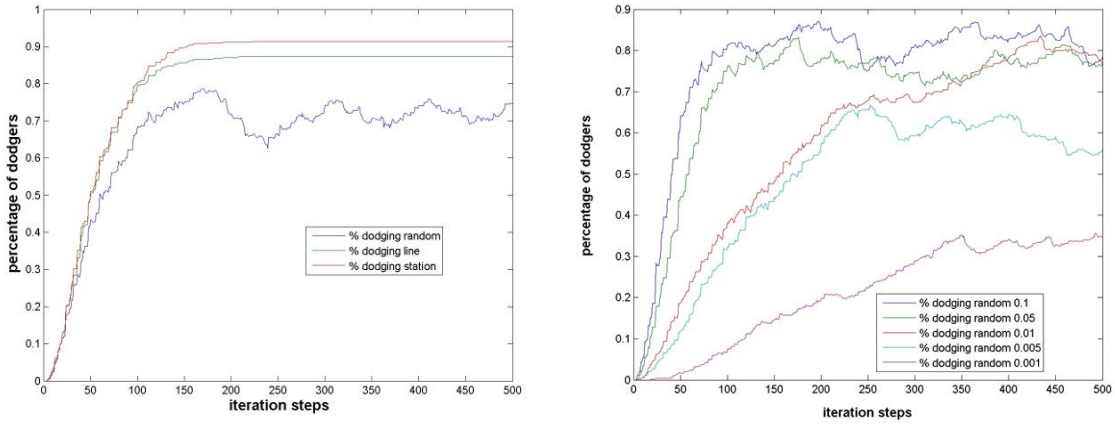

FIG. 5: THE EQUILIBRIUM WAS REACHED AFTER A CERTAIN AMOUNT OF RANDOM STEPS. THE CHOICE OF THE INPUT PARAMETER HAD AN INFLUENCE ON THE AMOUNT OF STEPS REQUIRED TO REACH EQUILIBRIUM. ON THE LEFT SIDE, THE PERCENTAGE OF FARE DODGERS IN THE SYSTEM IS PLOTTED VS. THE ITERATION STEPS FOR THE THREE INSPECTOR BEHAVIOURS. ON THE RIGHT HAND THE RANDOM BEHAVIOUR OF THE INSPECTOR IS IMPLEMENTED. THE PARAMETER Λ FOR THE FARE DODGING PROBABILITY IS VARIED FROM 0.001 TO 0.1. (500,500,5,0.5,SP,RANDOM,LAMNDA,0)

For a λ of 1 the relaxation time is about 100 steps whereas for a λ of 0.001 the relaxation time takes about 450 steps. Thus the sampling interval must be adjusted according to the steady state.

3.4 AMOUNT OF INSPECTORS

In order to control the inspection probability, its dependence on the amount of inspector in the system must be investigated. According to the analytic estimation (equation 1) the inspecting probability increases linearly with the amount of inspectors. This could be verified with the simulation (Fig. 6). The values from the simulation are about factor 2/3 smaller than the estimation. This can be ascribed to the fact that in the estimation inspections can be counted several times. If for instance a passenger wanders several stations in the same vehicle as an inspector, it is counted as several inspections within the estimation, but in the more

sophisticated simulation this is only counted as one inspection. Furthermore the implementation showed a higher inspecting probability for the passenger following a random commuting way than the one how on the shortest path commuting way. This difference can be explained by the passenger distribution on the net. It is different for the two models and thus the distribution that matches better with the distribution of the inspectors leads to a higher inspecting probability. Since the inspectors are also moving randomly on the net, their distribution obviously matches better with the commuting way and thus this model shows a higher inspecting probability. But the difference is small and therefore doesn't influence the fraction of dodgers.

In the net of Zürich about 1 percent of the passengers are inspected. That corresponds to an amount of about two inspectors in our model.

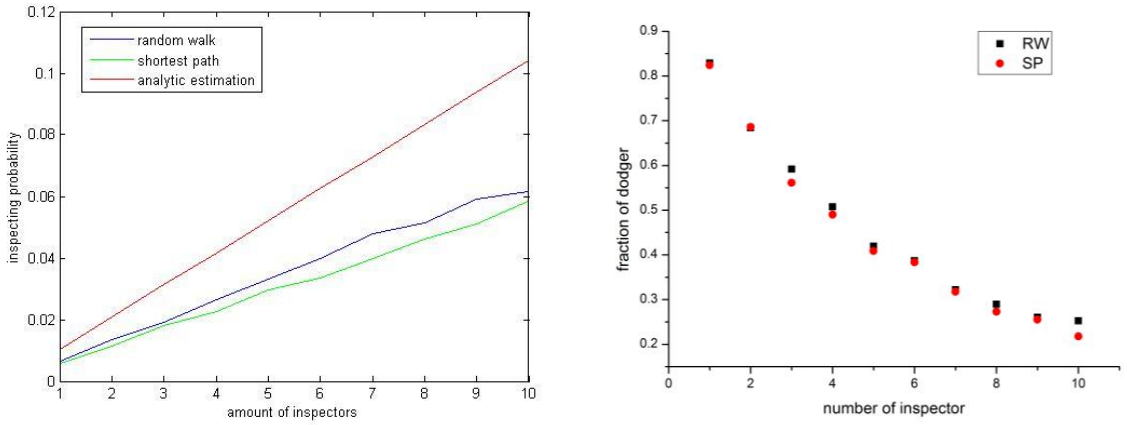

FIG. 6: THE INSPECTING PROBABILITY IS PLOTTED AGAINST THE AMOUNT OF INSPECTORS IN THE SYSTEM FOR THE RANDOM AND SHORTEST PATH COMMUTING WAY OF THE PASSENGER (LEFT). ON THE RIGHT SIDE THE FRACTION OF DODGER IS PLOTTED VS. THE NUMBER OF INSPECTORS. THE ANALYTIC ESTIMATION (EQUATION 3) IS PLOTTED FOR COMPARISON. (1000,500, I,5,SP,RANDOM,0.05,0)

