武漢理工大學

数学建模暑期培训论文

第 3 题

基于帝国竞争算法求解公交车弹性发车间隔优化模型

第 1 组

姓名	方向
阮滨（组长）	编程
王宇	建模
王家柯	写作

2020 年 8 月 15 日

摘要

公交车作为民众出行的主要公共交通之一，其服务质量的提高具有重大意义。本文首先基于泊松流，得到了**单起讫点公交客流量模型**，进而结合阈值得到了判断是否为高峰期的标准。之后通过对公交服务过程中关键状态指标进行模拟，得到了**弹性发车间隔优化模型**，并运用**帝国竞争算法**进行求解，获得理论最优发车方案。接下来，本文采用了**小波神经网络**对客流量进行预测。最后，本文将预测客流量对应的发车间隔方案代入到实际客流量数据中，分析了方案的有效性。

针对问题一，需要确定客流量的平峰和高峰的定义。本文首先获取了深圳市 2014 年 6 月 9 日的深圳通刷卡数据，在**时间离散化**后根据一定比例将其转化为实际登车的人数。进而**基于泊松流建立了单起讫点公交客流量模型**，并将实际登车的人数作为系统过程的数据，对实际客流量进行了推导。最后根据公交车的容量确定了阈值，得到了“高峰”的判断标准，并以该日内某条公交线路为例，判断其高峰期为 **6:40 至 9:20**，基本符合常识。

针对问题二，需要给出客流量在高峰期与平峰期之间转化的方案。首先基于公交车的上下车人数、承载人数以及站台处排队的人数等要素对公交车线路的运行状态进行了描述，并结合网格法的基本思想引入了**公交车状态网络**的概念，结合边界条件便可对任意一个节点处公交车对应的状态进行推导。接下来，本文将乘客等待时间与公交车运营成本作为目标函数，基于最大净运营成本、公交车容量等构建了约束条件，**建立起以发车间隔作为决策变量的优化模型**，并通过**帝国竞争算法**进行求解。最终得到最优方案下，公交车的总运营成本为 **43729 元**。

针对问题三，需要对公交客流量进行预测。考虑到客流量在特定时段下的增加与减少，本文采用了**小波神经网络**作为客流量的预测方法。选取深圳市 2014 年 6 月 9 日的数据进行预测，预测结果的**均方差为 3.84**。

针对问题四，需要根据实际运行结果对结果进行验证。本文模拟了实际发车间隔安排模式，即根据预测结果进行安排，并通过实际客流量对运营成本进行计算。通过计算得到**基于预测的实际客流运营成本为 51690 元，比实际客流量下理论最低运营成本高 15.4%**。

本文的优点包括：1. 客流量衡量指标是根据理论系统模型计算得到的，故可以排除数据采集中由于不合理的实际发车间隔导致高峰期延长的偏差。2. 构建了公交车状态网络，使得每趟公交车运行过程中的状态要素都能够通过迭代运算计算得到，便于后续分析调整。

关键词： 帝国竞争算法　小波神经网络　公交车状态网络　泊松流

目录

1 问题重述

1.1 问题背景

随着城市化进程逐步发展和经济的日益增长，城市的人均拥有机动车量越来越高，这就导致了交通日益拥堵和环境的恶化。因此提倡市民出行使用公交车是解决问题的途径之一，提高公交车的服务水平是吸引市民采取公共交通的关键，因此需要对公交系统运营效率进行优化，减少市民等待公交车的时间和高峰期的拥挤程度。本文结合问题尝试给出一个弹性的公交车调度方案。

