The Booth Tolls for Thee 程序实现

周吕文,大连大学数学建模工作室

November 26, 2011

在阅读本材料时，请先确定你已经了解排队论及元胞自动机理论,并已经精读完05年特等奖论文：The Booth Tolls for Thee

原文分别建立了三个模型来计算公路收费亭的最优数量,其中第一和第三个模型都是用计算机仿真实现的.本文就来分析这两个模型是怎么样用程序实现的.第一个模型是一个微观的离散的排队模型,不考虑瓶颈的作用. 第二个模型是元胞自动机模型. 在介绍这两个模型前,先介绍一下先前工作-交通流数据的傅里叶展开.

1 交通流的傅里叶展开

原文作者从之前的一篇文献[Boronico,1998]中找到了某个收费站一天的各个整点间的交通流量(cars/min)的平均值. 为得到每个时刻的交通流量,我们需要利用数据进行插值或拟合.原文作者近似认为交通流量是时间的周期函数,且周期是24小时,抛开周日和周六,这确实是个不错的相法.因此作者用了8阶傅里叶级数来近似交通流量函数$F(t)$

$$F(t) = a_0 + \sum_{i=1}^{8} a_i \cos(i\omega t) + b_i \sin(i\omega t)$$

这里的$\omega = 2\pi/24 = 0.2618$,因此我们就得想办法得到系数$a_i(i = 0, 1, \cdots, 8)$和$b_i(i = 1, \cdots, 8)$. 显然通过拟合或插值不容拟得到以上系数,我们却可以通过回归来得到这些系数.我们首先计算$\sin(i\omega t)$,然后利用matlab中的regress函数进行回归,具体处理如下:

```
T=[0.5:1:23.5]';
influx=[15.44 15.32 15.16 19.9  47.09 89.95 ...  %  0.5- 5.5 hour
        105.9 85.52 54.68 43.11 40.16 40.85 ...  %  6.5-11.5 hour
        41.72 44.54 48.88 53.2  51.61 48.38 ...  % 12.5-17.5 hour
        39.72 30.51 29.48 26.82 21.21 17.22]';   % 18.5-23.5 hour

omega =2*pi/24; %all Fourier Coefficients
fc=zeros(24,8);
fs=zeros(24,8);
for i = 1:8
   fc(:,i)=cos(i*omega*T);
   fs(:,i)=sin(i*omega*T);
end
```

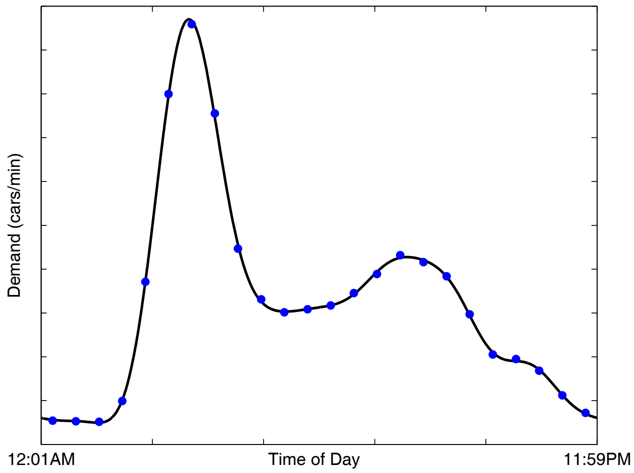
```
[B,BINT,R,RINT,STATS] = regress(influx,[ones(24,1),fc,fs],0.05);

t=0:0.01:24;
a0 = B(1);
a = B(2:9);
b = B(10:end);
```

这样我们就得到了交通流函数的系数a_0,a和b向量(原文作者由于在ω的值上略有失误,所以和这里我们得到的系数略有差异,但差异很小,影响不大,后面我们仍然尊重原文结果), 近而得到交通流函数$F(t)$:

$$\begin{aligned}F(t) = 42.77 &- 16.27\cos(1t\omega) - 19.59\cos(2t\omega) + 06.09\cos(3t\omega) + 07.79\cos(4t\omega)\\ &- 02.83\cos(5t\omega) - 03.06\cos(6t\omega) + 00.41\cos(7t\omega) + 00.77\cos(8t\omega)\\ &+ 11.01\sin(1t\omega) - 02.65\sin(2t\omega) - 12.38\sin(3t\omega) + 01.80\sin(4t\omega)\\ &+ 05.45\sin(5t\omega) - 00.59\sin(6t\omega) - 00.73\sin(7t\omega) + 00.08\sin(8t\omega)\end{aligned}$$

我们将傅里叶级数近似函数$F(t)$与实际点作对比,分别如图1的左图和右表.