3.5 CHANGING RATE OF THE INSPECTORS

In Fig. 5 it was already shown that local focussing of the inspectors leads to higher dodging rates. A more sophisticated method of regional focused inspecting was implemented by manipulating the probability of the inspector to change the line. If the inspectors seldom change the line they are focusing locally. Thus the local focusing can be implemented by choosing small changing rates. If this changing probability is very small, we end up with the same result as for the line behaviour (comparing Fig. 7 with Fig. 5). One can even see the single changing of the inspector: The graph with changing probability 0.01 shows a zig-zag band after about 800 steps. At each drop of the graph, an inspector changes the line and inspects passengers that have already changed to fare dodging.

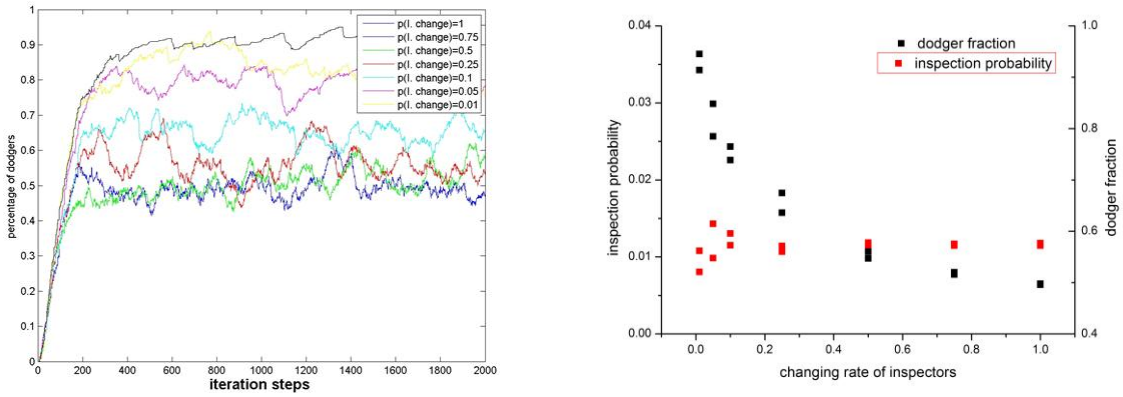

FIG. 7: THE PROBABILITY OF THE INSPECTOR TO CHANGE THE VEHICLES IS PLOTTED VS. THE DODGERS FRACTION IN THE SYSTEM. ON THE LEFT SIDE DODGERS FRACTION EVOLUTION FOR DIFFERENT INSPECTOR CHANGING PROBABILITIES ARE PLOTTED AS PROVE OF STABILITY. ON THE RIGHT SIDE THE AVERAGES OF THE DODGER FRACTIONS AND THE AVERAGE INSPECTION PROBABILITY ARE PLOTTED AGAINST THE CHANGING RATE OF THE INSPECTORS. (2000,500,2,CHANGEPROB,SP,RANDOM,0.01,600)

The averages in Fig. 7 show that the fare dodging increases with regional focusing of the inspector. Furthermore one can see that the inspection probability don't significantly change with the changing rate of the inspectors. This can be ascribed to the fact that at low changing rates the same passenger are inspected.

For a realistic representation, the changing probability must be set to 1, since the real inspectors most probably maximize their efficiency.

3.6 APPLYING TO THE NET IN ZÜRICH

As mentioned in chapter two in Zürich 1 percent of the passengers are inspected and about one per cent of the passengers are fare dodging. The first one of these two parameters was already achieved by setting the amount of inspectors to two (chapter 3.4). The remaining parameters were set according to the knowledge gained so far. The commuting way was set to the most realistic shortest path method (chapter 3.2). The inspector behaviour was set to random with highest changing rate as this was found to be most efficient (chapter 3.3 and 3.5). The only parameter left over is lambda. Thus lambda could be varied to find the favoured dodger fraction of 1%.