1.2 问题重述

问题一：结合文献以及相关数据，给出一条公交线路的“高峰”和“平峰”的定义，并利用数学语言说明给出定义的合理性及有效性。

问题二：结合问题一给出的高峰和平峰的定义，给定一条数据。给出“转换期”的调度方案模型，并提出判断方案可行的指标，进一步去评价判断参数对于此模型的稳健性和灵敏性。

问题三：建立能够预测高峰和平峰的数学模型。

问题四：结合收集到的实际数据说明模型的可靠性。

2 模型的假设

- 站间公交车匀速行驶，不考虑交叉口、信号灯以及交通路阻的影响；
- 每一个公交站点的距离相等;
- 不考虑上车人数对公交车停靠时间的影响。

3 符号说明

符号	意义
$\lambda(t)$	单位时间内乘车的总人数
g	固定时间间隔
$T(n)$	发车时间
V	公交车承载上限
W	公交车运营总成本

注：其他未列出的符号以第一次出现时的解释为准。

4 问题一的模型建立与求解

4.1 问题分析

问题一需要设计公交车流量的高峰期判断标准。在相关文献中[1]，主要根据经验对早高峰和晚高峰的时段进行划分，然而这种划分方式显然存在一定局限性，无法根据公交车线路的特性进行调整。

为此，本文提出使用公交车登车人数作为数据收集对象，并将其作为整个公交系统中的一个环节的数据，进而通过建立公交动态模拟系统，在系统中获取公交车客流量的衡量指标，最后根据阈值判断高峰期时间段。因为该指标为理论模拟系统中获取的数据，故能够避免由于数据采集中原本不合理的发车间隔安排导致的高峰期的延长。

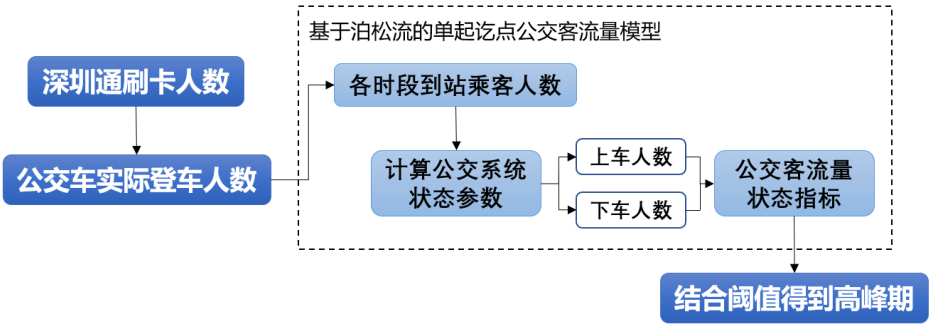

图 1　问题一思路图

4.2 数据预处理

本文通过收集深圳市各公交站点深圳通的刷卡数据，对数据等步长进行取样，再取样后发现纯在数据缺失的情况，于是采用三次样条插值法进行插值处理。

考虑到现实中并非所有人都使用深圳通进行乘车，且根据假设，认为使用深圳通乘车的人数与乘车总人数之间存在正比关系，故有：

$$\lambda(t) = \mu c(t)$$

其中 $c(t)$ 为单位时间内使用深圳通乘车的人数，$\lambda(t)$ 为单位时间内乘车的总人数。

4.3 建立基于泊松流的单起讫点公交客流量模型

4.3.1 基于泊松流的到站客流

记在时间区间 $(0,t](t>0)$ 内到达车站乘客为 $N(t)$, 在时间区间 $(t_1,t_2](t_2>t_1)$ 内到达 n 个乘客的概率为 $P_n(t_1,t_2) = PN(t_2) - N(t_1) = n(n \geq 0)$.

由乘客乘车排队满足泊松流的三个条件：

1、无后效性。在不相互重叠的时间区域内乘客到达数是相互独立的。

2、平稳性。在时间区域 $[a,a+t)$ 内有 n 乘客到达车站的概率与 $P_n(a,a+t)$ 与 a 无关, 仅仅与 t 和 n 有关。

3、普通性。对于充分小的 Δt，在时间区间 $[t, t+\Delta t)$ 内有两个或两个以乘客概率极小，即有 $P_1[t, t+\Delta t) = \lambda\Delta t + o(\Delta t)$，$\sum_{n=2}^{\infty} P_1(t, t+\Delta t) = o(\Delta t)$, 其中 $\lambda > 0$ 是常数，$o(\Delta t)$ 是 Δt 的高阶无穷小。

当输入过程为泊松流时，在长为 t 的时间区间内到达 n 乘客的概率遵从泊松分布，即公式 (1)：

$$P_n(a, a+t) = P_n(0, t) = \frac{(\lambda t)^n}{n!} e^{-\lambda t} \quad (t > 0, n = 0, 1, 2, ...), \tag{1}$$

而在单位长的时间区间内到达的平均顾客数为 λ，即 $E(N(a+1) - N(a)) = \lambda$.

