交通系统仿真课程报告

题目：Simulation of Transit Line Operation

姓　　名：王倩妮

班　　级：交通 2015-02 班

学　　号：2015112956

任课教师：范文博

2018 年 4 月 3 日

目录

一、 问题描述.....1
二、 问题分析.....1
2.1 问题分析构建流程.....1
2.2 主体分析.....2
2.3 主体属性.....2
2.4 主体行为与过程分析.....3
2.5 主体间关系.....4
三、 模型的构建.....5
3.1 搭建环境.....5
3.2 仿真建立与程序实现.....5
3.2.1 公交车运行模型.....5
3.2.2 行人步行模型.....5
3.2.3 程序实现过程.....5
四、 仿真的运行.....6
4.1 仿真可视化.....6
4.2 仿真参数输出设置.....6
4.2.1 公交车载客量.....7
4.2.2 公交站点等待（排队）人数.....7
4.2.3 乘客通勤时间.....7
五、 仿真运行结果分析及改进方向.....7
六、 总结.....9
附录.....10

一、问题描述

Simulation of transit line operation, that is to say, simulation the bus operation and stop behavior, the ahead/leave station by walking, waiting, and aboard/alight behavior of passenger with certain transit demand (e.g., aboard and alight demand), station location and transit departure frequency. Some the simple visualization and some system performance index are required to output, such as the total commuting time of passenger, the vehicle load factor, et al.

制作一个交通线路运行的仿真，仿真中需要体现公交的运行、停止行为，行人前往、离开公交站点的步行、等待行为，乘客的出现是交通需求的产生，需体现出乘客上下公交车的行为，在这一过程中，站点分布与公交发车频率可以呈一定规律性分布。在制作仿真的过程中需要达到可视化以及输出一些系统运行参数的要求，例如可以输出乘客的总体通勤时间、公交车车内容量等等。

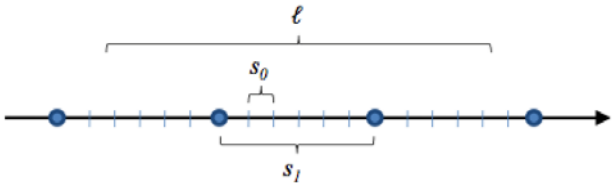


图 1 问题描述示意图

二、问题分析

2.1 问题分析构建流程

为构建题设仿真，主要依照如下图所示步骤进行。首先，确定仿真过程中的主体，进而分析每个主体在仿真过程中具有的属性，即一些变量与其对应的值。确定以上内容后，进行主体行为与过程分析，如公交车运行过程中的启动-运行-停止；行人步行-等车-上车-乘车-下车-步行；站点人员到达-人员离开等主体的各项行为与变化过程，在这一过程中，及时补充必要的主体属性。由于系统中各部分存在相互作用，因此需要通过主体间相互关系的分析将各个主体串联起来。在做好以上各项分析后，即可编程实现并进行参数输出与可视化，在这一过程中若发现系统运行出现问题则需要及时进行调整。最终得到满意仿真效果后进行分析与评价。