傅里级数对交通流量的近似

开始时间	结束时间	时间	平均流量 (cars/min)	傅里叶近似 (cars/min)
12:00AM	1:00AM	0.5	15.44	15.16272478
1:00AM	2:00AM	1.5	15.32	15.42467822
2:00AM	3:00AM	2.5	15.16	15.18796896
3:00AM	4:00AM	3.5	19.9	19.81853474
4:00AM	5:00AM	4.5	47.09	47.22251986
5:00AM	6:00AM	5.5	89.95	89.61825869
6:00AM	7:00AM	6.5	105.9	106.4828683
7:00AM	8:00AM	7.5	85.52	84.72959878
8:00AM	9:00AM	8.5	54.68	55.57942216
9:00AM	10:00AM	9.5	43.11	42.42662327
10:00AM	11:00AM	10.5	40.16	40.49538486
11:00AM	12:00PM	11.5	40.85	40.83544106
12:00PM	1:00PM	12.5	41.72	41.63346483
1:00PM	2:00PM	13.5	44.54	44.44085865
2:00PM	3:00PM	14.5	48.88	49.29448007
3:00PM	4:00PM	15.5	53.20	52.55619485
4:00PM	5:00PM	16.5	51.61	52.21058951
5:00PM	6:00PM	17.5	48.38	48.16410937
6:00PM	7:00PM	18.5	39.72	39.50374966
7:00PM	8:00PM	19.5	30.51	31.11397219
8:00PM	9:00PM	20.5	29.48	28.86864636
9:00PM	10:00PM	21.5	26.82	27.19686700
10:00PM	11:00PM	22.5	21.21	21.26085220
11:00PM	12:00AM	23.5	17.22	16.91795178

图 1: 某个收费站一天的各个整点间的交通流量(cars/min)的平均值及其傅里叶级数展开的近似

2　排队模型

日常生活中存在大量有形和无形的排队或拥挤现象，如旅客购票排队，市内电话占线等现象。排队论的基本思想是1910年丹麦电话工程师A.K.埃尔朗在解决自动电话设计问题时开始形成的，当时称为话务理论。后经许多数学家发展,开成今天的排队理论。对于简单的系统,我们可以直接用排队理论

的相关公式来研究等待时间、排队长度、忙期长短等数量指标. 当排队系统的到达间隔时间和服务时间的概率分布很复杂时，或不能用公式给出时，那么就不能用解析法求解,这时就需用计算机模拟来求解.

排队模型的计算机模拟一般先要确定的随机变量概率分布,如顾客到达间隔服从指数分布$\exp(\lambda)$；产品需求量服从正态分布$N(\mu,\sigma^2)$ ；订票后但未能按时前往机场登机的人数服从二项分布$B(n,p)$ 。当然对于本文是不需要的,因为我们已经知道交通流量对时间的函数$F(t)$,可以直接根据$F(t)$产生车的数量.

排队模型一般分为两种,一种是并联模型,另一种是串联模型.本文的收费站属于第一种模型,可认为该系统是一个队列,多个服务台(收费站),其示意图如2图

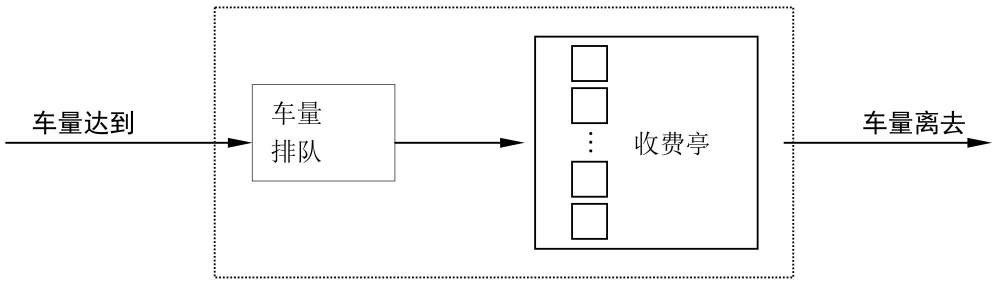


图 2: 收费站的排队模型示意图

首先我们根据上面得到的车流量函数,可以积分得到一天中每条公路到达的车辆为61565,并定义相关的数组:

```
i =ceil(61565 * lanes); %number of cars... 61565 found through integration of the influx equation
S(1:i) = 0; %service duration time for car i
A(1:i) = 0; %arrival time of car i
L(1:i) = 0; %leaving time of car i
B(1:i) = 0; %tollbooth used by car i
Start(1:i) = 0; %service start time for car i
```