4.3.2 基于单起讫点公交客流量模型确定高峰期

对于以泊松流到达乘客，相继到达的时间间隔服从指数分布：

$$f(t_c) = \begin{cases} 1 - e^{-\lambda x}, & t_c \geq 0 \\ 0, & t_c < 0 \end{cases} \tag{2}$$

可以认为在一段固定的时间 g 内，共到达乘客人数为 $\int_{t-g}^{t} \lambda(t)dt$.

$\lambda(t)$ 表示的是在 t 时刻时，单位时段内到达车站乘客数量。由于每一站上车人数的数据难以查找，我们仅仅收集到每站上车刷卡数，于是我们借助此刷卡数来预计每一站的上车数。如公式（3）

$$\lambda(t) = \mu n_c(t_a, t_a + v), t \in [t_a, t_a + v), \tag{3}$$

其中 μ 为比例系数，t_a 为某个区间的统计起始时间，v 为步长。

假设 $\lambda(t)$ 对于某一个时间节点上一条线路的各站点都相同，故该趟车到达第 k 个站点为止，需要满足的乘车需求量为：$\sum_{i=1}^{k} g\lfloor\lambda(\int_{T+(i-1)t'-g}^{T+(i-1)t'} \lambda(t)dt)\rfloor$，其中 t' 为公交车经过相邻两站所需要花费的时间。

另外认为在任意一个站点上车乘客都将在之后的站点上均匀的下车，故可以认为需要在 k 站点下车的人数为 $\sum_{i=2}^{k} \frac{\lambda(\int_{T+(i-1)t'-g}^{T+(i-1)t'} \lambda(t)dt)}{K+i-1}$.

由此可得对于 T 时刻始发的公交车到达 k 站时需要满足乘客人数需求为：

$$F(T, k, \lambda, g) = \begin{cases} \int_{t-g}^{t} \lambda(t)dt, & k = 1 \\ \sum_{i=1}^{k} g\lambda(\int_{t-g}^{t} \lambda(t)dt) - \sum_{j=2}^{k} \frac{\int_{T+(j-2)t'-g}^{T+(j-2)t'} \lambda(t)dt}{K+1-j}, & 1 < k < K \\ 0, & k = K \end{cases} \tag{4}$$

需要强调，$F(T, k, \lambda, g)$ 并不完全等于公交车上的人数，而是在公交车容量无限的前提下，反映公交车客流量的一种指标。此外，因为该指标是由上述理论模型确定的，故可以避免现实生活中不合理的排班调度导致高峰期异常延长的情况。