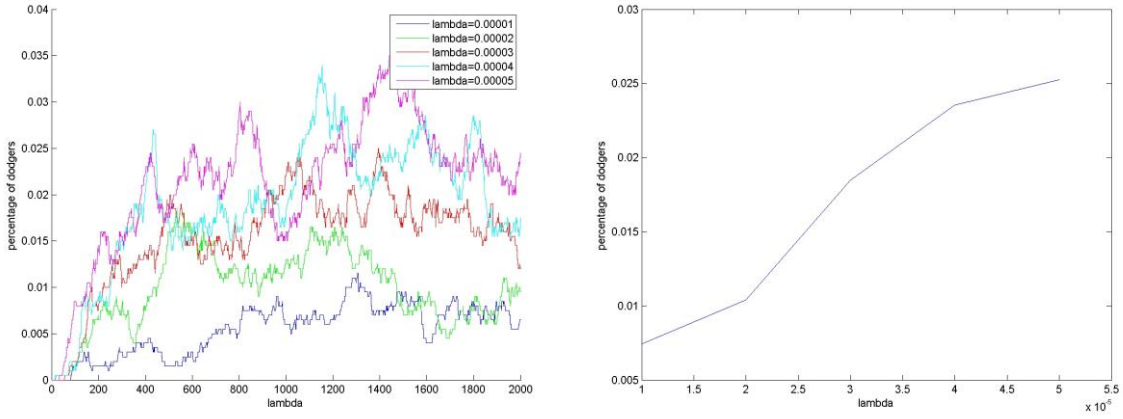

FIG. 8: THE INPUT PARAMETERS WERE SET SO THAT IT REPRESENTS THE SITUATION IN ZÜRICH. THE PARAMETER LAMBDA, WHICH IS THE PARAMETER FOR THE DODGING PROBABILITY WAS VARIED. SIMULATION WITH INPUT PARAMETER OF ZÜRICH. (2000,PASSENGER,2,1,SP,RANDOM,LAMBDA,1000)

As very small lambda were used, the simulation was quite unstable (Fig. 8). Thus 2000 steps were simulated and the averaging was started from 1000 steps on. At $\lambda = 2 * 10^{-5}$ an inspection probability of 1.1 % and a fare dodging fraction of 1.0 % was found. With this lambda the expectation value and its standard deviation for the average amount of tours done in the legal state if not being inspected (equation 5) was determined to be 278 +/-145 commuting ways. This corresponds to about 2224 +/- 1160 trip.

4 CONCLUSION

A new agent based model of fare evasion and inspection interplay was successfully implemented on the tram net of Zürich.

The movement of the passenger turned out to be most realistic for the shortest path implementation, since its distribution showed higher density in the central edges. The random walk implementation in contrast showed a higher density at the periphery of the net, which of course is rather unrealistic. Though the length could only be set in the random behaviour, the shortest path method was not unrealistic with an average commuting way of eight stations

In order to avoid fare evasion, it might be assumed, that the inspecting probability must be maximized and thus the inspection enhanced at regions with high passenger densities. The simulation revealed that this strategy leads to a small increase in the inspection probability. But unfortunately it showed no effect on the dodger fraction. Instead of the inspecting probability the presents of the inspectors that is noted by the passenger turned out to be most important. This present requires two different measures. First the inspection must cover the entire net, thus their distribution must spread over the net evenly. This measure derives from the lower dodging rate of the “random” inspector behaviour compared to the line and station behaviour. Secondly considering time evolution, the inspecting rate of each station must be high and also uniform distributed, otherwise the passenger in the current uninspected area start fare dodging. This claim is derived from the fact that for high changing rate the dodging fraction was significantly smaller than for low rates. This finding is verified by the experience of the inspectors of VBZ net. They state that the dodging rate increases instantly in uninspected areas [1]. This leads to the conclusion that in a proof-of-payment system the inspection is more efficient if being focused on an spatial and temporal even distribution on the net instead of a high inspection rate.

The model was also adapted to the situation in Zürich. The findings of a statistic of the net in Zürich -one per cent inspecting probability and one per cent dodging rate - could be reproduced in our model. Under these parameters the simulation determined an average of 278 legal commuting ways until the passengers start dodging. That means the passengers buy in average tickets for 278 day (corresponding to 2224 trip) and if they are not inspected during this time, they start dodging. On first sight this finding seems to be contradicting with the fixed input parameter. One expects the dodging rate to be smaller than one per cent if it takes the passenger about 2224 trips for starting dodging and they are inspected every hundredth trip. In order to understand this, one has to encounter the standard deviation of 1160 trips, which has the consequences that there are passenger that start dodging much after much less uninspected trips.

5 REFERENCES

[1] Jagd auf den Schwarzfahrer, NZZ Folie 10/08 Thomas Schenk

[2] Uncovering San Francisco Muni's Proof-of-Payment Patterns to Help Reduce Fare Evasion, Jason Le

[3] On a Routing Problem. Bellman, R. (1958). Quarterly of Applied Mathematics 16(1), 87-90.

[4] Umweltfahrplan SBB, Oktober 2012

[5] Eidgenössische Volkszälung 2000, Pendlermobilität in der Schweiz, Roman Frick

