VMS 可变信息板对交通系统的影响分析

作者：刘欣豪 2020112921

摘要：VMS 可变信息板可为出行者提供出行参考，其优秀的信息提供策略可以有效减少拥堵、提高交通效率。本文通过模拟仿真对比提供预测信息与提供瞬时信息的系统在不同敏感度与一切强制交通事故下的表现，得到结论 1、分析提供预测信息的系统优于提供瞬时交通信息的系统，2、敏感度越高，系统效果不一定越好，3、发生交通事件时，提供预测信息的 VMS 系统能有效降低延误。同时在模型在实际应用上应考虑交通信息获取与现实中可能存在的多路径的问题。

关键字：VMS 可变信息板，激励-反应，预测信息， 延误

一、文献概述

1.1 先进的出行者信息服务系统

先进的出行者信息服务系统主要是对交通出行者提供及时的信息服务。在出行前，通过办公室或家庭的计算机终端、咨询电话、咨询广播系统等，向出行者提供当前的交通和道路状况以及服务信息，帮助出行者选择出行方式、出行时间和出行路线；在出行途中，通过车载信息单元或路边动态信息显示板，向出行者提供道路条件、交通状况、车辆运行情况、交通服务等实时信息，通过路径诱导系统对车辆定位和导航，使汽车始终行驶在最佳路线上，使出行者以最佳的出行方式和路线到达目的地。

ATIS 可以通过车载设施、可变标志、交通信息广播、移动电话等，向驾驶员提供互动信息，让他们始终行驶在最短路线上。ATIS 提供的信息可以分为三类：出行前信息、途中信息、目的地信息。

1.2 VMS 可变信息板

VMS 可变信息板为一种动态的公路交通标志。当前方道路因天气、自然灾害、交通事故等原因而导致行车环境发生变化时，VMS 可实时为驾驶员提供信息，供驾驶员提高警惕和选择路径使用。

在国内研究中，辛飞飞等使用 RP 调查获取驾驶员行为数据,探讨可变信息板此类交

通诱导信息对驾驶员行为的影响[1]，张敖木翰等研究突了发事故下临时性的车辆禁行设计与可变信息板选址组合优化问题，得到组合控制措施能够有效地降低突发事故所导致的交通拥堵,提高交通网络的系统性能[2]，干宏程等运用混合选择模型研究了图状路径信息板对驾驶员路径选择的影响[3]。在国外研究中，Ilse M. Harms, Chris Dijksterhuis 等研究可变信息标志（VMS）是否可用于显示与交通无关的信息，提出交通无关消息可能不会干扰流量管理[4]，AlKheder, S.等研究了不同 VMS 信息下影响驾驶员决定切换路线的因素。

本文利用激励-反应分析模拟选择某一路径的比例，仿真运行得到 VMS 可变信息板对交通系统的影响结果，并通过平均延误、排队长度等对不同预测模型、不同交通状态、不同敏感性情况下模型的表现进行评价。

二、模型描述与实现

2.1 符号说明

表 1：符号说明

序号	符号	定义
1	$\gamma_j(t)$	选择 j 路径的出行者比例
2	p^{nc}	不受限制的出行者比例
3	$\gamma_j^{captives}(t)$	受到限制的出行者选择 j 路径的比例
4	$\gamma_j^{nc}(t)$	不受限制的出行者选择 j 路径的比例
5	β_0	不受限制出行者的习惯性选择参数
6	β	受限制出行者对交通信息的敏感性
7	$T_j^{inst}(t)$	提供信息时推测路径 j 的瞬时行程时间
8	$T_j^{\exp}(t)$	提供信息时推测路径 j 的预测行程时间
9	$C_j(t)$	路径 j 的容量
10	$P_j(t)$	路径 j 的外部交通量
11	$T_j^{free}(t)$	路径 j 的自由流时间
12	$N_J(t)$	路径 j 于 t 时刻的排队车辆数

2.2 模型建立

关于可变信息标志对驾驶员选择路径的影响，考虑到部分出行者可能出行路径受到限制，一般出行路径不受限制的驾驶者会根据驶入两段路之前对交通信息（排队时间）进行判断，从而选择路径；而出行路径受到限制（必须选择某条路）将不对该激励（交通信息）做出反应。最终选择路径的比例如下：

$$\gamma_j(t) = [1 - p^{nc}]\gamma_j^{captives}(t) + p^{nc}\gamma_j^{nc}(t) \quad (1)$$

受到限制的出行者将不受 VMS 可变信息板的影响，p^{nc}、$\gamma_j^{captives}(t)$ 与道路交通信息无关，为常量。对于系统提供交通信息主要影响 $\gamma_j^{nc}(t)$ 的取值，利用激励-反应模型。即选择某一路径的比例为不受限制出行者的习惯性选择比例加上出行者对交通信息的敏感性与提供两道路可能通行时间差的乘积，系统可提供预测信息与瞬时行程时间信息