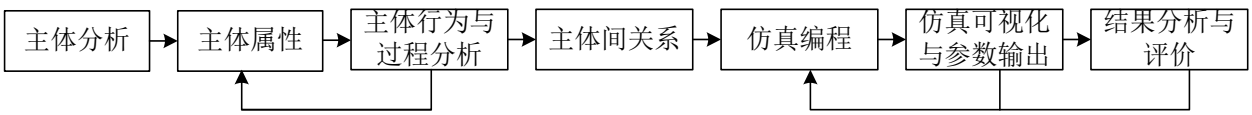


图 2 仿真问题分析及构建流程图

2.2 主体分析

对于公交运行的全流程分析后，可归纳出仿真世界、公交车、行人、公交线路等四个主体，由于交通是“时间-空间”过程，主体间依靠仿真时间的推移而相互联系、作用。

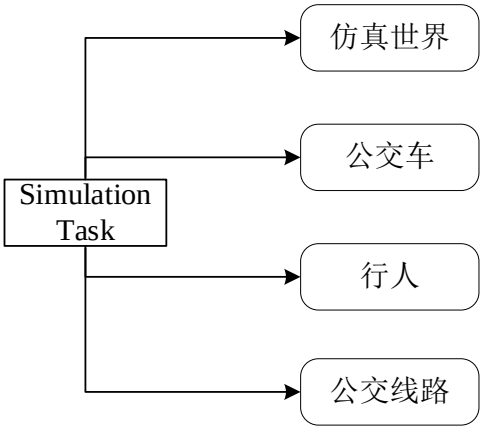


图 3 仿真主体

2.3 主体属性

由 2.2 分析知，本次仿真共构建 4 个主体，因此需确定与主体相关的属性。这其中省略了部分可由基本属性推算得到的中间属性。

（1）仿真世界（仿真环境）及其属性

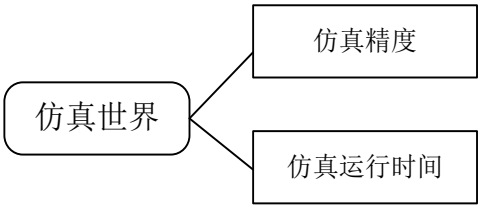


图 4 仿真世界及其属性

（2）公交车及其属性

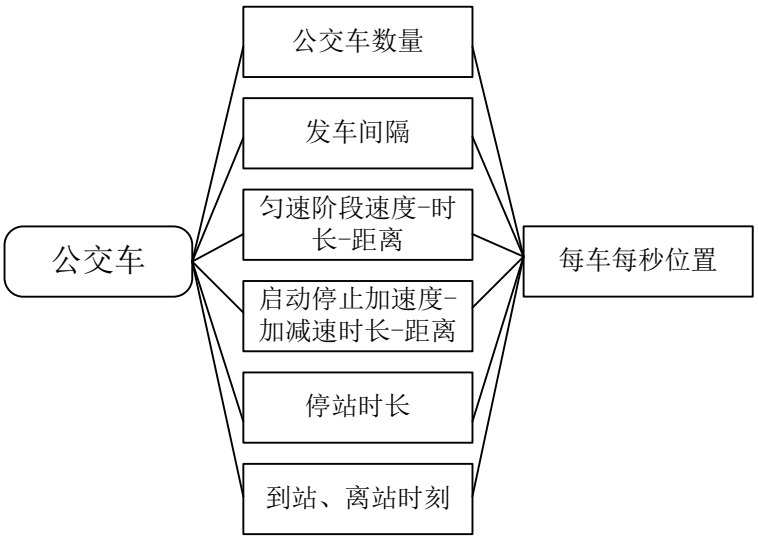


图 5 公交车及其属性

（3）行人及其属性

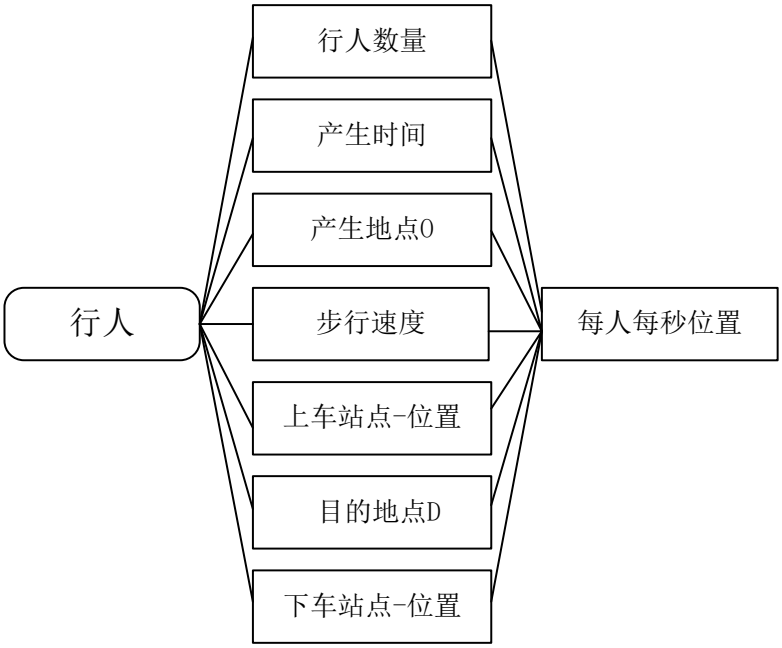


图 6 行人及其属性

（4）公交线路及其属性

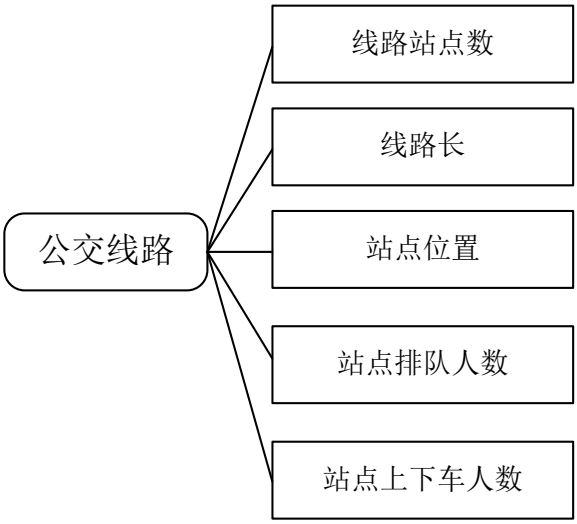


图 7 公交线路及其属性

2.4 主体行为与过程分析

主体在仿真过程中具有各自的行为，现将主体在仿真中的行为分析如下：

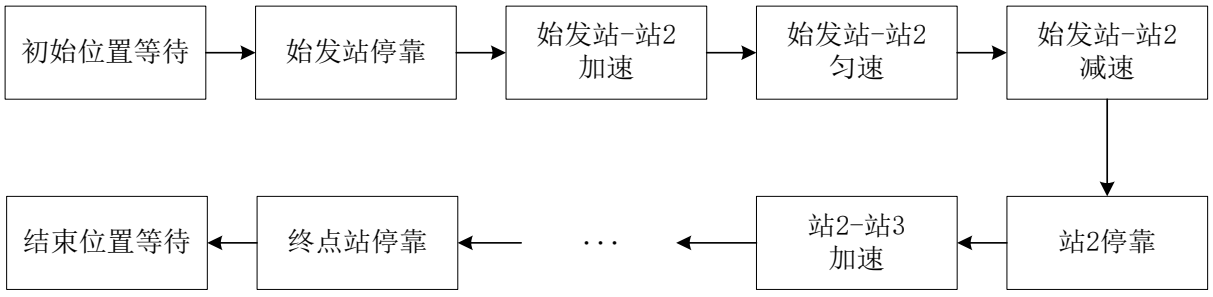


图 8 公交车主体行为

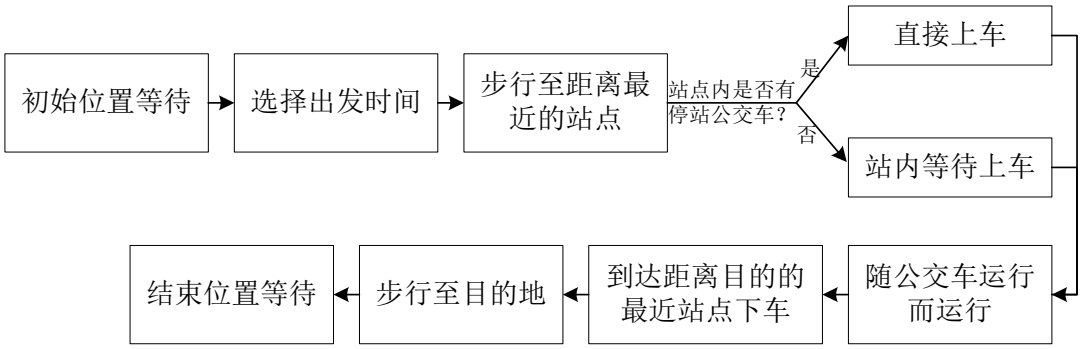


图 9 行人主体行为

2.5 主体间关系

仿真的运行及参数的计算与主体间的相互关系有着密切的联系，整个仿真系统也在仿真运行时间 T 的推移下相互串联起来，在完成了主体属性的分析后，需要对于主体间的相互关系进行分析，以达到串联仿真各个主体的目的。

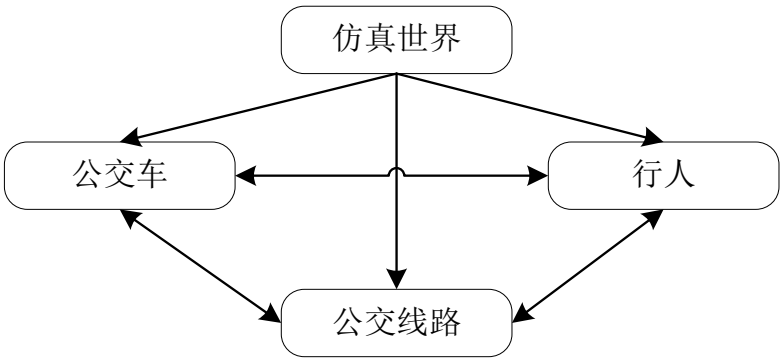


图 10 主体间关系示意图

由上图可知公交车、公交线路、行人三个主体均以仿真世界为背景进行串联，即在仿真世界背景下建立，这种关系主要体现为一种“时间-空间”关系。而公交车、公交线路、行人三者之间又有着相互的联系。

（1）公交车-公交线路关系

公交车在公交线路上运行，在站点完成停靠，在线路段完成加速-匀速-减速过程，依靠公交线路属性与公交车属性确定公交车的时空位置。并使公交车与站点间产生联系，记录公交车到达、离开各个站点的时间，经过每个站点及线路上通过的公交车数量等，这一过程也为人与公交车间的联系奠定基础。