6 APPENDIX

6.1MATLAB CODE

Main.m

```
% project: ticket evasion
% editor: Lukas, Jens
% HS 2012

clear all;
%profile on;

amountp=500;
steps=500;
model={'random' 'line' 'station'};
amounti=3;
lambda=0.0002;


%%testing, simulating scenarios and visualize statistics
%{
stat2=[];

for i=linspace(0.001,0.0001,10)
    stat2 = [stat2;runsim(steps,amountp,amounti,1,model(1),i)];
end

figure;
i=linspace(0.001,0.0001,10);

plot(i,stat2(:,7)*100,i,stat2(:,8)*100,i,stat2(:,9)*100)

legend('% controlled','% dodging','% dodging and controlled', 'Location', 'Best');
xlabel('{\bf lambda}','fontsize',14);
name = ['figure\station500-500-lambda.jpg'];
print('-djpeg95', name);




stat2 = runsim(steps,amountp,amounti,0.05,model(1),0.1);
stat2 = [stat2;runsim(steps,amountp,amounti,0.05,model(1),0.05)];
stat2 = [stat2;runsim(steps,amountp,amounti,0.05,model(1),0.01)];
stat2 = [stat2;runsim(steps,amountp,amounti,0.05,model(1),0.005)];
stat2 = [stat2;runsim(steps,amountp,amounti,0.05,model(1),0.001)];


figure;
i=1:steps;

plot(i,stat2(1,:),i,stat2(2,:),i,stat2(3,:),i,stat2(4,:),i,stat2(5,:));
```

```
legend('% dodging random 0.1','% dodging random 0.05','% dodging random 0.01','% dodging
random 0.005','% dodging random 0.001', 'Location', 'Best');
name = ['figure\random0,1-0,005.jpg'];
print('-djpeg95', name);

%}


    stat2=[];

    for i=1:3
        stat2 = [stat2;runsim(steps,amountp,amounti,1,model(i),lambda)];
    end
    i=1:3;
    figure;
    plot(i,stat2(:,7).*100,'--c',i,stat2(:,8).*100,'--g',i,stat2(:,9).*100,'--b')
    legend('% controlled','% ppl doging','% ppl controlled w. doging', 'Location',
'WestOutside')
    name = ['figure\fixedevery.jpg'];
    print('-djpeg95', name);
```

```
function [stat] = runsim(steps,pass,insp,Ichange,model,lambda_dodg)

%simulation parameter

%--------------------------------------------------------------------------
itstep=steps;            % iteration steps
amount_passenger=pass;    % amount of Passenger
amount_inspector=insp;    % amount of ticket inspectors
p0_schwarz=1;        % initual evation probability and initual evation fraction
change_probe=0.4;

%Pendelweg
station_mean=4.6;       %mittlerer Weg Pendelhinfahrweg
station_sigma=2.1;              %stantartabweichung vom Pendehinfahrtweg

%--------------------------------------------------------------------------

%Initialisation

net=createNetz;
[Passenger]=inPass(amount_passenger,p0_schwarz,station_mean,station_sigma,change_probe,ne
t);

Way_P=zeros(amount_passenger,6);

A=zeros(42);
A=num2cell(A);

for p=1:amount_passenger
    % fill in first way of Passenger
```

```
    Way_P(p,1:3)=Passenger{p,4}(1,:);
    % pendelschritt
    Way_P(p,5)=1;
end

if(amount_inspector>0)
    %fill in first way of Inspectors
    Way_K=inWay_K(amount_inspector,net);
end

%Iteration
for i=1:itstep
  schwarz=0;
  erwischt=0;
  poskont=0;
  tours=0;

  %update Passenger

  for p=1:amount_passenger
    [Way_P(p,1:6), Passenger(p,1:4), schwarzp, toursp,A]=
upPass(Way_P(p,1:6),Passenger(p,1:4),1,A,lambda_dodg);
    schwarz=schwarz+schwarzp;
    tours=tours+toursp;
  end

  %update ticket inspector
  for p=1:amount_inspector
    [Way_P, Way_K, erwischtp, poskontp,A]= upKont(Way_P, Way_K, p,A);
    Way_K(p,:)=selectKwaynew(net,1,Way_K(p,:),model,Ichange);
    erwischt=erwischt+erwischtp;
    poskont=poskont+poskontp;
  end


  %update statistics
  Statistic(i,:)=[schwarz,erwischt,poskont,tours];

end

stat = [(Statistic(:,1)./tours)'];
figure;
plot(1:itstep,Statistic(:,1),1:itstep,Statistic(:,2),1:itstep,Statistic(:,3),1:itstep,Sta
tistic(:,4));
legend('schwarz','erwischt','poskont','tours','Location','best')
%title([model 'p' num2str(pass) 'i' num2str(insp)]);
%name = ['figure\' model 'p' num2str(pass) 'i' num2str(insp) '.jpg'];
%print('-djpeg95', name);

[counts, percentage]=getstats(Statistic);

stat=[counts,percentage];
%graph(A,model,insp,pass);
```

```
function Netz = createNetz()

    %dimension
    n=42;
    Netz=cell(n);
    Netz(1,7)={17};
    Netz(2,7)={4};
    Netz(3,6)={2};
    Netz(4,6)={3};
    Netz(5,8)={9};
    Netz(6,8)={[2,3]};
    Netz(7,14)={[4,17]};
        Netz(8,13)={[2,9]};
    Netz(8,15)={[3,8]};
    Netz(9,11)={[5,13]};
    Netz(10,11)={7};
    Netz(11,42)={5};
    Netz(11,13)={[6,7,13]};
    Netz(12,42)={[2,5,8,9,11]};
    Netz(12,28)={[2,4]};
    Netz(12,25)={[11,15]};
    Netz(12,30)={[5,8,9]};
    Netz(12,17)={[4,15]};
    Netz(13,42)={[2,8,9,11]};
    Netz(13,16)={[6,7,11,13]};
    Netz(14,15)={[14,17]};
    Netz(14,31)={[11,14]};
    Netz(14,16)={[11,13]};
    Netz(14,17)={4};