① 提供预测行程时间

$$\gamma_j^{nc}(t) = \beta_0 + \beta[T_k^{\exp}(t) - T_j^{\exp}(t)] \quad (2)$$

② 提供瞬时行程时间

$$\gamma_j^{nc}(t) = \beta_0 + \beta[T_k^{inst}(t) - T_j^{inst}(t)] \quad (3)$$

瞬时行程时间为自由流时间加上瞬时的排队车辆数比道路的通行能力，而预测行程时间为自由流时间加上预测的排队车辆数比道路的通行能力。

$$T_j^{inst}(t+\Delta t) = T_j^{free} + \frac{N_j(t)}{C_j(t)} \quad (4)$$

$$T_j^{\exp}(t+\Delta t) = T_j^{free} + \frac{N_j(t+T_j^{free})}{C_j(t+T_j^{free})} \quad (5)$$

排队车辆数为上一个时间的排队车辆加上时间间隔与路上交通量加外部交通量减去道路通行能力的乘积：

$$N_j(t+\Delta t) = N_j(t) + \Delta t[Q_j(t-T_j^{free}) + P_j(t) - C_j(t)] \quad (6)$$

三、模型求解

3.1 场景搭建

现从 A 地到 B 地有两条路径可供选择，起点处装有 VMS 提供信息给司机，对于路

径 1 与路径 2 分别有 P1 与 P2 外部交通量。如图 1：

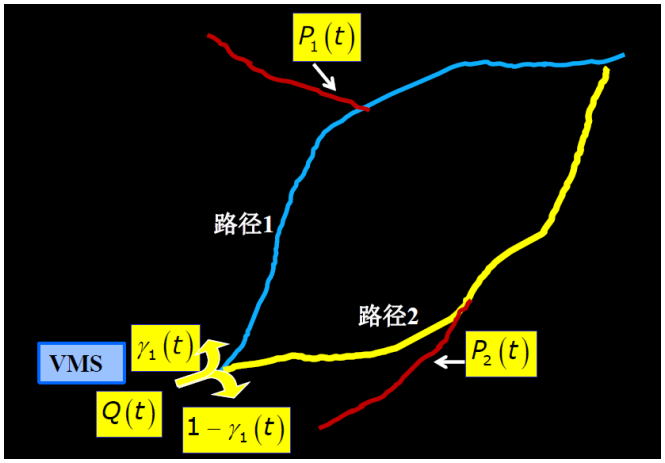

图 1：路径选择图

对于两条路径，路径基本信息如表 2：

TT1：路径 1 自由流时间	18	mins
TT2：路径 2 自由流时间	16	mins
L1：路径 1 长度	30	km
L2：路径 2 长度	27	km
车辆长度	5	m
缺省路径 1 选择比率	58	%
实际事件持续时间长度	20	mins

表 2：路径基本信息

对于两条路径的交通流量信息如表 3：

基础交通流量 (i)	2502
外部交通需求量 (p1)	2859
外部交通需求量 (p2)	3849
路径 1 通行能力	6116
路径 2 通行能力	6116

表 3：交通流量信息

同时，对于该场景，假设对应于正常交通状况，路径 2 可能发生交通事故，此时路径 2 的通行能力降低为原通行能力的 10%，为 616.6pcu/h。

3.2 模型求解

该研究主要针对选择路径不受限制的出行者，同时应模拟不同系统（反馈与现实系统）下对交通的影响。

现依靠 Python 编程构建此迭代过程。建立一 simulate 类，初始化的仿真信息为 DataFrame 空表，simulate 类其中包含敏感度、系统信息参数、是否发生事件（详情见附件一 simulate 类下__init__函数）诸如此类的信息。下面，将分别对各个过程进行说明。

（1）初始化建立

初始化建立生成一空表，需要输入的为仿真的时间参数，同时自动对基础参数进行赋-1 值处理。

（2）仿真运行

仿真运行需要导入是否发生事件、系统类别的参数（默认为不发生事件与非反馈系统），首先将基本交通需求量参数导入仿真表，从仿真时间 0 开始进行仿真，直至运行至初始化设定的最后一仿真秒。