（2）行人-公交线路关系

行人在某刻于空间中某处生成，并于最终到达空间中某处。联系 O、D 位置与公交线路站点，可以确定距离 O、D 位置最近的公交车站；行人至公交站点后，可能在公交站点发生等待行为；行人到达目的站点后完成下车。

（3）公交车-行人关系

公交车与行人间的关系通过公交线路间接联系起来，当行人与公交线路联系（行人到达、停留在公交站点）与公交车与公交线路联系（车辆到达、停留公交站点）同时发

生时，公交车与行人建立联系；而后，行人位置与公交车位置进入一致阶段；当公交车、行人与公交线路联系（公交车到站、行人到达目的车站）时，公交车-行人关系结束。在这一过程中可动态测得公交车载客量等参数。

三、模型的构建

3.1 搭建环境

- 编程环境：Anaconda Spyder 编译器下使用 Python3 编写
- 矩阵运算：Numpy
- 仿真可视化：Matplotlib

3.2 仿真建立与程序实现

3.2.1 公交车运行模型

公交车运行模型主要采用牛顿运动定律。

$$S_{bus}=\begin{cases} vt & (匀速阶段) \\ \dfrac{v^2}{2a} & （加减速阶段） \end{cases}$$

其中：S 为在线段上运行时各阶段的位移，v 为公交车匀速阶段速度，a为公交车在加减速阶段的加速度。

$$T_{停站}=t$$

公交车停站时间采用定值 t。

3.2.2 行人步行模型

行人在步行阶段运行为匀速运动。

$$S_{pedestrian}=v_{人}t$$

3.2.3 程序实现过程

仿真建立过程参照以下步骤，具体计算过程参见附录代码注释。

（1） 定义仿真所需全局变量，包括如：运行时间、公交车速度、加速度、行人步行速度、线路长度、站点数量等。

（2） 定义 bus_ride 函数，确定公交发车、到站、离站、到达时刻。

（3） 定义公交车行驶于两站之间的过程中，每一秒的与“相对起点”间的距离。

（4） 定义 bus_location 函数，确定一辆公交车在仿真开始至仿真结束过程中，每一秒的位置。

（5） 计算每辆运行的公交车到达、离开每一站的时刻。

（6） 调用 bus_location 函数，确定仿真过程中所有公交车在每一时刻的位置。

（7） 确定在仿真过程中不能完整开行的公交车、不能完成行程的行人不进行仿真。

（8） 计算行人仿真过程中的参数，包括乘客产生时间、乘客到达上车站时间、乘客产生地点、乘客产生与上车站距离、乘客消失地点、乘客下车站与乘客消失距离、乘客上车位置、乘客下车位置、上车前步行时间、下车后步行时间、起始站点、下车站点。