    Netz(15,17)={[3,10]};
    Netz(16,17)={[6,7]};
    Netz(17,18)={[6,7,10,15]};
    Netz(17,30)={3};
    Netz(18,19)={[6,10]};
    Netz(18,31)={[7,15]};
    Netz(19,20)={9};
    Netz(19,21)={6};
    Netz(19,32)={[9,10]};
    Netz(20,21)={5};
    Netz(20,30)={[5,9]};
    Netz(21,22)={[5,6]};
    Netz(22,23)={6};
        Netz(24,25)={15};
    Netz(24,26)={[3,8,15]};
    Netz(24,30)={[3,8]};
    Netz(25,27)={11};
    Netz(27,29)={11};
    Netz(31,32)={[7,14]};
    Netz(31,33)={[11,15]};
    Netz(32,34)={[7,9]};
```

```
    Netz(32,37)={[10,14]};
    Netz(33,37)={11};
    Netz(34,35)={7};
    Netz(34,36)={9};
    Netz(37,38)={[10,14]};
    Netz(37,39)={11};
    Netz(38,40)={14};
    Netz(38,41)={10};

    %make the matrix symmetric
    for i=1:n
       for j=1:n
           [sn, sm] = size(Netz{i,j});
           if sn==0&&sm==0
               Netz(i,j)={0};
           end
           Netz(j,i)=Netz(i,j);
       end
    end
end
```

```
function line = findline(net,s,d,l)

lines = net{s,d};
li = length(lines);
line = 0;

%if line continues, take this one
for i=1:li
   if(lines(i)==l)
       line = l;
   end
end

%need to change the line (take random)
if(line==0)
    line=lines(randi(li,1));
end
```

```
function [Way_P, Passenger, schwarz, tours,A] = upPass(Way, Passenger, PassID,A,lambda)

    %PassID: unique passenger ID, index in matrix
    %Way[n x 6]: matrix with all passenger ways;
    %           start, dest, line, doging, pendelschritt, already
    %           controlled (erwischt) =1 odr =2 for controlled with a
    %           ticket
    %Passenger[n x 4]: cell-matrix with properties of passengers;
```

```
    %                   initial probaility of taking a ticket, # of tours without control,
dogging (yes=1,no=0), Passenger{PassID,4}
    %%%%%%%%%%%%%Passenger{PassID,4}[ x 3]: matrix with Passenger{PassID,4}
    %Statistic

    tours=0;
```

```
    schwarz=0; %number of 'fare-dogers'

    [maxschr]=size(Passenger{PassID,4},1);

    %update pendelschritt
    if Way(PassID,5)>=maxschr
        Way(PassID,5)=1;
    else
        Way(PassID,5)=Way(PassID,5)+1;
    end
```

```
%calculate fare-doging probability
```

```
    %first station
    if Way(PassID,5)==1
        %passenger just got controlled and was caught
        if Way(PassID,6)==1
            Passenger{PassID,2}=0;
            Passenger{PassID,3}=0;
            Way(PassID,4)=0;
            Way(PassID,6)=0;
        %passenger just got controlled and had a ticket
        elseif (Way(PassID,6)==2)
            Passenger{PassID,2}=0;
            Passenger{PassID,3}=0;
            Way(PassID,4)=0;
            Way(PassID,6)=0;
        else
            Way(PassID,6)=0;
            %passenger is doging
            if Passenger{PassID,3}==1
                % because he was doging last time he continues
                Way(PassID,4)=1;
                Way(PassID,6)=0;
            else
                %probability that passenger takes a ticket
                %Passenger{PassID,2}=(1-(1-Passenger{PassID,2})*(1-Passenger{PassID,1}));
%geometic probability with p=starting probability (set at initialisation)
                %actprob =  1-Passenger{PassID,1}-Passenger{PassID,2}*((1-
Passenger{PassID,1})/10);
                %actprob2 = 1-Passenger{PassID,1}-
Passenger{PassID,2}*Passenger{PassID,2}*((1-Passenger{PassID,1})/10)
                actprob3 = exp(-lambda*Passenger{PassID,2});
                ran = rand(1,1);
                if ran>actprob3
                    Way(PassID,4)=1;
                    Passenger{PassID,3}=1;
                    %ran
```

```
                    %actprob3
                else
                    %disp('ran<');
                end
            end
            Passenger{PassID,2}=Passenger{PassID,2}+1;
        end
    else
        if Way(PassID,6)==1
            Passenger{PassID,2}=0;
            Passenger{PassID,3}=0;
            Way(PassID,4)=0;
            Way(PassID,6)=0;
        elseif Way(PassID,6)==2
            Way(PassID,4)=0;
            Way(PassID,6)=0;
            Passenger{PassID,2}=0;
            Passenger{PassID,3}=0;
        elseif Way(PassID,6)==0

        end
    end

    %if passenger is doging between actual stations update statistics
    if Way(PassID,4)==1
        schwarz=schwarz+1;
        A=statup(Way(PassID,1:3),0,A);
    end

    %update statistics
    tours=tours+1;
    A=statup(Way(PassID,1:3),1,A);

%update Way matrix

    Way(PassID,1)=Passenger{PassID,4}(Way(PassID,5),1);
    Way(PassID,2)=Passenger{PassID,4}(Way(PassID,5),2);