仿真中,将时间以5秒为一个单位进行离散化,并定义相关数组

```
t = 86400/5+1; %number of 5 second periods in a day
inrate(1:t) = 0; %influx at time t
L2(1:t) = 0; %number of cars leaving tollbooths at time t
```

以5秒为一个时间步长,我们可以根据交通流量函数可得到任何一个时刻的交通流量inrate,相关程序如下:

```
h = linspace(0,24,17281); %indexing vector for each 5 second time period
a0 = 41.68;
```

```
a = [-16.38, -18.59, 3.572, 7.876, -0.5048, -2.97, 0.2518, 0.5785];
b = [12.53, 0.6307, -13.67, 0.4378, 6.93, 0.4869, -1.554, -0.5871];
omega = 0.2513; %all Fourier Coefficients
inrate = a0 * ones(size(h));
for n = 1:8
    inrate = inrate + a(n).*cos(n.*h.*omega) + b(n).*sin(n.*h.*omega);
end

inrate = inrate * lanes
```
....

流量inrate所表示的意思是,单位时间内到达的车的数量,其倒数为来一辆车要经历的时间(单位:min),考虑到$1min = 12s$,则第k辆车到达要花费的时间为$12 \cdot (1/inrate(k))$, 因此我们可以得到每辆车到达的时刻(以时间步长,即5秒为单位):

....
```
A(1) = 12*( 1/inrate(1) ); %arrival time of car 1 in terms of inrate

for j = 2:i
    k = floor(A(j-1));
    if k == 0
       k = 1;
    end
    A(j) = A(j-1) + 12/inrate(k); %arrival time of car i in terms of inrate
end
```
....

我们将每辆车看作一个体,那么对于第i辆车,其离开时间(Leaving time),开始服务的时间(Start time),到达时间(Arrival time),服务时长(Service time)及等待时间(Waiting time)有如下关系:

$$\text{Leaving time}(i) = \text{Start time}(i) + \text{Service time}(i)$$

$$\text{Waiting time}(i) = \text{Leaving time}(i) - \text{Arrivla time}(i)$$

某辆车一到达收费站就有空着的收费亭为其服务,则对于该车辆其开始服务时间(Start time) 等于其到达时间(Arrival time); 如果某辆车到达收费站后暂时没有空闲的收费为其服务,我们假设任何一辆车将会选择队列最短的收费亭,因此该车辆的开始服务时间(Start time)等于当前所有收费亭最快结束服务的时间(min(last)).当然,服务亭结束服务的时间(Last time)等于其服务的上一辆车离开的时间.如果认为服务时长服从以2.4个时间步长(12秒)为均值的指数分布,则根据以上公式和假设,可算得每个车的离开时间和等待时间, 以及每个收费亭结束服务的时间:

....
```
last(1:booths) = 0; %last time at which a car left a particular booth
mu = 2.4; %mean service duration time in 5-second periods
S = exprnd(mu,1,i); %service time as an exponential random variable
for j = 1:i
    for k = 1:booths
        if (last(k) == min(last))
           B(j) = k; %find booth that was/will be emptied soonest
        end
```

```
    end
    if A(j) > last(B(j)) % if there is an empty booth, then...
       Start(j) = A(j); % start right away
       L(j) = Start(j) + S(j);
       last(B(j)) = L(j);
    else %if not...
       Start(j) = last(B(j)); %start once the soonest one becomes available
       L(j) = Start(j) + S(j);
       last(B(j)) = L(j);
    end
end

W = L - (A + S); %waiting time is line time - (arrival time + service time)
```

我们还可以计算任一时间步长,在收费站车的数量. 对于某辆车,它在它到达时间和离开时间的这段时间内在收费站. 因此m时刻在收费站的数量可计算如下:

```
for j = 1:i
    k = ceil(L(j));
    if k > t
       k = t;
    end
    for m = ceil(A(j)):k
        N(m) = N(m) + 1; %counts the number of people in line
    end
end
```

另外,定义L2向量来统计任一时间步长时离开收费的车辆数(流出量),比如第k辆车离开的时间是L(k), 则在L2相应位置ceil(L(k))的元素上加1,表示有一辆车在此刻离去.把所有此刻离去的车都加在L2的第ceil(L(k))个的元素上, 就可得到第ceil(L(k))个时间步长离开的车辆了

```
for k = 1:i
    if L(k) ≤ t
       L2(ceil(L(k))) = L2(ceil(L(k))) + 1; %creation of L2, outflux from tollbooths
    end
end
```