（9） 确定仿真过程中，每一位行人每一时刻的位置。

（10）计算输出的 3 个指标。

（11）利用 matplotlib 模块 animation 函数输出仿真。

四、仿真的运行

4.1 仿真可视化

图 11 仿真运行图

如上图所示是仿真的可视化效果，其中红色点代表公交车，蓝色“×”代表行人，S1-S6 代表 6 个均匀分布的公交站点，对应的 x 坐标为站点的位置坐标，行人、公交车均做一维运动。右上角显示仿真时间的变化。行人于上部“PS route”行产生、完成步行、等待上车；于下部“PS route”下车、步行至目的地、消失。行人及公交车的初始终止位置（与仿真过程无关）位于灰色文字示意位置。

在仿真过程中行人与公交车均做一维运动，因此可能会有行人间的重叠现象发生。

4.2 仿真参数输出设置

交通仿真指用仿真技术来研究交通行为，是一门对交通运动随时间和空间的变化进行跟踪描述的技术。其含有随机特性，可以是微观的，也可以是宏观的，并且涉及描述交通运输系统在一定期间实时运动的数学模型。仿真参数输出是仿真结果的一个部分，可以用于进行：交通工程理论研究、道路几何设计方案评价、交通管理系统方案评价、智能交通系统 ITS 效果评价等。仿真输出一般需要输出在后期研究中需要使用到的变量以及在难以在可视化过程中体现的量。

4.2.1 公交车载客量

公交车载客量是指在仿真运行环境下，每辆公交车上实时搭载的乘客数量，是随时间动态变化的数据。由于在可视化过程中，行人示意点与公交车示意点会发生重合现象，不便于人们直观感受载客量的直观变化，因此确定公交车载客量为输出量之一。

公交车载客量可以用于判断在特定的需求量状况下，现有公交班次、发车间隔等交通供给是否能够满足交通需求，进而可以用于调整公交车运行计划、方案。

4.2.2 公交站点等待（排队）人数

公交站点等待（排队）人数是指在仿真运行条件下，每个站点实时等待上车的人数，是随时间动态变化的数据。由于站点位置固定，在可视化过程中行人候车时行人示意点出现重合状况，此时不易直观感受公交站点排队人数，因此将此指标作为输出量之一。

由于本仿真未设每辆公交车容量上限，即只要来车，均可上车。实际情况下每辆公交车会有容量上限，未来可制作依据站点排队反馈控制的公交发车安排仿真，即通过实时监测到站点排队人数，安排公交车的发车班次、路段上运行公交的速度以及停站时间长、停站地点等，以达到提高公交的时效性与便捷性的目的。

4.2.3 乘客通勤时间

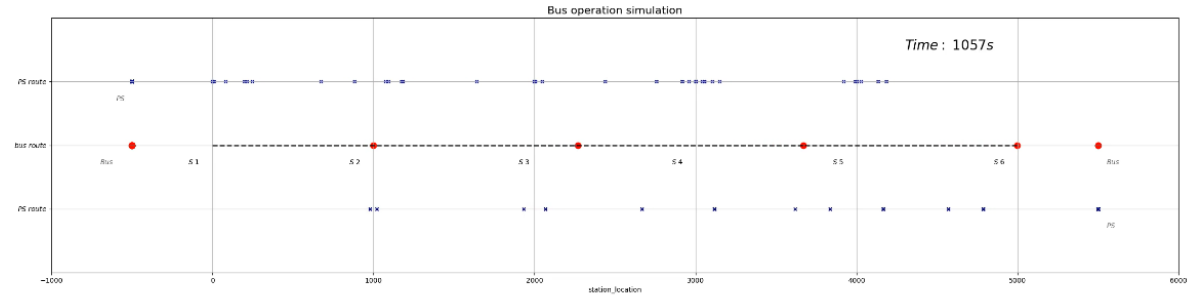
乘客通勤时间指在当前仿真环境下产生的出行需求完成由 O 到 D 的出行所需要的总时间，是个体通勤时间的叠加。为提高公共交通的服务水平，可以通过减小居民出行的费用（包括时间消耗、金钱支出等）、提高公共交通准点性、舒适度等措施，乘客通勤时间这一指标即可量化时间消耗，为公交运行状况评价提供依据。但随机生成因素对此值影响较大，具体使用“累积”值还是“平均”值还需进行进一步讨论。

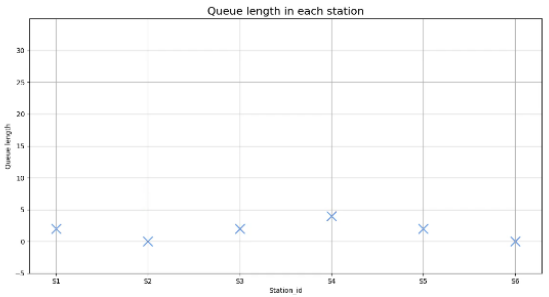
未来，也可以对指标进行拓展，加入如乘客累计等待时间、公交车容量限制等指标与影响因素，对仿真过程进行进一步量化、限制。