（3）运行评价

运行评价将对方针的结果进行平均延误、总延误有、系统性能等可评价指标的运算，并赋值给相应的类属性

（4）仿真导出

将导出’output.xlsx’类的表格，其内容为仿真 table。

（5）画图

将针对不同的结果利用 matplotlib.pyplot 模块进行画图，包括关于时间的延误、选择率等信息。

具体代码如附件 1。

四、结果分析

4.1 系统提供瞬时交通信息与预测交通信息

当不提供 VMS 时，延误与路径选择率如图 3，可见路径 1 的选择率稳定在 0.58，无变化。

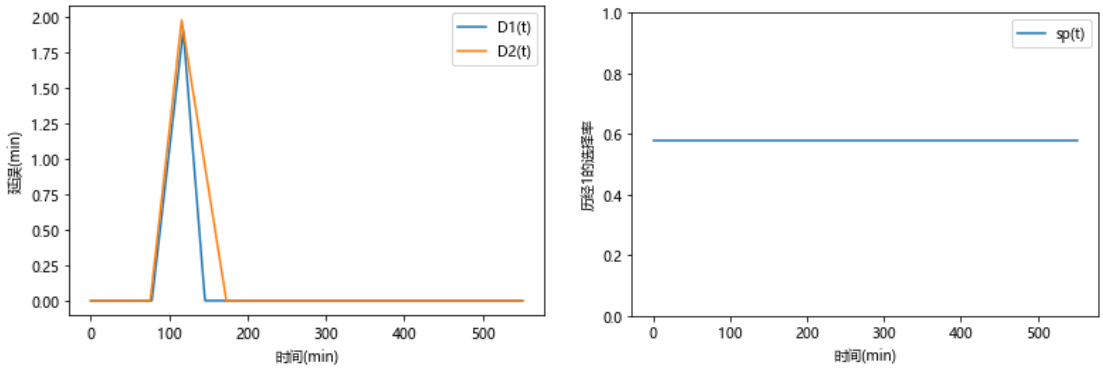

图 2：不提供 VMS 时延误（左）与路径选择率（右）

对比图 3 与图 4，当敏感度取值 0.05 时，路径选择率在 100s 出现延误时出现波动，延误，提供瞬时信息的系统与提供预测信息的系统延误都有所降低。

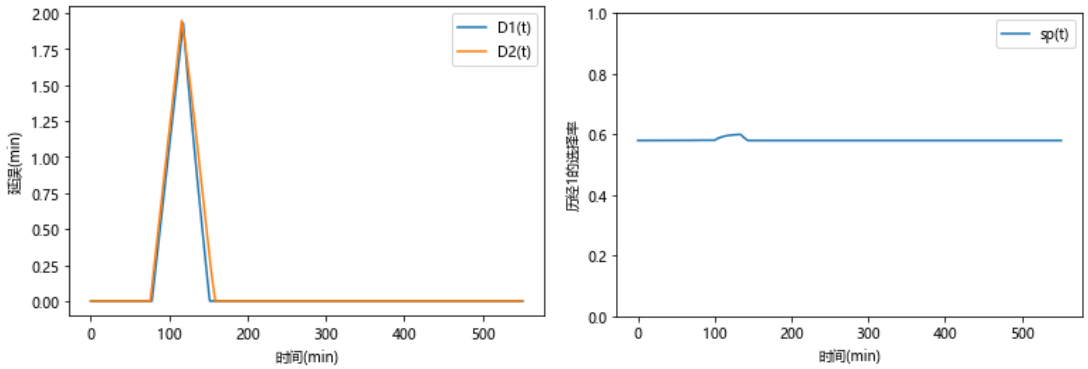

图 3：β 为 0.05 提供预测信息时延误（左）与路径选择率（右）

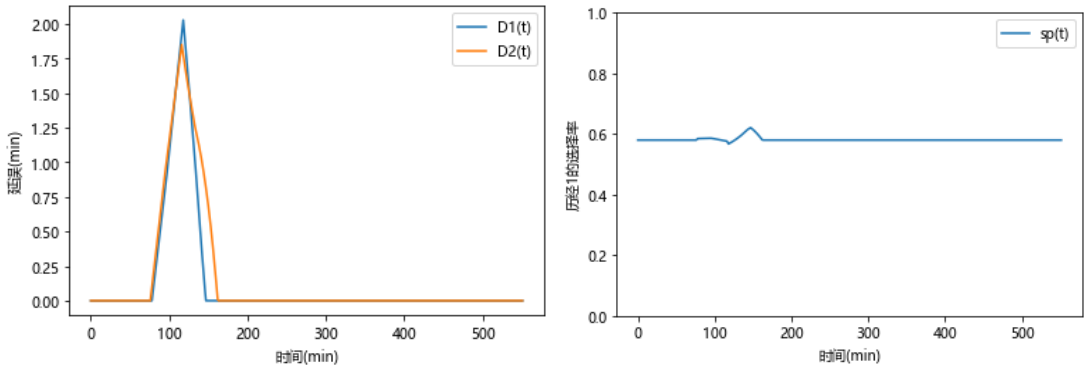

图 4：β 为 0.05 提供瞬时信息时延误（左）与路径选择率（右）

对比图 5 与图 6，当敏感度取值 0.05 时，提供预测信息的系统，路径选择率在 100s 出现延误时发生波动，路径 1 的选择率有所增加，路径 1 与路径 2 的延误在事件发生两趋于一致；提供瞬时信息的系统路径选择率产生极大的波动，路径 1 与路径 2 延误大大增加，系统极不稳定。