最后,统计一下所有车辆的平均等待时间,等待车辆的平均等待时间,最大等待时间:

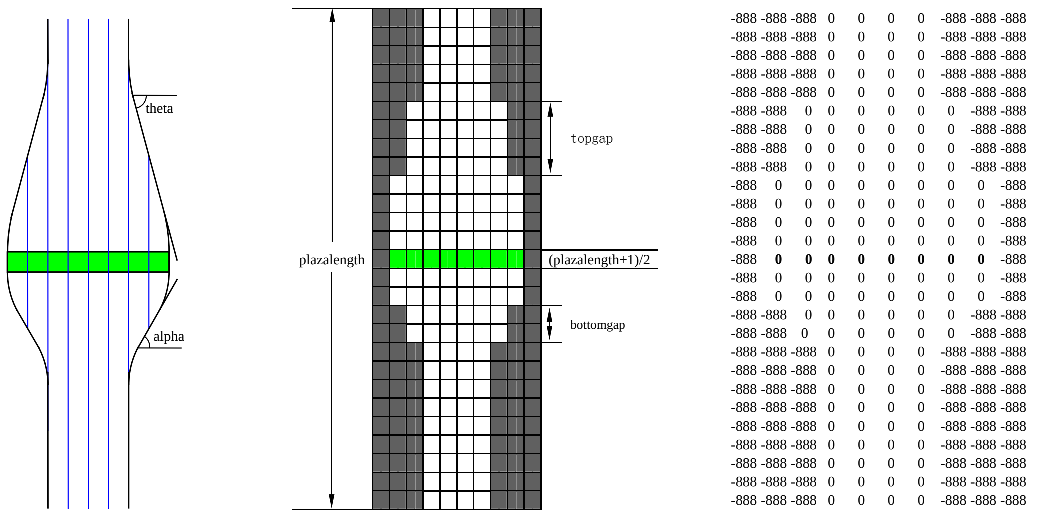
```
k = length(W);
for j = 1:i
    if W(j) == 0
       k = k-1;
    end
end
totalavgwait = sum(W)/i/12/60;
carsavgwait = sum(W)/k/12/60;
```

```
maxwait = max(W)/12/60;
```

3 元胞自动机模型

3.1 收费站平面图的矩阵化

本文要研究的是对于某一高速公路,设置多少收费亭最佳. 对于某一高速公路,可以测试不同数目的收费亭情况,然后从时间角度来判断哪种情况最佳. 对于元胞自动机模型,需要根据给定的高速公路(车道数L),和收费亭数目B以及与收费站结构有关的参数生成相应的格子(矩阵).每个格子可能的状态有三种:用1表示车辆,0表示空位,-888表示不可进入区域.



(a) (b) (c)

图 3: 收费站平面图的离散化

图3(a)为收费站平面示意图, 除了B和L,与其结构相关的参数还有θ和α, 在离散的过程中,通过topgap和bootomgap对θ和α加以考虑(如图3(b)), 其关系为

$$\text{topgap} = [[\tan(\theta)]], \text{bootomgap} = [[\tan(\alpha)]]$$

其中$[[x]]$表示对x取整.这样,从图3(a)到3(b)就完成了收费站的离散化(格子化), 就可以用如图3(c)所示的数组plaza在计算机中将收费站表示出来(收费站中没有车,因此plaza数组中没有值为1的元素). 通过以下程序,可以生成图3(c)所示的数组plaza.

```
B = 8
L = 4
plazalength=27;
topgap = 4;
```

```
bottomgap = 2;
plaza = zeros(plazalength,B+2);
plaza(1:plazalength,[1,2+B]) = -888;
for col = 2:B/2 - L/2 + 1
    for row = 1:(plazalength-1)/2 - topgap * (col-1)
        plaza(row,[col, B+3-col]) = -888;
    end
    for row = (plazalength+3)/2 + bottomgap*(col-1):plazalength
        plaza(row,[col, B+3-col]) = -888;
    end
end
```

....

上面所考虑的是车道数-收费亭=偶数($\mathrm{mod}(B-L,2)=0$)的情况,对于车道数-收费亭=奇数($\mathrm{mod}(B-L,2)=1$)的情况. 由于其不再具有对称性,需单独考虑. 对于图3中的收费站,如果需要设置收费亭的数量B为7, 则直接将图3(c)的数组plaza的倒数第二列元素全设置为-888. 相应的程序如下:

....

```
B = 7
L = 4
plazalength=27;
topgap = 4;
bottomgap = 2;
plaza = zeros(plazalength,B+2);
plaza(1:plazalength,[1,2+B]) = -888;
plaza(1:plazalength, B+3) = -888;
for col = 2:(B+1)/2 - L/2 + 1
    for row = 1:(plazalength-1)/2 - topgap * (col-1)
        plaza(row, [col, B+4-col]) = -888;
    end
    for row = (plazalength+3)/2 + bottomgap*(col-1):plazalength
        plaza(row, [col, B+4-col]) = -888;
    end
end
```

....