    Way(PassID,3)=Passenger{PassID,4}(Way(PassID,5),3);

    Way_P=Way;

end
```

```
function [Way_P, Way_K, erwischt, poskont,A] = upKont(Way_P, Way_K, KontID,A)

    %KontID: unique passenger ID, index in matrix
    %Way_P[n x 6]: matrix with all passenger ways
    %Way_K[m x 4]: matrix with all controller ways
    %model: controlling model
    %A: matrix with statistics about controls, fare-doging...
```

```
    orig=Way_K(KontID,1);
    dest=Way_K(KontID,2);
    line=Way_K(KontID,3);
    erwischt=0;
    poskont=0;

    %control the passengers
    [n,m]=size(Way_P);
    for i=1:n
        %see if control and passenger is on same wagon(tram)
       if Way_P(i,3)==line&&Way_P(i,1)==orig&&Way_P(i,2)==dest
           if ((Way_P(i,4)==1)&&Way_P(i,6)==0)
               erwischt=erwischt+1;
               A=statup(Way_P(i,1:3),2,A);
               Way_P(i,6)=1;
           elseif ((Way_P(i,6)~=1)&&Way_P(i,6)~=2)
                poskont=poskont+1;
                A=statup(Way_P(i,1:3),3,A);
                Way_P(i,6)=2;
           end
       end
    end
end
```

```
function A = statup(Way,arg,A)
    %arg:
    %0:doging
    %1:ride
    %2:erwischt
    %3:poskont
    %
    if(arg==0)
        A{Way(1,1),Way(1,2)}=A{Way(1,1),Way(1,2)}+[1,0,0,0];
    end
    if(arg==1)
        A{Way(1,1),Way(1,2)}=A{Way(1,1),Way(1,2)}+[0,1,0,0];
    end
    if(arg==2)
        A{Way(1,1),Way(1,2)}=A{Way(1,1),Way(1,2)}+[0,0,1,0];
    end
    if(arg==3)
        A{Way(1,1),Way(1,2)}=A{Way(1,1),Way(1,2)}+[0,0,0,1];
    end
end
```

```
function [counts,percentage] = getstats(Statistic)

% calculate percentages
%

controldoging = sum(Statistic(:,2));
controlnondoging = sum(Statistic(:,3));
totalcontrols = controlnondoging+controldoging;
totaldoging = sum(Statistic(:,1));
totaltours = sum(Statistic(:,4));

format shortg

percentage=[totalcontrols/totaltours,totaldoging/totaltours,controldoging/totalcontrols];
counts=[controldoging,controlnondoging,totalcontrols, totaldoging, totaltours-
totaldoging, totaltours];
```

```
function x=selectrandLine(net, Position)

%selects a a random aim and line from a given position
counter=0;
for i=1:size(net,1)
    if (net{i,Position} ~= 0)
        counter=counter+1;
        connection(counter)=i;
    end
end
    %select aim
    aim=connection(randi([1,size(connection,2)]));
    %select origin
    line=net{aim, Position}(1,randi(size(net{aim,Position})));

    x=[Position,aim,line];
```

```
function newWay =selectLine(net,lastWay,stay_change)

%stay_change: 'stay' passenger stay on line,
%             'change' passenger change line
%'stay':selects a a random new aim and line from a given position
%'stay' and 'change': if passenger is at death end(node with only one edge): x --> 0


%%check if passenger reaches death end
counter=0;

for k=1:size(net,1) %counts available directions
```

```
    %lastWay(2)
    if (net{k,lastWay(2)}~=0) %select available connections
        if(k~=lastWay(1)); %avoid last direction
        counter=counter+1;
        connection(counter)=k;
        end
    end
end

if (counter==0) %if death end (node with only one edge) is reached
    newWay=[lastWay(2),lastWay(1),lastWay(3)];
    return;
end
```

```
%Passanger stay
```

```
if (strcmp(stay_change,'stay')==1)
     %see where line goes next --cell n and cellelement m
    for n=1:size(net,1)
        for m=1:size(net{lastWay(2),n},2)
             %Passanger can continue on the line
            if  (net{lastWay(2),n}(1,m)==lastWay(3)) && (n~=lastWay(1))
              newWay(1)=lastWay(2);
              newWay(2)= n;
              newWay(3)=lastWay(3);
            else
                  stay_change='change';
            end

        end
    end

end
```

```
if (strcmp(stay_change,'change')==1)
    %select aim
    aim=connection(randi([1,size(connection,2)]));
    %select line
    line=net{aim,lastWay(2)}(1,randi(size(net{aim,lastWay(2)})));
    newWay=[lastWay(2),aim,line];
end
```

```
function [newWay]=selectKway(net,Inspector_ID,lastWay,model,modelproperty1)

%upgrates Inspector Way of the Inspector with respectiv ID
%newWay [3 x amount inspector]: new way of inspectors
%Inspector_ID:
%lastWay: [3 x amount inspector]
%model: 'random' --> movement with change line probablitiy p=
%modelproperty1
```

```
%'line' --> inspectors do not change line
%'station' --> inspectors shuttles between two stations

%%random
    if (strcmp(model,'random')==1)
        if (rand()<modelproperty1)
             newWay= selectLine(net,lastWay(Inspector_ID,:),'change');
        else
            %see where line goes next --cell n and cellelement m