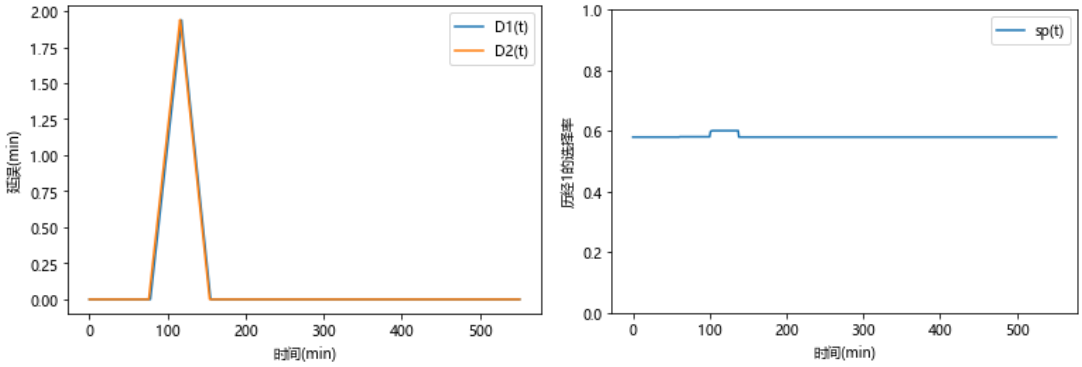

图 5：β 为 0.5 提供预测信息时延误（左）与路径选择率（右）

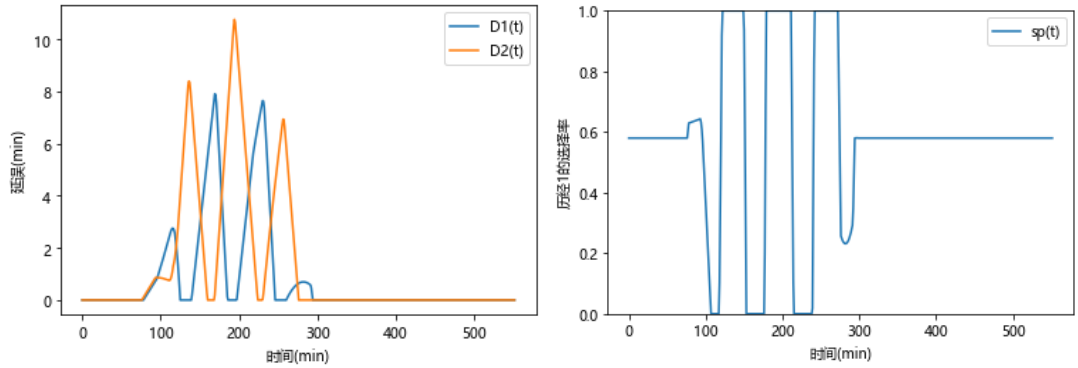

图 6：β 为 0.5 提供瞬时信息时延误（左）与路径选择率（右）

4.2 敏感性变化

在正常车流量状态（不发生交通事件）下，设敏感性 β 从 0 间隔 0.05 变化到 1，瞬时交通信息与预测交通信息的平均延误（图 7）、总延误可变性（图 8）如下：

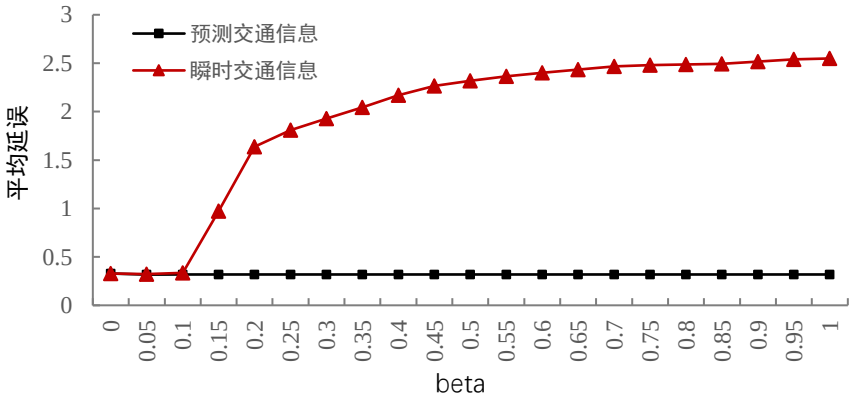

图 7：β 变化下两种信息的平均延误

随着敏感度 β 的增加，预测交通信息的平均延误基本稳定，有小幅降低的趋势；瞬时交通信息的平均延误在敏感度小于 0.1 时平均延误降低，但大于 0.1 时延误大幅增长。

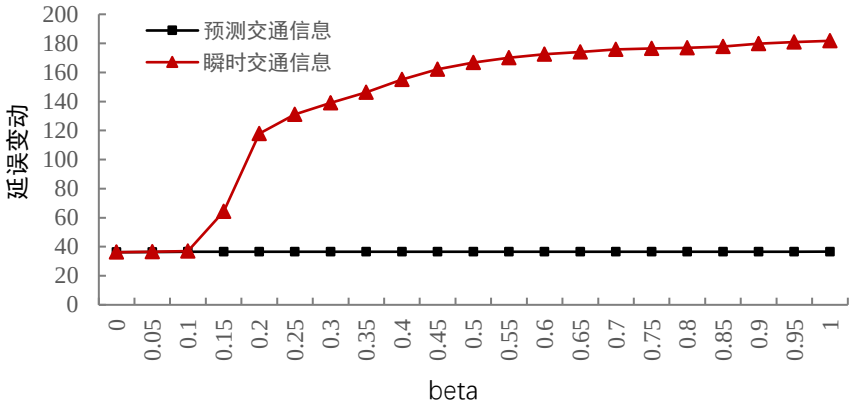

图 8：β 变化下两种信息的总延误可变性

随着敏感度 β 的增加，相较于不提供 VMS 服务，预测交通信息的总延误可变性基本稳定；瞬时交通信息总延误可变性在敏感度小于 0.1 时基本不变，但大于 0.1 时延误可变性大幅增长。