根据以上分析,我们将上述两种情况的收费站数组plaza的生成写成函数create_plaza(B, L),具体见附录.

3.2 新车辆的生成

在交通流的傅里叶展开中, 得到了流通流量函数$F(t)$,其单位是cars/min, 本文的仿真时间步长为$\Delta t = 2.5\mathrm{s} = 1/24\mathrm{min}$. 因此第$n$个时间步长($t = n\Delta t$)到达系统的车辆为

$$\mathrm{entry}(n) = F(t)\cdot\Delta t = F(t)\ \mathrm{cars/min}\ \cdot\ \frac{1}{24}\mathrm{min} = F(t)/24\ \mathrm{cars}$$

实际上,由于元胞自动机中新到来的车辆数只能是整数,还必需对entry(n)进行取整,以作为第n个时间步长到达的车辆数.生成entry向量的函数如下:

....

```
function entry = create_entry(T,L)
k = linspace(0,T,T.*60.*24);
a0 = 41.68;
entry = a0.*ones(size(k));
a = [-16.38, -18.59, 3.572, 7.876, -.5048, -2.97, 0.2518, 0.5785];
b = [12.53, 0.6307, -13.67, 0.4378, 6.93, 0.4869, -1.554, -0.5871];
omega = 0.2513;
for n = 1:8
    entry = entry + a(n).*cos(n.*k.*omega) + b(n).*sin(n.*k.*omega);
end
k = k.*1440;
entry = entry./24;
entry = round(entry);
```

设第i个时间步长应该进入的车辆数为entryi(entryi=entry(i)), plaza数组第一行为0的位置的个数为n(入口处可容纳新进入车的最大数量). 则在第i个时间步长中实际新进入的车辆数只能为min(entryi, n). 因此, 需要新车的进入就转换为在plaza数组的第一行, 随机的选择min(entryi,n)个为0的位置,并设值为1的问题. 首先找出plaza数组第一行为0的位置unoccupied及第一行为0的位置个数n, 再用randperm函数将$1, 2, \cdots, n$位置进行随机排序, 并选择前min(entryi,n)个位置设值为1. 具体程序如下:

```
function plaza = new_cars(plaza, entryi)

if entryi > 0
    % Find the empty lanes of the entrance where a new car can be add.
    unoccupied = find( plaza(1,:) == 0 );
    n = length(unoccupied); % number of available lanes
    x = randperm(n);
    for i = 1:min(entryi,n)
         plaza(1, unoccupied(x(i)) ) = 1;
    end
end
```

4 前进规则

每个时间步长,系统中每一辆都要根据一定前进规则前进. 分为三段从车前进方向面向后扫描每一辆车,分别是收费亭前面, 收费亭和收费亭后面. 对于收费亭前或收费亭后的车(i =(L+1)/2),如果该车前面位置被一辆车占用(plaza(i+1, j) = 0),则该车被标记为-2,为换道作准备; 否则, 该车将以概率prob前进到它前面的位置. 比如对于收费亭前方和车辆, 其前进规则的matlab程序如下:

```
[L, W] = size(plaza);
prob = 0.7;
for i = (L-1):-1:((L + 1)/2 + 1)
    for j = 1:W
        if plaza(i,j) == 1
```

```
            if plaza(i+1, j) ~= 0
                plaza(i,j) = -2;
            elseif prob ≥ rand
                plaza(i,j) = 0;
                plaza(i+1, j) = 1;
            end
        end
    end
end
```

对于收费亭中的车(i = (L+1)/2), 我们用plaza矩阵中相应位置的元素记录其在收费亭中的时间,当这个服务时间等于或大于标准服务时间(plaza(i,j) ¿= delay),且收费亭前面位置为空位(plaza(i+1,j) == 0)时,那么就该车就离开当前收费亭前进到前面的位置. 相应的matlab程序如下:

```
delay = 3;
for i = (L+1)/2
    for j = 1:W
        if plaza(i,j) > 0
            if plaza(i,j) ≥ delay & plaza(i+1,j) == 0
                plaza(i,j) = 0;
                plaza(i+1,j) = 1;
            else
                plaza(i,j) = plaza(i,j) + 1;
            end
        end
    end
end
```

对于收费亭后的车辆,其前进规则的程序与收费亭前类似, 具体附录中的move_forward函数.