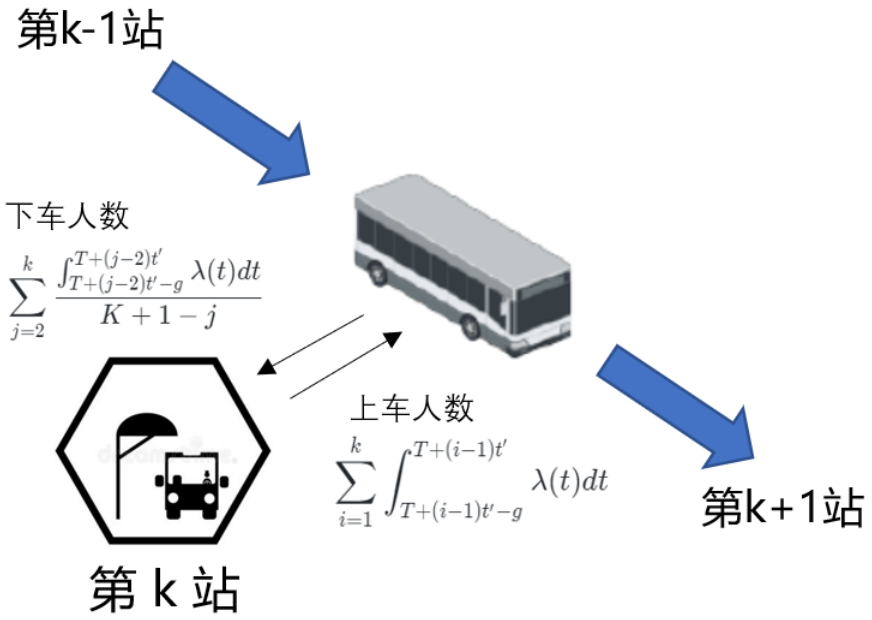

图 2　上下车人数示意图

由此，可以在任意一个事件段内获得若干公交客流量状态指标，并在其中选择最大值作为是否为高峰期的参考依据。本文认为若一个时间段内存在任意 $F(T,k,\lambda,g)$ 与公交车正常容量 V 的比值超过 δ，即可认为该时间段处于高峰期。

$$\max_{(T+(k-1)t')\in(t_a,t_a+\rho)}\frac{F(T,k,\lambda,g)}{V}\geq\delta$$

其中，ρ 为讨论的时间段长度。

4.4 指派问题的模型求解

通过 Sun 等人在《Timetable optimization for single bus line b ased on hybrid vehicle size mode》中的研究, 将以下参数进行确定：

表 1　参数设定

参数/单位	取值
t'/min	5
ρ/min	5
g/min	30
K	10
δ	2

通过 MATLAB 编程，根据深圳市 2014 年 6 月 9 的数据进行计算，得到如下结果：

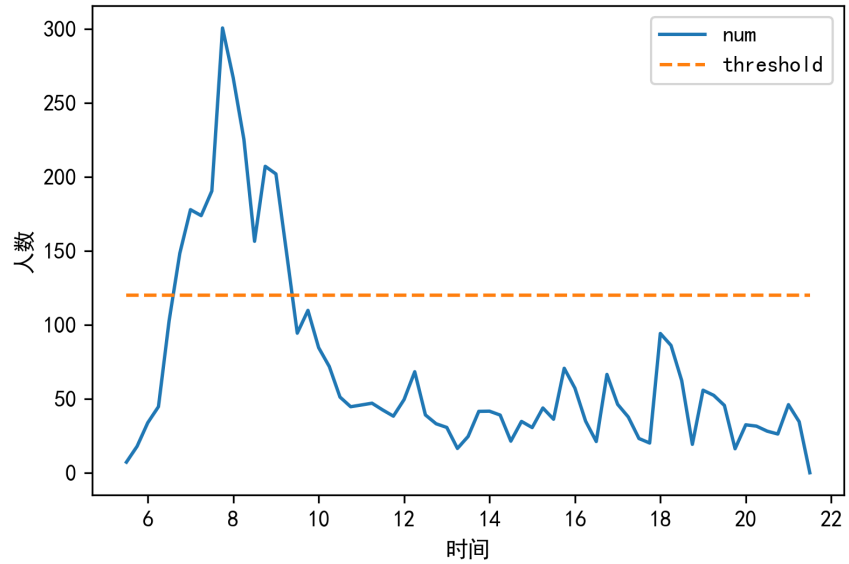

图 3　客流量示意图

综上，该线路于 $6:40-9:20$ 为高峰时期，由于数据的来源于单向公交线路，故对于某个方向仅存在早高峰/晚高峰，基本符合常识。

5 问题二模型的建立与求解

5.1 问题分析

此问题需要根据公交客流量合理安排发车间隔时间，降低乘客等待时间、减少公交运营成本。本文拟通过还原公交车上下车以及站台排队等基本流程与环节，并重点关注其中状态数据的变化。