4.3 正常交通状况与交通事件发生情况下

交通事件发生即在 150s 到 169s 时道路 2 的通行能力下降到原值的 0.1，其对交通系统延误的影响如图 9。

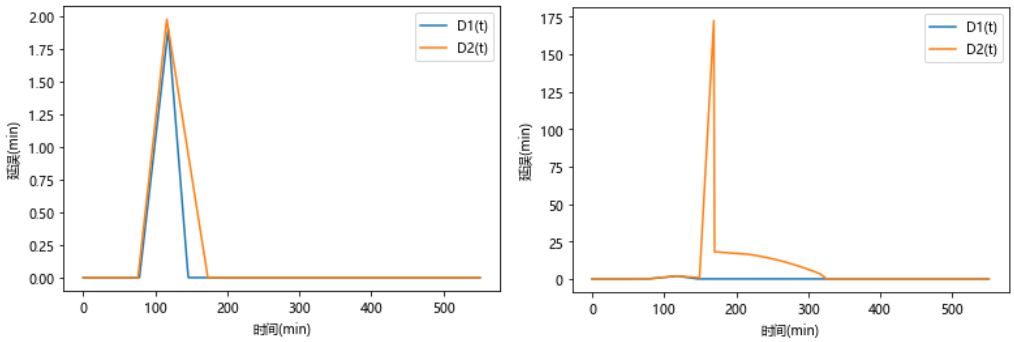

图 9：不提供 VMS 时正常交通状况延误（左）与交通事件发生延误（右）

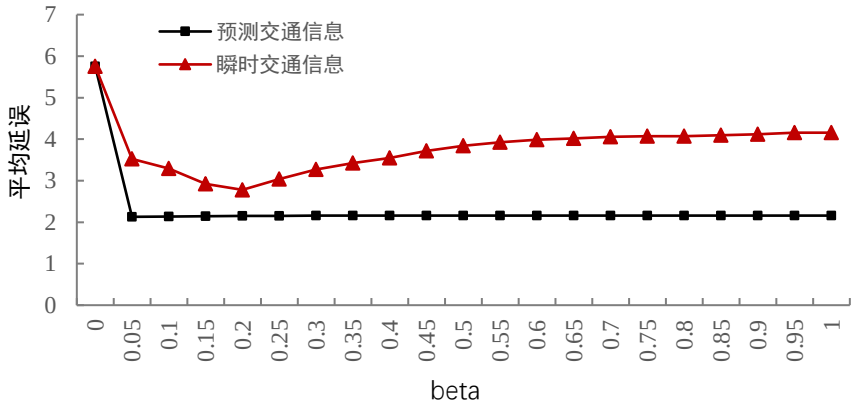

图 10：发生事件时 β 变化下两种信息的平均延误

当发生事件时，应用提供瞬时信息的系统与提供预测信息的系统，在 β 逐渐增大的情况下平均延误得到显著降低，且提供预测交通的系统效果更为明显。考虑到当 β 较大时提供瞬时信息的系统不稳定，下面只具体分析瞬时信息系统。