5 换道规则

当某辆车前面的位置被别的车辆占据, 则该辆得考虑换道, 这种车辆在4前进规则中已经被标记为-2. 车辆以概率prob进行换车道,即对某一辆车,当产生的0-1间均匀分布的随机数rand ¡ prob时, 该车将进行换道. 该车向左或右换的概率相同,为0.5, 当向某一方向换道失败后,则再尝试另一方同. 如果两个方向都换道失败(左右方向的车道都有车辆占据), 那么该车保持不同, 并取消标记(-2改为1). 相关程序如下:

```
if plaza(i,j) == -2
    if rand < prob %chance turn will be made
        if rand > 0.5 %will attempt left
            if plaza(i, j-1) == 0
                plaza(i, j-1) = 1;
                plaza(i, j) = 0;
            elseif plaza(i, j+1) == 0
                plaza(i, j+1) = 1;
```

```
                plaza(i,j) = 0;
            else
                plaza(i,j) = 1;
            end
        else %will attempt right
            if plaza(i, j+1) == 0
                plaza(i,j+1) = 1;
                plaza(i,j) = 0;
            elseif plaza(i, j-1) == 0
                plaza(i, j-1) = 1;
                plaza(i,j) = 0;
            else
                plaza(i,j) = 1;
            end
        end
    else
        plaza(i,j) = 1;
    end
end
```

以上程序为换道规则的主要程序, 换道规则是由switch_lanes函数实现的, 见附录.

6 花费的总时间及离开车量的计算

花费的总时间是指所有车经过整个收费站的时间总和, 或者说所有车辆在plaza矩阵中存在的时间总和. 当某辆车在plaza中逗留一个时间间隔,那么plaza矩阵中必有一个大于0的元素与之相对应, 比如一辆车经过整个收费站共用去10个时间步长, 那么在这10步长中表示该车的元素必然出现在plaza矩阵中. 因此, 我们只需把每个步长plaza中大于0的元素的个数算出来,就可以变相的计算花费的总时间. 对于任一步长, 由一个简单的函数compute_wait来实现.

```
function time = compute_wait(plaza)
time = sum(sum(plaza>0));
```

在主程序, 每个步中都调用该函数, 并所反回值累加起来就可以得到花费的总时间了.

能直接反应问题的不是花费的总时间, 而应该是每辆车的平均费时. 因此还需要计算离开收费站的总车辆数. 对于每个步长, 通过统计plaza矩阵中最后一行中大于0的元素的个数,就可以得知该时刻离开收费站的总车数了, 具体是通过compute_output函数来实现的:

```
function count = compute_output(plaza)
count = sum(plaza(end,:)>0);
```

统计完要离开系统的车辆后, 需要将车量从系统中移除, 即将plaza矩阵中最后一行中大于0的元素设回0，具体见附录中的clear_boundary函数。

7 主程序

最后, 编写一个主程序, 调用各个函数来完成这一仿真, 对于8个收费亭,6条车道的收费站, 可以编写如下主程序:

```
B = 8; %number booths
L = 6; %number lanes in highway before and after plaza
T = 24; % # hrs to simulate
global plazalength;
plazalength = 101;
plaza = create_plaza(B,L);
h = show_plaza(plaza,B,NaN);

entry_vector = create_entry(T,L);
waiting_time = 0;
output = 0;
for i = 1:T*1440
    plaza = move_forward(plaza); %move cars forward
    plaza = new_cars(plaza, entry_vector(i)); %allow new cars to enter
    plaza = switch_lanes(plaza); %allow lane changes
    %compute waiting time during timestep i
    waiting_time = waiting_time + compute_wait(plaza);
    output = output + compute_output(plaza);
    %================================================
    h = show_plaza(plaza,B,h);
    drawnow
    %================================================
    plaza = clear_boundary(plaza);
end
show_plaza(plaza,B,h);
W = waiting_time/output;
xlabel({strcat('B = ',num2str(B)), ...
       strcat('mean cost time = ', num2str(round(W)))})
```

其中show_plaza函数是用来将plaza矩阵图形化的,有具体见附录. 在仿真中,为了快速的得到结果,可以通过注释掉两个%==...==之间的内容, 不显示动画来节少计算的时间.

8 仿真效果

通过仿真，可以得到车辆的平均费时。图4是对6条车道，6-12个收费亭的仿真效果图，图下标有相应仿真得到的车辆平均费时(mean cost time).