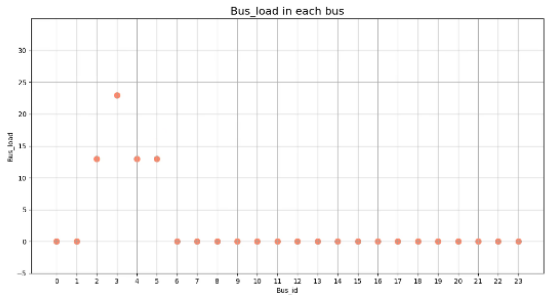
五、仿真运行结果分析及改进方向

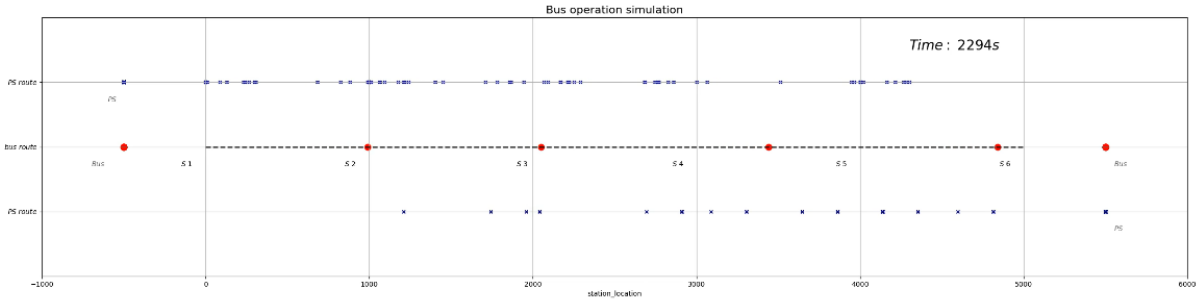
在 T = 5000s （仿真运行时间），V = 10 m/s（公交车匀速速度），A = 0.5 m/s^2 （公交加速度），Bus_stop = 20 s（公交停站时间），L = 5000m（公交路线长），Bus_t = 180 s（公交发车间隔），S_num = 6（站点数目），N = 1000 人（仿真周期内乘客数（需求量））等条件下，运行仿真。

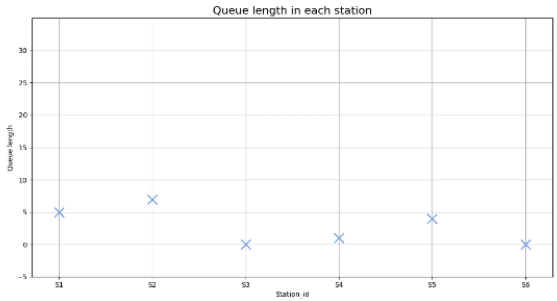
在仿真过程中计算输出的参数，其中公交车载客量、公交站点等待（排队）人数两个指标通过条形图的输出，通过图像直观反映该指标随时间的变化过程；乘客通勤时间则在计算后直接进行输出。如下图所示，是仿真过程中几个时刻的参数情况：

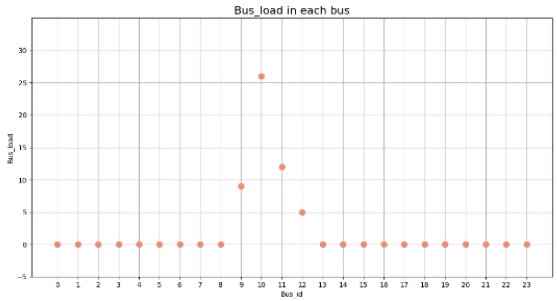

Bus operation simulation
Time : 1057s
PS route
bus route
PS route
S 1
S 2
S 3
S 4
S 5
S 6
station_location

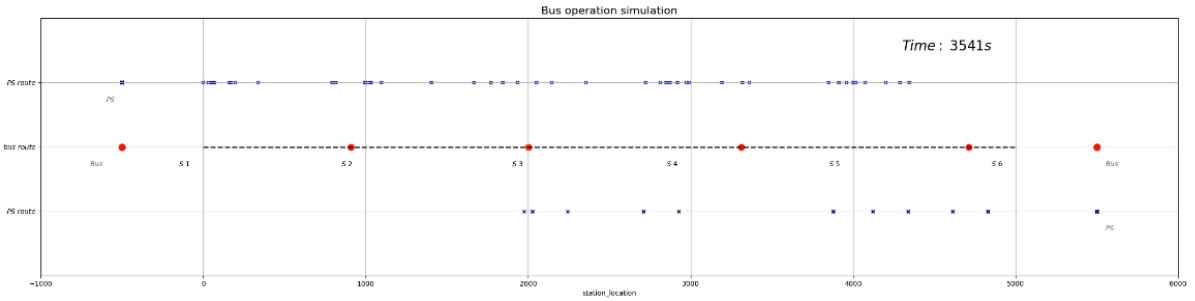

Queue length in each station
Queue length
Station_id

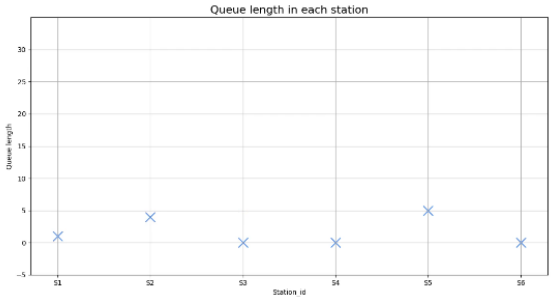

Bus_load in each bus
Bus_load
Bus_id


Bus operation simulation
Time : 2294s
PS route
bus route
PS route
S 1
S 2
S 3
S 4
S 5
S 6
station_location


Queue length in each station
Queue length
Station_id


Bus_load in each bus
Bus_load
Bus_id


Bus operation simulation
Time : 3541s
PS route
bus route
PS route
S 1
S 2
S 3
S 4
S 5
S 6
station_location