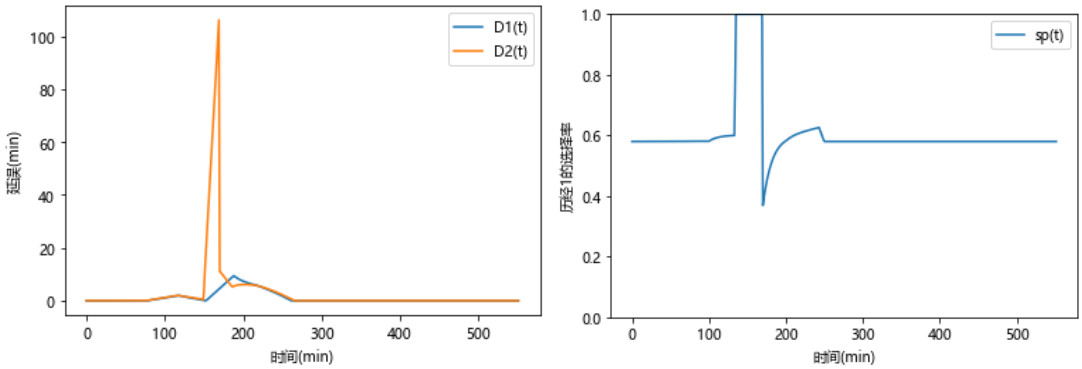

图 11：提供瞬时信息时 β 为 0.05 交通事件延误（左）与路径选择率（右）

如图 11，当时 β 较小时路径 1 的延误大幅降低，路径 2 的延误小幅增高，总体延误降低，系统起到了良好的调节作用。

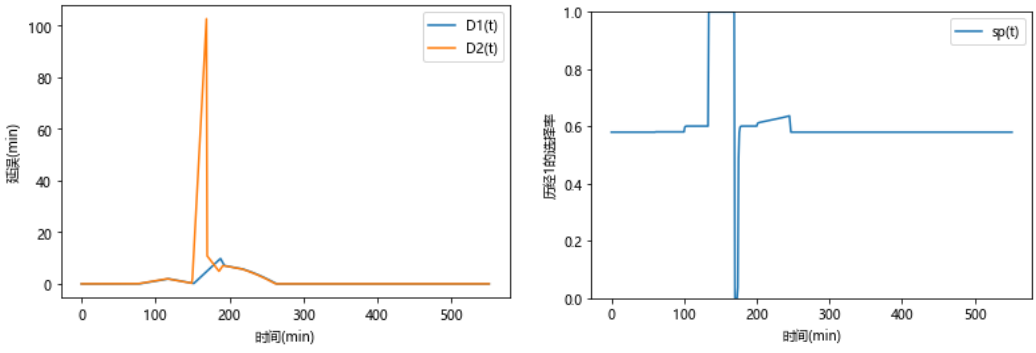

图 12：提供瞬时信息时 β 为 0. 5 正常交通状况延误（左）与路径选择率（右）

如图 12，当β为 0.5 时，路径 1 的选择出现一次大幅波动，但总体延误仍然大幅降低，调节能力增强但稳定性减弱。

五、结论和存在问题

5.1 结论

1、提供预测信息的系统优于提供瞬时交通信息的系统

无论事件是否发生，提供预测信息的系统的降低延误的作用与系统稳定性都强于提供瞬时交通信息的系统，尤其是当敏感度较大时，提供瞬时交通信息的系统会出现振荡现象，延误升值增加。故提供预测信息的系统明显优于提供瞬时交通信息的系统。

2、敏感度越高，系统效果不一定越好

敏感度较低时，无论是提供预测信息的系统还是提供瞬时信息的系统都能有效降低延误，但当敏感度高至一定程度（如β=0.5 时），提供瞬时信息的系统将会出现振荡现象，而提供预测信息的系统稳定性也有小幅降低，选择路径 1 的概率会发生一次大的跳跃。

3、发生交通事件时，提供预测信息的 VMS 系统能有效降低延误

在发生交通事件时，如果存在提供预测信息的 VMS 系统，其延误降低效果明显，能够提高交通效率。

5.2 存在问题

1、现实应用中的交通信息获取

在现实中，交通需求信息是动态变化的且难以准确监测的，如何准确获得这些道路交通需求、外部交通需求用于系统做预测计算，从而发挥系统的价值有待考虑。

2、多路径选择问题

当存在多路径（3 个以上），该模型不适用，而现实中某些起止点一般可选择路径甚至可能更多。

参考文献

[1]辛飞飞,韦龙雨.交通诱导信息对驾驶员路径选择行为影响调查分析[J].交通信息与安全,2013,31(03):64-68.

[2]张敖木翰,高自友.突发事故下交通拥堵控制策略设计[J].系统工程理论与实践,2013,33(05):1307-1317.

[3]干宏程,孙亦凡.基于混合选择模型的图状路径信息板影响分析[J].上海理工大学学报,2019,41(01):58-63.DOI:10.13255/j.cnki.jusst.2019.01.009.

[4] Ilse M. Harms, Chris Dijksterhuis, Bart Jelijs, Dick de Waard, Karel A. Brookhuis, Don't shoot the messenger: Traffic-irrelevant messages on variable message signs (VMSs) might not interfere with traffic management, Transportation Research Part F: Traffic Psychology and Behaviour, 65,564-575(2019), ISSN 1369-8478, https://doi.org/10.1016/j.trf.2018.09.011.

[5] AlKheder, S., AlRukaibi, F. & Aiash, A. Drivers' response to variable message signs (VMS) in Kuwait. Cogn Tech Work 21, 457–471 (2019). https://doi-org-s.era.lib.swjtu.edu.cn:443/10.1007/s10111-019-00538-7

附件：

附件 1：simulate.py

说明：建立基本仿真类 ，用于仿真模拟

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt


plt.rcParams['font.sans-serif'] = ['Microsoft YaHei']


#以下交通量单位为pcu/h
CODE = 278
basic_flow = CODE/200 * 1800
demand_p1 = CODE/175*1800
demand_p2 = CODE/130*1800
capacity_p1 = CODE/100*2200
capacity_p2 = CODE/100*2200