在此基础上，通过最高净运营成本以及发车顺序等要求构建约束条件，并将公交运营成本和乘客等待时间作为目标函数，建立起以发车间隔安排为决策变量的优化模型。之后，本文引入了公交车状态网络的概念，结合边界条件，得到了整个公交系统状态的运算方法，最后通过帝国竞争算法对发车间隔进行寻优。

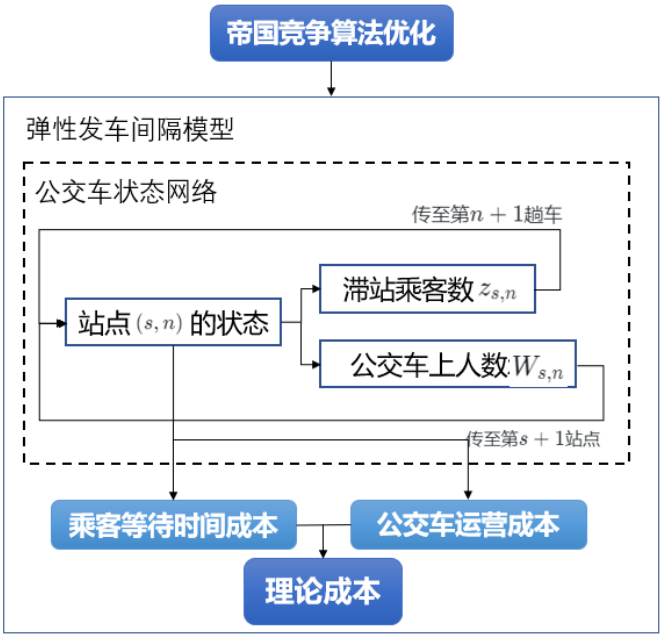

图 4　问题二思路图

5.2 建立弹性发车间隔优化模型

决策变量

求解 n 辆车的发车时间：$T(1), T(2), ..., T(n)$

约束条件

i. 公交车数量的约束：由于公交车发车数不会太大所以我们可以依据生活经验给定上界 $n < 100$

ii. 发车时间的约束：1. 发车时间不会早于最早一班车的发车时间不会晚于最后一班车的发车时间 $t_a \leq T \leq t_b$。2. 下一辆公交车的发车时间一定比上一辆晚 $T(i) \leq T(i+1)$。

iii. 公交车容积约束: 公交车满载情况下可以坐 $V = 60$。

iv. 承载人数分析:

针对此问题，我们只考虑一条公交线路上的一个方向的通勤车上下车情况，假设一天一共需要 N 量车在线路上运行，此线路上一共有 A 个公交车站。此时我们设 $W_{s,n}$ 为第 n 辆车在经过第 s 个车站后车上的人数。其中 $0 < s \leq A, 0 < n \leq N$。我们可以算出在第 s 个车站第 n 辆公交车停靠后，车上改变的人数为:

$$\Delta W = W_{s,n} - W_{s-1,n}. \tag{5}$$

公式 (6) 中，车上改变的人数也等于在该站上车的人数减去下车的人数 $\Delta W =$

$U_{s,n} - D_{s,n}$，式中 $U_{s,n}$,$D_{s,n}$ 分别代表的是该站上车和下车的人数。于是得到公式 (6)

$$\Delta W = W_{s,n} - W_{s-1,n} = U_{s,n} - D_{s,n}. \tag{6}$$

公式 (6) 中上车人数 $U_{s,n}$ 受到两个限制, 当前车站的等待人数和公交车的容量限制。于是我们对上车人数 $U_{s,n}$ 做了如下约定。

$$U_{s,n} = \begin{cases} q_{s,n} & ,V - W_{s-1,n} + D_{s,n} \geq q_{s,n}; \\ V - W_{s-1,n} + D_{s,n} & ,else. \end{cases} \tag{7}$$

公式 (7) 中，表示的是当车站的等待人数小于车上容量 V 时，则所有等待乘客都可以上车，反之则将车坐满剩下的等待乘客滞留在车站 (简称滞站) 准备乘坐下一班公交车。我们将滞站乘客设为 $Z_{s,n}$ 为等待人数减去上车人数，$Z_{s,n} = q_{s,n} - U_{s,n}$，我们将滞站的乘客人数加到下一班车的等待人数 $q_{s,n+1} = Z_{s,n} + \int \lambda(t)dt$