Queue length in each station
Queue length
Station_id

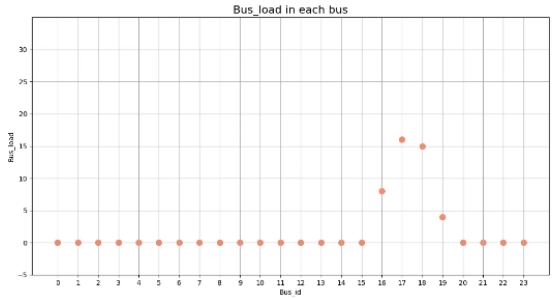

Bus_load in each bus
Bus_load
Bus_id

图 12 仿真运行结果

由上图可知，在每图左下角显示了当前时刻各站点的排队人数情况，右下角显示各趟公交承载的人数状况。该指标显示较为直观，易于直接由图中读出人数。乘客累计通勤时耗与仿真全过程有关，不随时间实时变化，在仿真结束后输出。

接下来，可以调整如站点间距、发车间隔、行车速度、行车加速度、停站时间等变量值，观察输出指标在数值及趋势上的变化情况。

未来，还可以对本仿真过程进行进一步改进。可以通过加入车辆跟驰、车辆换道、停站时间闭环控制等过程，使仿真更加真实。在仿真设计方面，编写前段 GUI 交互式界面，方便用户进行参数设置与调整。在代码编写方面，也应使用更为高效的编码模式，特别是减少结果生成（绘图）部分的耗时。

六、总结

本次仿真依靠 Python 及 numpy 矩阵运算模块进行编程，借助 matplotlib 绘图模块进行可视化步骤。在进行仿真构建的过程中极大锻炼了思维的逻辑性与协调性，也在一定程度上提高了编程能力。但由于本人未接触、学习过面对对象的编程，未考虑到可以使用类刻画实体，对函数进行有效规整，造成代码阅读、修改较为困难，需依靠大量注释表示的问题。与此同时，还有很多细节在这一过程中未考虑周全，如仿真精度设定等，也会在未来的学习、研究过程中争取做到更好。在完成课程作业的过程中，同学们间的沟通交流对于进度的推进、问题的修正与改进至关重要，一起钻研文档，搞清函数的使用方法；一起讨论，设置仿真刻画参数；一起交流，开阔构建思路……在这一过程中也要感谢老师们的耐心帮助，感谢范文博老师对成果提出修改意见，以及熊耀华老师对于我在使用 Python 中遇到的问题不厌其烦地解答。努力理清问题、推进进度的过程是一个不断超越自我、突破想象的过程，犹如完成一步步升级，到这也接近尾声了，希望通过一步步的努力，可以不断做到更好。