L_p1 = 30 #路径1长度，单位为km
L_p2 = 27 #路径2长度，单位为km
len_car = 5 #车辆长度，单位为m
Tfree_p1 = 18 #路径1自由流时间，单位为min
Tfree_p2 = 16 #路径2自由流时间，单位为min


class simulatation(object):
    def __init__(self, time):
        self.time = time  # 仿真时间
        # 以下交通量单位为pcu/h
        self.basic_flow = basic_flow
        self.demand_p1 = demand_p1
```

```
        self.demand_p2 = demand_p2
        self.capacity_p1 = capacity_p1
        self.capacity_p2 = capacity_p2

        self.L_p1 = L_p1  # 路径 1 长度，单位为 km
        self.L_p2 = L_p2  # 路径 2 长度，单位为 km
        self.len_car = len_car  # 车辆长度，单位为 m
        self.Tfree_p1 = Tfree_p1  # 路径 1 自由流时间，单位为 min
        self.Tfree_p2 = Tfree_p2  # 路径 2 自由流时间，单位为 min

        self.length = self.time + max(self.Tfree_p1, self.Tfree_p2) + 1

        self.table = -1

        self.beta = 0  # 敏感度
        self.is_feedback = 0  # 默认不是反馈控制
        self.is_event = 0  # 默认不发生事件
        self.pre_time = 20  # 默认事件持续时间为 20min

        # 结果参数
        self.delay = -1  # 延误
        self.avg_delay = -1  # 平均延误
        self.queue_length = -1  # 总排队长度
        self.avg_performance = -1  # 平均性能
        self.delay_variability_p1 = -1  # 路径一延误可变性
        self.delay_variability_p2 = -1  # 路径二延误可变性
        return

    # 生成初始条件
```

```
    def generate_traffic_demand(self):
        table = pd.DataFrame([[i for i in range(self.length)]] + [[0] *
(self.length)] * 16,
                             index=['Time (min)', 'i(t)', 'p1(t)', 'p2(t)',
                                    'i1(t)', 'i2(t)', 'C1(t)', 'C2(t)',
'r1(t)', 'r2(t)',
                                    'L1(t)', 'L2(t)', 'D1(t)', 'D2(t)',
'sp(t)', 'Dtot', 'Ptot'])
        self.table = table.T
        s = []
        for i in range(self.length):
            if i < 60:
                j = (self.basic_flow)
            elif i < 100:
                j = (self.capacity_p1)
            elif i < 200:
                j = (0.8 * self.capacity_p1)
            elif i < 300:
                j = (self.basic_flow + (0.7 * self.capacity_p1 -
self.basic_flow)
                     * (0.05 * self.capacity_p1 - i) / 100)
            else:
                j = (0.1 * self.basic_flow)
            s.append(j)
        self.table['i(t)'] = s
        self.table['p1(t)'] = self.demand_p1
        self.table['p2(t)'] = self.demand_p2
        self.table['C1(t)'] = self.capacity_p1
        self.table['C2(t)'] = self.capacity_p2
```

```
        return


    # 开始仿真
    def start(self, beta=0, is_event=0, is_feedback=1):
        self.beta = beta
        self.is_event = is_event
        self.is_ = is_feedback


        self.generate_traffic_demand()


        if is_event:
            if self.time > 170:
                self.table.loc[150:169, 'C2(t)'] = 0.1 *
self.table.loc[150:169, 'C2(t)']
            else:
                self.is_event = 0
                print('conflict between time and is_event')
                return -1


        table = self.table
        for i in range(self.Tfree_p1):
            table.loc[i, 'r1(t)'] = 0
            table.loc[i, 'L1(t)'] = 0.005 * table.loc[i, 'r1(t)']
            table.loc[i, 'D1(t)'] = 60 * table.loc[i, 'r1(t)'] /
table.loc[i, 'C1(t)']
        for i in range(self.Tfree_p2):
            table.loc[i, 'r2(t)'] = 0
            table.loc[i, 'L2(t)'] = 0.005 * table.loc[i, 'r2(t)']
            table.loc[i, 'D2(t)'] = 60 * table.loc[i, 'r2(t)'] /
```

```
table.loc[i, 'C2(t)']


        for i in range(self.time + 1):
            if is_feedback:
                table.loc[i, 'sp(t)'] = min(1,
                                            max(0, 0.58 - self.beta *
(table.loc[i, 'D1(t)'] - table.loc[i, 'D2(t)'])))


            else:
                if i >= 150 and i <= 169 and self.is_event:
                    table.loc[i, 'sp(t)'] = min(1, max(0, 0.58 - self.beta *
(
                                table.loc[i + self.Tfree_p1, 'D1(t)'] -
table.loc[
                            i + self.Tfree_p2, 'D2(t)'] - self.pre_time)))
                else:
                    table.loc[i, 'sp(t)'] = min(1, max(0, 0.58 - self.beta *
(
                                table.loc[i + self.Tfree_p1, 'D1(t)'] -
table.loc[i + self.Tfree_p2, 'D2(t)'])))


            table.loc[i, 'i1(t)'] = table.loc[i, 'i(t)'] * table.loc[i,
'sp(t)']
            table.loc[i, 'i2(t)'] = table.loc[i, 'i(t)'] - table.loc[i,
'i1(t)']


            table.loc[i + self.Tfree_p1 + 1, 'r1(t)'] = max(0, table.loc[i
+ self.Tfree_p1, 'r1(t)'] + (
                        table.loc[i, 'i1(t)'] + table.loc[i + self.Tfree_p1,
```