9 评论

本文程序源于论文The Booth Tolls for Thee的作者。本人已经修改了程序中的错误，简化了多处程序语句，并对程序进行了优化及图形化(show_plaza函数及相关部分由本人编写，指在将仿真过程图

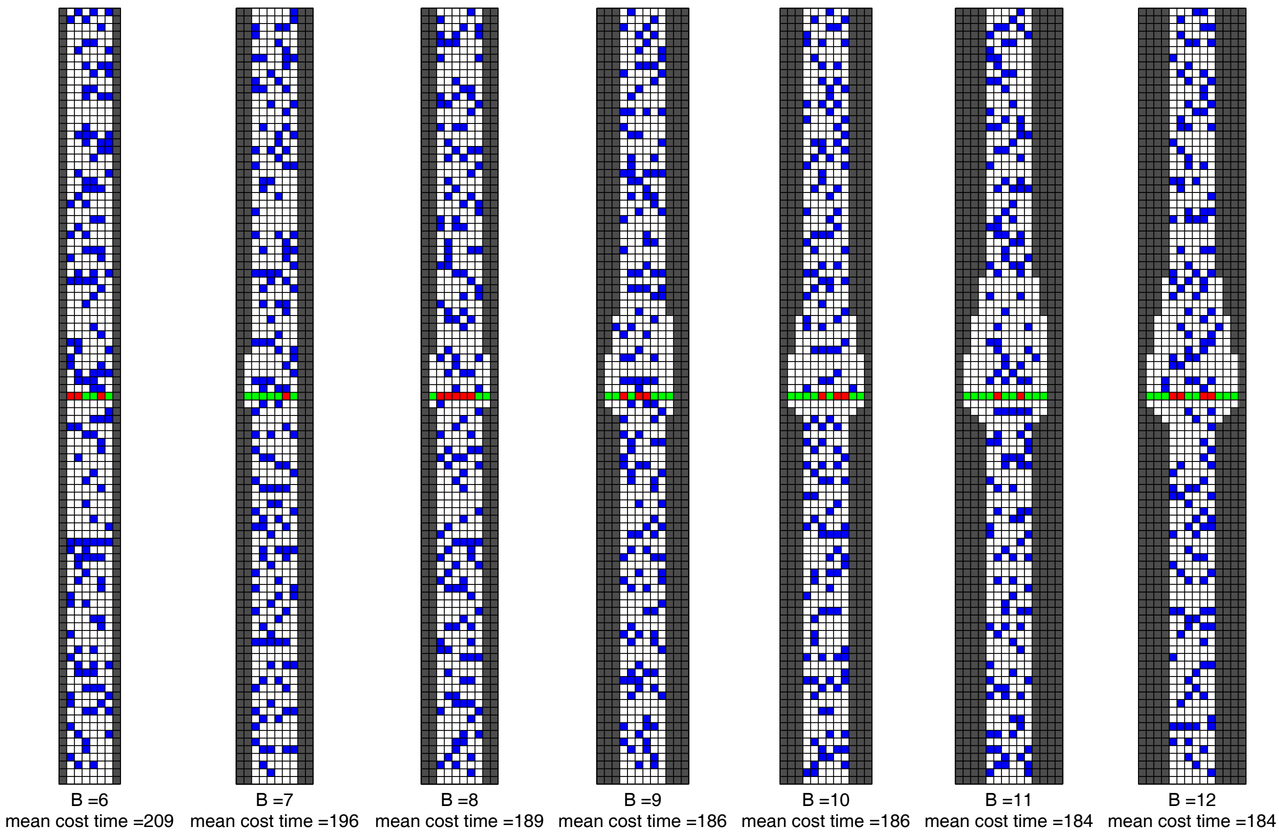


图 4: 仿真结果

形化，动画化)。但由于模型本身的局限性，使得该模型和仿真还有较多可以改进的地方:

- 收费站结构失真。对于**收费亭-车道数=奇数**的情况，生成的收费站矩阵很不对称，对仿真结果可能造成影响。

- 所有车的速度都一样，要么1,要么0. 虽然已经用概率模拟车辆的停停走走，但车辆的速度还不够丰富，不能很好的模拟车的减速，加速等行为。

- 改换车道具有方向性及盲目性。

  方向性：由于循环具有方向很，从左到右顺次对每个车进行换道，因此使得改换车道具有了方向性。比如中间车道有个空位，两边都有车想换到该位置，但由于循环是从左向右扫的，因此左侧的车换到该车道上的概率要大些，但实际上这两个车换到中间车道的概率应该一样。