附录

文件名称：

文件 1：交通系统仿真报告王倩妮 2015112956.pdf；

文件 2：仿真程序王倩妮 2015112956.py；

文件 3：仿真结果视频王倩妮 2015112956.mp4；

文件 4：报告流程图王倩妮 2015112956.vsd；

仿真程序源代码：

```
# -*- coding: utf-8 -*-
"""
Created on Tue Apr  3 11:11:35 2018

@author: lenovo
"""

# -*- coding: utf-8 -*-
"""
Created on Wed Mar 21 09:50:33 2018

@author: lenovo
"""

# -*- coding: utf-8 -*-
"""
Spyder Editor
"""
###仿真精度为每秒进行###
###导入模块####
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation



###全局变量定义####
T = 5000 #仿真运行时间 s
V = 10 #匀速速度 m/s
A = 0.5 #站点加减速速度 m/s^2
Bus_stop = 20 #上下车时间长度 s
L = 5000.0 #公交路线长 m
Bus_t = 180.0 #公交发车间隔 s
S_num = 6 #包含始末站点的站点数目
S = np.linspace(0, L, num=S_num)   #站点一维坐标
S_L = L/(S_num-1) #站点距离均匀分布下站点间的距离
S_T = int(2*V/A+((S_L - (V*V/A))/V)) #站间时间
spd_up_time = int(V/A)#站间加速时间（用于定车的位置）
spd_dwn_time = int(V/A)#站间减速时间（用于定车的位置）
avr_time = S_T-spd_up_time-spd_dwn_time#站间匀速时间（用于定车的位置）
N = 1000 ###仿真周期内乘客数（需求量）

###定义公交发车、到站、离站、到达时刻#####
def bus_ride(bus_number, v=V, a=A, bs=Bus_stop, bt=Bus_t):
    start_time = int(bus_number*bt) #起始时刻
    arr = np.repeat(start_time, S_num)+np.arange(S_num)*S_T + np.arange(S_num)*bs#到达站点时刻
    lea = arr + np.repeat(Bus_stop, S_num)#离开站点时刻
    stop_time = lea[-1]#停止时刻
    period = stop_time - start_time#每班运行时长
```

```
    return(start_time, arr, lea, stop_time,period)

###定义公交车在每两站之间的时间中，每一秒的位置##
s_loc=[]
for k in range(S_T):
    if k<spd_up_time:#加速阶段
        s_loc.append(A*k*k/2)
    elif (k>=spd_up_time)&(k<(spd_up_time+avr_time)):#匀速阶段
        s_loc.append(A*spd_up_time*spd_up_time/2+V*(k-spd_up_time))
    else:
        s_loc.append(S_L-A*(S_T-k)*(S_T-k)/2)#减速阶段
s_loc = np.array(s_loc)

###定义每个公交车在仿真期间每一秒的位置###
def bus_location(bus_number):
    loc = np.ones((T))*(-500)#位置初始化
    for j in range(S_num):
        if LEA[bus_number,-1]<T:
            loc[ARR[bus_number,j]:LEA[bus_number,j]] =  S[j]#填满位于站点内部的 location
            if j!=(S_num-1):
                loc[LEA[bus_number,j]:ARR[bus_number,j+1]] = np.repeat(S[j],S_T)+s_loc#分段调用站间位置
            else:
                pass
            loc[LEA[bus_number,-1]:] = S[-1]+500#填满到站后的 location
        else:
            pass#仿真时间内开不完全程的车就不开了
    return(loc)

###计算每辆公交的进出站时刻###
    '''
    此处还有改进空间，bus_ride 可以返回更多数据
    '''
ARR=[]
LEA=[]
LOC=[]
for i in range(T):
    if i%Bus_t == 0.0:
        _,arr,lea,_,_ = bus_ride(int(i/Bus_t))
        ARR.append(arr)
        LEA.append(lea)
ARR=np.array(ARR)
LEA = np.array(LEA)

###计算每辆公交车每个仿真秒的位置
for p in range(ARR.shape[0]):#遍利每一辆公交车
    loc = bus_location(p)
    LOC.append(loc)
LOC = np.array(LOC)

####看那些公交车在仿真时间内开出来了，哪些没开出来####
OPENID=[]
CLOSEID=[]
for n in range(LOC.shape[0]):
    if LOC[n,-1] == S[-1]+500:
        OPENID.append(n)
    else:
        CLOSEID.append(n)

####定义行人的一些时间#####
def passenger():
    p_starttime = np.random.randint(T)#乘客产生时间
    walkspeed = 1.2#乘客步行速度
    startloc = np.random.random(1)*((S[-1]+S[-2])/2)#乘客产生地点
    startdiff = (abs(np.repeat(startloc,S_num)-S)).min()#乘客产生与上车站距离
```

```
    startst = np.argmin(abs(np.repeat(startloc,S_num)-S))#乘客上车车站
    walktimef = int(startdiff/walkspeed)#上车前步行时间
    arrive_st_t =   p_starttime + walktimef#乘客到达上车站时间
    stoploc = np.random.random(1)*(S[-1]-(S[startst+1]-500))+(S[startst+1]-500)#乘客消失地点
    stopdif = (abs(np.repeat(stoploc,S_num)-S)).min()#乘客下车站与乘客消失距离
    stopst = np.argmin(abs(np.repeat(stoploc,S_num)-S))#乘客下车车站
    walktimel=int(stopdif/walkspeed)#下车后步行时间
    psger = [p_starttime,arrive_st_t,startloc,startdiff,stoploc,stopdif,S[startst],S[stopst],walktimef,walktimel,startst,stopst]
    #[0:乘客产生时间,1:乘客到达上车站时间,2:乘客产生地点,3:乘客产生与上车站距离,
    #4:乘客消失地点,5:乘客下车站与乘客消失距离,6:乘客上车位置,7:乘客下车位置,8：上车前步行时间，9：下车后步行时间
    #10:起始站点,11:下车站点]
    return(psger)

PSGER=[]
for p in range(N):
    PSGER.append(passenger())
PSGER = np.array(PSGER)#返回行人的各种信息矩阵


######行人在仿真期间的位置######
PSXLOC=[]#行人在仿真期间的 X 横向位置
PSYLOC=[]#行人在仿真期间的 Y 纵向位置
for h in range(N):#便利每一个行人样本
    psxloc = (np.ones(T)*(-500)).reshape([1,T])#初始化 x 位置
    psyloc = (np.ones(T)*(30)).reshape([1,T])#初始化 y 位置
    #仿真开始-行人产生
        #在初始化位置
    #行人产生-行人到达站台
    if PSGER[h,1] >= T:#如果到站时间超过 T，跳出本次循环
        continue
    psxloc[0,int(PSGER[h,0]):int(PSGER[h,1])]=np.linspace(start=PSGER[h,2],stop=PSGER[h,6],num=int(PSGER[h,8]))
    #行人到达站台-行人上车（可能有等待时间，也可能没有）
    temp = np.repeat(PSGER[h,1],LEA.shape[0])-LEA[:,int(PSGER[h,10])]
    for u in range(temp.shape[0]):
        if temp[u]>0:
            temp[u]=-1000
    busid = np.argmax(temp)
    if ARR[busid,int(PSGER[h,10])] >= T:
        continue
    if (PSGER[h,1]-ARR[busid,int(PSGER[h,10])])<0:
        psyloc[0,int(ARR[busid,int(PSGER[h,10])]):int(ARR[busid,int(PSGER[h,11])])] = 20