              for n=1:size(net,1)
                    for m=1:size(net{lastWay(Inspector_ID,2),n},2)
                        %Passanger can continue on the line
                          if
(net{lastWay(Inspector_ID,2),n}(1,m)==lastWay(Inspector_ID,3)) &&
(n~=lastWay(Inspector_ID,1))
                              newWay(Inspector_ID,1)=lastWay(Inspector_ID,2);
                              newWay(Inspector_ID,2)= n;
                              newWay(Inspector_ID,3)=lastWay(Inspector_ID,3);
                          else
                              newWay(Inspector_ID,1)=lastWay(Inspector_ID,2);
                              newWay(Inspector_ID,2)=lastWay(Inspector_ID,1);
                              newWay(Inspector_ID,3)=lastWay(Inspector_ID,3); %end of
line
                          end
                    end
             end
        end
    end

%%line
    if (strcmp(model,'line')==1)
             %see where line goes next --cell n and cellelement m
            for n=1:size(net,1)
                    for m=1:size(net{lastWay(Inspector_ID,2),n},2)
                        %Passanger can continue on the line
                          if
(net{lastWay(Inspector_ID,2),n}(1,m)==lastWay(Inspector_ID,3)) &&
(n~=lastWay(Inspector_ID,1))
                              newWay(Inspector_ID,1)=lastWay(Inspector_ID,2);
                              newWay(Inspector_ID,2)= n;
                              newWay(Inspector_ID,3)=lastWay(Inspector_ID,3);
                          else
                              newWay(Inspector_ID,1)=lastWay(Inspector_ID,2);
                              newWay(Inspector_ID,2)=lastWay(Inspector_ID,1);
                              newWay(Inspector_ID,3)=lastWay(Inspector_ID,3); %end of
line
                          end
                    end
             end

    end
    %%station
    if (strcmp(model,'station')==1)
```

```
        newWay(Inspector_ID,1)=lastWay(Inspector_ID,2);
        newWay(Inspector_ID,2)=lastWay(Inspector_ID,1);
        newWay(Inspector_ID,3)=lastWay(Inspector_ID,3);
    end
end
```

```
function [Way_K]=inWay_K(amount_inspector,net)

    Position=randi([1,41],amount_inspector,1);

    for p=1:amount_inspector
       Way_K(p,1:3)= selectrandLine(net,Position(p));
    end
end
```

```
function [Passenger]=
inPass(amount_passanger,p0_schwarz,station_mean,station_sigma,change_prob,net)

    %p0_schwarz: initual probability for fare evastion
    %station mean: avrarage station traveld for the Pendelweg
    %sigma: sigma of station traveld for the Pendelweg
    %change_prop: passagiere können auch wieder auf selbe linie umsteigen !!
    % initalisierung Matrix Passanger:colums
    %(initial probaility of taking a ticket, actual probaility of taking a ticket (inital
0), dogging (yes=1, no=0), pendelweg)

    Passenger=zeros(amount_passanger,4);
    Passenger=num2cell(Passenger);
    M=zeros(42,42);
    for m=1:42
       for n=1:42
           if (net{m,n}~=0)
                M(m,n)=1;
           end
       end
    end
    M=sparse(M);

    for p=1:amount_passanger
        Passenger{p,1}= p0_schwarz;
    %fill in P_schwarz to matrix Passanger
    end

    for p=1:amount_passanger
         clear way;

         source=randi(42,1);
```

```
          dest=source;
          while(dest==source)
              dest=randi(42,1);
          end

          %compute shortest path
          [dist,path,pred]=graphshortestpath(M,source,dest,'DIRECTED',false);
          way=zeros(1,3);
          line=0;
          for i=1:dist
              way(i,1)=path(i);
              way(i,2)=path(i+1);
              way(i,3)=findline(net,path(i),path(i+1),line);
              line=way(i,3);
          end

         %compute way home
         tmpway=way(:,1);
         tway=way;
         tway(:,1)=tway(:,2);
         tway(:,2)=tmpway;

         way=[way;flipud(tway)];

         Passenger(p,4)={way};
%write way to passanger matrix


    end
end
```

```
function [stat,den_P,den_K] = runsim_distribution(steps,pass,insp,p,model)

%postcondition
%den [station x station]:atributes of nodes in each cell

%simulation parameter

%-------------------------------------------------------------------------
itstep=steps;            % iteration steps
amount_passenger=pass;    % amount of Passenger
amount_inspector=insp;    % amount of ticket inspectors
p0_schwarz=p;        % initual evation probability and initual evation fraction
change_probe=0.4;

%Pendelweg
station_mean=4.6;       %mittlerer Weg Pendelhinfahrweg