  盲目性：根据给定的换道规则，当一辆车前面被别的车占据，则它就开始以一概率换车道。但如果它换完车道后情况不比没换前好(比如换完后的车道前面仍有一辆车占据着)，那么它就没必要换了。因此改换车道后可能发现白换，这使得换道具有盲目性。

- 对花费时间的统计太过宏观。只能统计得到平均花费的时间，而不能得到每个车，每个时段通过收费站所要花费的时间。而这对于车流量有变化的情况，或者要保证每个车辆的个体利益时，尤为重要。比如要保证任一车的花费时间都不高于平均花费时间的3倍时，仅能得到平均花费时间是不能检验某种收费站结构是否满足要求的。

本文参考05年关于收费站的其它多篇特等奖论文，编写了新的收费站元胞自动机仿真程序，改进了上述缺点,程序见附录, 具体实现过程及步骤不再祥述。得到的结果如图5. 注意，图5和图4中的每格代表的距离不同，仿真的时间步长也不同。因此两结果的平均花费时间可存在较大差异。

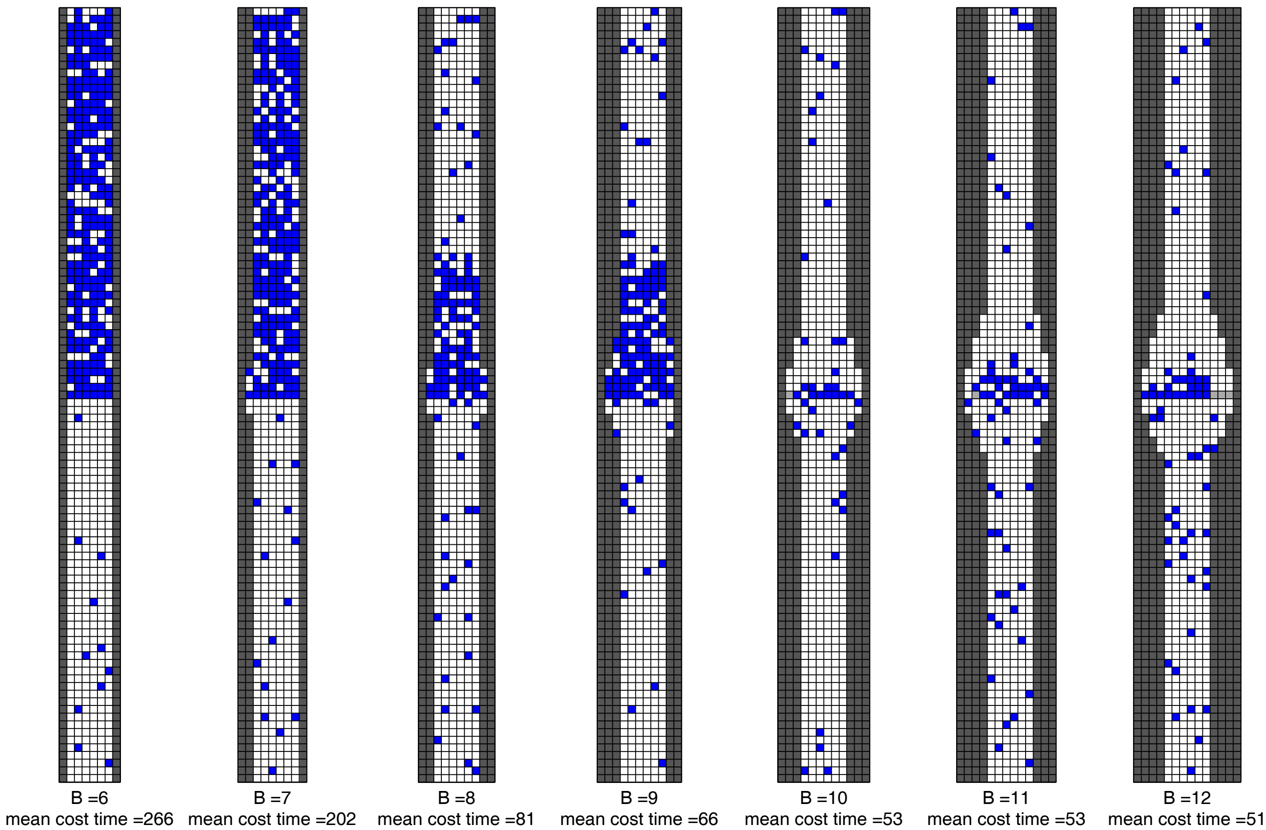


图 5: 仿真结果