    else:
        psyloc[0,int(PSGER[h,1]):int(ARR[busid,int(PSGER[h,11])])] = 20
    psxloc[0,int(PSGER[h,1]):int(LEA[busid,int(PSGER[h,10])])] = PSGER[h,6]
    #行人上车-行人下车到站台
    if ARR[busid,int(PSGER[h,11])]>=T:
        continue
    psxloc[0,int(LEA[busid,int(PSGER[h,10])]):int(ARR[busid,int(PSGER[h,11])])] = LOC[busid,
int(LEA[busid,int(PSGER[h,10])]):ARR[busid, int(PSGER[h,11])]]
    psyloc[0,int(LEA[busid,int(PSGER[h,10])]):int(ARR[busid,int(PSGER[h,11])])] = 20
    #行人下车到站台-行人到 D
    if ARR[busid,int(PSGER[h,11])]+PSGER[h,9]>=T:
        continue
    psxloc[0,int(ARR[busid,int(PSGER[h,11])]):int(ARR[busid,int(PSGER[h,11])]+PSGER[h,9])] =
np.linspace(PSGER[h,7],stop=PSGER[h,4],num=int(PSGER[h,9]))
    psyloc[0,int(ARR[busid,int(PSGER[h,11])]):int(ARR[busid,int(PSGER[h,11])]+PSGER[h,9])] = 10
    #行人到 D-仿真结束
    psxloc[0,int(ARR[busid,int(PSGER[h,11])]+PSGER[h,9]):] = 5500
    psyloc[0,int(ARR[busid,int(PSGER[h,11])]+PSGER[h,9]):] = 10
    PSXLOC.append(psxloc)
    PSYLOC.append(psyloc)
PSXLOC = np.array(PSXLOC)
PSYLOC = np.array(PSYLOC)
```

```
PSXLOC1 = PSXLOC.reshape([PSXLOC.shape[0],T])
PSYLOC1 = PSYLOC.reshape([PSYLOC.shape[0],T])

#指标计算#
####乘客通勤时间计算####
####公交站点等待人数计算####
wait_num = np.zeros((S_num, T))#行为 6 个站，列为仿真秒内人数
for ii in range(T):
    wait_num[:,ii]=np.array([np.count_nonzero(PSXLOC1[:,ii]==S[0]),
                             np.count_nonzero(PSXLOC1[:,ii]==S[1]),
                             np.count_nonzero(PSXLOC1[:,ii]==S[2]),
                             np.count_nonzero(PSXLOC1[:,ii]==S[3]),
                             np.count_nonzero(PSXLOC1[:,ii]==S[4]),
                             np.count_nonzero(PSXLOC1[:,ii]==S[5])])
for ii in range(S_num):#对于每个站点
    arrtime = ARR[:,ii]#所有车到 ii 站的时间
    leatime = LEA[:,ii]#所有车离开 ii 站的时间
    for p,q in zip(arrtime,leatime):
        wait_num[ii,p:q] = 0
####公交车实时载客量计算####
load_num = np.zeros((len(OPENID), T))
for i in OPENID:#d 对于每个公交车
    arrivetime = ARR[i,:]
    leavetime = LEA[i,:]
    for p,q in zip(arrivetime,leavetime):#在停站期间的变动
        for t in range(p,q):
            load_num[i,t] = np.count_nonzero(PSXLOC1[:,t]==LOC[i,t])
    for p,q in zip(leavetime[:-1],arrivetime[1:]):#在运行期间
        for t in range(p,q):
            load_num[i,t] =load_num[i,t-1]

###固定内容####
fig=plt.figure(figsize=(30,15))
ax = fig.add_subplot(211)
ax.set_xlim(-1000,6000)
ax.set_ylim(0,40)
ax.grid(True)
ax.set_title('Bus operation simulation',fontsize=16,fontweight='medium')
ax.set_xlabel('station_location')
ax.set_yticks([10,20,30])
ax.set_yticklabels([r'$PS\ route$', r'$bus\ route$', r'$PS\ route$'])
ax.plot([S[0], S[-1],], [20, 20,], 'k--', linewidth=2,alpha=0.7)#串联始终的一条线
ax.text(S[0]-150, 17, r'$S\ 1$',fontdict={'size': 10, 'color': 'k'})
ax.text(S[1]-150, 17, r'$S\ 2$',fontdict={'size': 10, 'color': 'k'})
ax.text(S[2]-100, 17, r'$S\ 3$',fontdict={'size': 10, 'color': 'k'})
ax.text(S[3]-150, 17, r'$S\ 4$',fontdict={'size': 10, 'color': 'k'})
ax.text(S[4]-150, 17, r'$S\ 5$',fontdict={'size': 10, 'color': 'k'})
ax.text(S[5]-150, 17, r'$S\ 6$',fontdict={'size': 10, 'color': 'k'})
ax.text(S[0]-700, 17, r'$Bus$',fontdict={'size': 10, 'color': 'dimgrey'})
ax.text(S[-1]+550, 17, r'$Bus$',fontdict={'size': 10, 'color': 'dimgrey'})
ax.text(S[0]-600, 27, r'$PS$',fontdict={'size': 10, 'color': 'dimgrey'})
ax.text(S[-1]+550, 7, r'$PS$',fontdict={'size': 10, 'color': 'dimgrey'})
ax2 = fig.add_subplot(223)
ax2.grid(True)
ax2.set_title('Queue length in each station',fontsize=16,fontweight='medium')
ax2.set_xlabel('Station_id')
ax2.set_ylabel('Queue length')
ax2.set_xticklabels(['','S1','S2','S3','S4','S5','S6'])
ax2.set_ylim(-5,35)
ax2.set_yticks(np.arange(-5,35,5))
ax3 = fig.add_subplot(224)
ax3.grid(True)
ax3.set_title('Bus_load in each bus',fontsize=16,fontweight='medium')
ax3.set_xlabel('Bus_id')
ax3.set_ylabel('Bus_load')
```

```
ax3.set_xticks(np.arange(0,len(OPENID)))
ax3.set_ylim(-5,35)
ax3.set_yticks(np.arange(-5,35,5))
###动画绘制#####
ims=[]

for i in range(T):
    l1 = ax.scatter(x=PSXLOC1[:,i],y=PSYLOC1[:,i], c='darkblue',marker='x',s=20,animated=True)#乘客散点
    l2 = ax.scatter(x=LOC[:,i],y=np.repeat(20,LOC.shape[0]), c='r',marker='o', s=80,animated=True)#车辆散点
    l3 = ax.text(S[-1]-700, 35, r'$Time:\ %ss$'%i,fontdict={'size': 20, 'color': 'k'},animated=True)
    l4 = ax2.scatter(np.arange(6),wait_num[:,i], c='cornflowerblue',marker='x',s=200,animated=True)#乘客散点
    l5 = ax3.scatter(np.arange(len(OPENID)),load_num[:,i], c='salmon',marker='o', s=70,animated=True)
    im=[l2,l1,l3,l4,l5]
    ims.append(im)
ani = animation.ArtistAnimation(fig,ims,interval=100)

ani.save('3axis.mp4')
```