```
'p1(t)'] - table.loc[
                i + self.Tfree_p1, 'C1(t)']) / 60)
            table.loc[i + self.Tfree_p2 + 1, 'r2(t)'] = max(0, table.loc[i
+ self.Tfree_p2, 'r2(t)'] + (
                    table.loc[i, 'i2(t)'] + table.loc[i + self.Tfree_p2,
'p2(t)'] - table.loc[
                i + self.Tfree_p2, 'C2(t)']) / 60)

            table.loc[i + self.Tfree_p1 + 1, 'L1(t)'] = 0.005 * table.loc[i
+ self.Tfree_p1 + 1, 'r1(t)']
            table.loc[i + self.Tfree_p2 + 1, 'L2(t)'] = 0.005 * table.loc[i
+ self.Tfree_p2 + 1, 'r2(t)']

            table.loc[i + self.Tfree_p1 + 1, 'D1(t)'] = 60 * table.loc[i +
self.Tfree_p1 + 1, 'r1(t)'] / table.loc[
                i + self.Tfree_p1 + 1, 'C1(t)']
            table.loc[i + self.Tfree_p2 + 1, 'D2(t)'] = 60 * table.loc[i +
self.Tfree_p2 + 1, 'r2(t)'] / table.loc[
                i + self.Tfree_p2 + 1, 'C2(t)']

            table.loc[i, 'Dtot'] = table.loc[i + self.Tfree_p1, 'D1(t)'] *
table.loc[i, 'i1(t)'] + table.loc[
                i + self.Tfree_p2, 'D2(t)'] * table.loc[i, 'i2(t)']
            table.loc[i, 'Ptot'] = self.L_p1 * table.loc[i, 'i1(t)'] +
self.L_p2 * table.loc[i, 'i2(t)']

        self.table = table.loc[:self.time, :]
        #           self.table = table
        return
```

```
        # 进行评估

    def evaluation(self):
        self.delay = sum(self.table['D1(t)']) + sum(self.table['D2(t)'])  # 延误
        self.avg_delay = sum(self.table['Dtot']) / sum(self.table['i(t)'])  # 平均延误
        self.queue_length = (sum(self.table['r1(t)']) + sum(self.table['r2(t)'])) / (self.time + 1)  # 总排队长度
        self.avg_performance = sum(self.table['Ptot']) / sum(self.table['i(t)'])  # 平均性能
        self.delay_variability_p1 = self.table['D1(t)'].std(ddof=1)  # 路径一延误可变性
        self.delay_variability_p2 = self.table['D2(t)'].std(ddof=1)  # 路径二延误可变性
        return

    # 打印评估结果
    def print_eva(self):
        print('totle time:', self.time)
        print('delay:', self.delay)
        print('avg_delay:', self.avg_delay)
        print('queue_length:', self.queue_length)
        print('avg_performance:', self.avg_performance)
        print('Delay variability (Path 1)', self.delay_variability_p1)
        print('Delay variability (Path 2)', self.delay_variability_p2)

    # 导出表
```

```
    def export_table(self):
        self.table.to_excel('test1.xlsx', index=False)
        return


    def plot_r(self):
        plt.plot(self.table['Time (min)'], self.table['r1(t)'],
label='r1(t)')
        plt.plot(self.table['Time (min)'], self.table['r2(t)'],
label='r2(t)')
        plt.xlabel('时间(min)')
        plt.ylabel('排队长度')


        plt.legend()
        plt.show()
        return


    def plot_D(self):
        plt.plot(self.table['Time (min)'], self.table['D1(t)'],
label='D1(t)')
        plt.plot(self.table['Time (min)'], self.table['D2(t)'],
label='D2(t)')
        plt.xlabel('时间(min)')
        plt.ylabel('延误(min)')


        plt.legend()
        plt.show()
        return


    def plot_i(self):
```

```
        plt.plot(self.table['Time (min)'], self.table['i1(t)'],
label='i1(t)')
        plt.plot(self.table['Time (min)'], self.table['i2(t)'],
label='i2(t)')
        plt.plot(self.table['Time (min)'], self.table['i(t)'],
label='i(t)')
        plt.xlabel('时间(min)')
        plt.ylabel('交通需求量(veh/h)')


        plt.legend()
        plt.show()
        return


    def plot_sp(self):
        plt.plot(self.table['Time (min)'], self.table['sp(t)'],
label='sp(t)')
        plt.xlabel('时间(min)')
        plt.ylabel('路径 1 的选择率')
        plt.ylim(-0.05, 1.05)


        plt.legend()
        plt.show()
        return


    def plot(self):
        self.plot_r()
        self.plot_D()
        self.plot_i()
        self.plot_sp()
```

```
    def export_plot(self):
        return
```