station_sigma=2.1;              %stantartabweichung vom Pendehinfahrtweg

%-------------------------------------------------------------------------
```

```
%Initialisation

net=createNetz;
den_P=zeros(size(net,1));
den_K=zeros(size(net,1));
[Passenger]=inPass(amount_passenger,p0_schwarz,station_mean,station_sigma,change_probe,ne
t);

Way_P=zeros(amount_passenger,6);

A=zeros(42);
A=num2cell(A);

for p=1:amount_passenger
    Way_P(p,1:3)=Passenger{p,4}(1,:);    % fill in first way of Passenger
    Way_P(p,5)=1;                        % pendelschritt
end

if(amount_inspector>0)
    Way_K=inWay_K(amount_inspector,net);  %fill in first way of Inspectors
end

%Iteration
for i=1:itstep
  schwarz=0;
  erwischt=0;
  poskont=0;
  tours=0;

  %update Passenger

  for p=1:amount_passenger
    [Way_P(p,1:6), Passenger(p,1:4), schwarzp, toursp,A]=
upPass(Way_P(p,1:6),Passenger(p,1:4),1,A);
    schwarz=schwarz+schwarzp;
    tours=tours+toursp;
  end

  %update ticket inspector
  for p=1:amount_inspector
    [Way_P, Way_K, erwischtp, poskontp,A]= upKont(Way_P, Way_K, p,A);
    Way_K(p,:)=selectKway(net,1,Way_K(p,:),model,randi(10,1)/10);
    erwischt=erwischt+erwischtp;
    poskont=poskont+poskontp;
  end

  %update statistics
  Statistic(i,:)=[schwarz,erwischt,poskont,tours];
  for m=1:size(Way_P,1)
        den_P(Way_P(m,1),Way_P(m,2))=den_P(Way_P(m,1),Way_P(m,2))+1;
  end
  for m=1:size(Way_K,1)
```

```
        den_K(Way_K(m,1),Way_K(m,2))=den_K(Way_K(m,1),Way_K(m,2))+1;
  end

  %attNodes_P= attNodes_P+hist(Way_P(:,1),size(net,1));
  %attNodes_K= attNodes_K+hist(Way_K(:,1),size(net,1));
end

figure;
plot(1:itstep,Statistic(:,1),1:itstep,Statistic(:,2),1:itstep,Statistic(:,3),1:itstep,Sta
tistic(:,4));
legend('schwarz','erwischt','poskont','tours','Location','best')
title([model 'p' num2str(pass) 'i' num2str(insp)]);
name = ['figure\' model 'p' num2str(pass) 'i' num2str(insp) '.jpg'];
print('-djpeg95', name);

[counts, percentage]=getstats(Statistic);
%[counts, percentage]=showstats(Statistic);
 den_P=den_P'+den_P;
 den_K=den_K'+den_K;

stat=[counts,percentage];
graph_distribution(den_K,model,insp,pass,'denK');
graph_distribution(den_P,model,insp,pass,'denP');
```

```
function graph(A,model,insp,pass)

net=createNetz;
%create header
r='graph\r\n[\r\n';

stations=cell({'Werdhölzli','Altstetten
Nord','Farbhof','Albisrieden','Triemli','Albisriederplatz','Escher-Wyss-
Platz','Stauffacher','Laubegg','Wollishofen','Enge','Bellevue','Paradeplatz','Bahnhofquai
','Bahnhofplatz','Bahnhofstrasse','Central','Haldenegg','ETH/Unispital','Kantonsschule','
Platte','Kirche
Fluntern','Zoo','Römerhof','Kreuzplatz','Klusplatz','Hegibachplatz','Tiefenbrunnen','Reha
lp','Kunsthaus','Schaffhauserplatz','Milchbuck','Bucheggplatz','Schwamendingerplatz','Ste
ttbach','Hirzenbach','Sternen Oerlikon','Oerlikon
Ost','Auzelg','Seebach','Flughafen','Bürkliplatz'});

%creats nodes
for l=1:size(net,1)
    p=num2str(l);
    r=strcat(r,'node\r\n');
    r=strcat(r,'[\r\n');
    r=strcat(r,{'id '}, p,'\r\n');
    r=strcat(r,{'label "'},stations(l),'"\r\n');
    r=strcat(r,']\r\n');
end
```

```
%create csv file for edges

r='Source,Target,Type,Id,Label,Weight\r\n';
id=0;
%creats edges with statistics
for m=1:size(A,1)
   for n=m:size(A,1)
       if (size(A{m,n})==[1 4])
            r=strcat(r,num2str(m),'.0,');
            r=strcat(r,num2str(n),'.0,');
            r=strcat(r,'Undirected,');
            r=strcat(r,num2str(id),',');
                      r=strcat(r,num2str(A{m,n}),',');
                      r=strcat(r,'1.0\r\n');
                      id=id+1;
       end
   end
end

% write string to file
name=['csv\' model 'p' num2str(pass) 'i' num2str(insp) '.csv'];
fid = fopen(name,'w');
fprintf(fid,r);
fclose(fid);
```

```
function graph_distribution(A,model,insp,pass,name)

%precondition
%A= graph matrix A with amount of trips

net=createNetz;

%create csv file for edges
r='Source,Target,Type,Id,Label,Weight\r\n';
id=1;
%creats edges with statistics
for m=1:size(A,1)
   for n=m:size(A,1)
       if (A(m,n)~=0)
            r=strcat(r,num2str(m),'.0,');
            r=strcat(r,num2str(n),'.0,');
            r=strcat(r,'Undirected,');
            r=strcat(r,num2str(id),',');
                      r=strcat(r,num2str(net{m,n}),',');
            r=strcat(r,num2str(A(m,n)),',');
                      r=strcat(r,num2str(n),'\r\n');
```

```
                        id=id+1;
        end
    end
end


% write string to file
name=['csv\' model 'p' num2str(pass) 'i' num2str(insp) '_' name '.csv'];
fid = fopen(name,'w');
fprintf(fid,r);
fclose(fid);
```