针对公式 (6) 中，我们假设下车人数 $D_{s,n}$ 只取决于车上的人数和接下来的站台数，易得始发站的下车概率为 0，而终点站的下车概率为 1。在中间的站台我们引入下车概率 P_s，为第 n 辆车在第 s 个站台车上人的下车概率。引入下车概率：

$$P_s = \frac{2}{2 + A - s}$$

这个概率可以反映越靠近终点站下车的概率越大，并且保证在最后一站所有乘客都能下车。

目标函数

i. 等待时间成本分析

关注到第 $n-1$ 趟车刚刚经过的第 s 个站点，此站点的乘客都在等待第 n 趟车。我们认为在第 n 趟车到达时，正在排队的人数为 $q_{s,n}$.

这些乘客可以被分为由于未能赶上 $n-1$ 趟车的部分，人数为 $z_{s,n}$；以及在这段时间陆续到达的乘客，人数为 $q_{s,n} - z_{s,n} = \int_{t(n-1)+(s-1)t'}^{t(n)+(s-1)t'} \lambda(t)dt$，这部分乘客到达站点的时间都不相同，但在等间隔期望的假设下，认为等待时间一共为 $1/2(z_{s,n} - q_{s,n})[t(n) - t(n-1)]$.

由此可得所有乘客的等待时间为：

$$T_w = \sum_{i=1}^{n} \sum_{s=1}^{A} \frac{1}{2}(z_{s,n-1} + q_{s,n})[t(n) - t(n-1)] \tag{8}$$

ii. 运营成本分析

我国的公交车运营多依靠政府补贴，因此本文不强调净利润，而是认为总运营成

本不能高于某个阈值。

$$C = \sum_{i=1}^{n} c_i x_i - f_1 \int_{t_b}^{t_a} \lambda(t)dt \leq K_0 \tag{9}$$

c_i 为第 i 趟车的运营开销，由式 (9) 得：

$$c_i = f_2 S t'.$$

其中，f_1 为票价，f_2 为单位时间内的公交车成本。根据田双双在《基于站点需求弹性的多模式公交频率分配研究》确定阈值 $K_0 = 785$。

综上，通过 Sun 等人在《Timetable optimization for single bus line b ased on hybrid vehicle size mode》中的研究给定的单位时间价值系数 $\delta = 8.6$ 元/小时，可以将两个目标转化为同一量纲，故得到总目标函数：

$$\min = w_1 T_w \delta + w_2 C \tag{10}$$

成本目标，在约束下乘客的等待成本和公交车的运营成本之和最小：
定义成本函数 $W = \min = w_1 T_w \delta + w_2 C$

模型总述

$$\begin{aligned}
&\min = w_1 T_w \delta + w_2 C; \\
&s.t. \begin{cases}
n < 100; \\
t_a \leq T(n) \leq t_b; \\
t(i) \leq T(i+1); \\
V = 60; \\
T_w = \sum_{i=1}^{n} \sum_{s=1}^{A} \frac{1}{2}(z_{s,n-1} + q_{s,n})[t(n) - t(n-1)]; \\
C = \sum_{i=1}^{n} c_i x_i - f_1 \int_{t_b}^{t_a} \lambda(t)dt \leq K_0; \\
q_{s,n} - z_{s,n} = \int_{t(n-1)+(s-1)t'}^{t(n)+(s-1)t'} \lambda(t)dt.
\end{cases}
\end{aligned} \tag{11}$$

公交车状态网格迭代运算

考虑到实际过程中一天内的发车过程均为相互关联，且易知对于第 n 趟车经过第 s 个站点时，节点 (s, n) 的状态是根据邻近的两个节点 $(s, n-1)$ 和 $(s-1, n)$ 决定，故在模型求解时，借鉴网格法的基本思想，引入了公交车状态网格对模型进行分析求解。

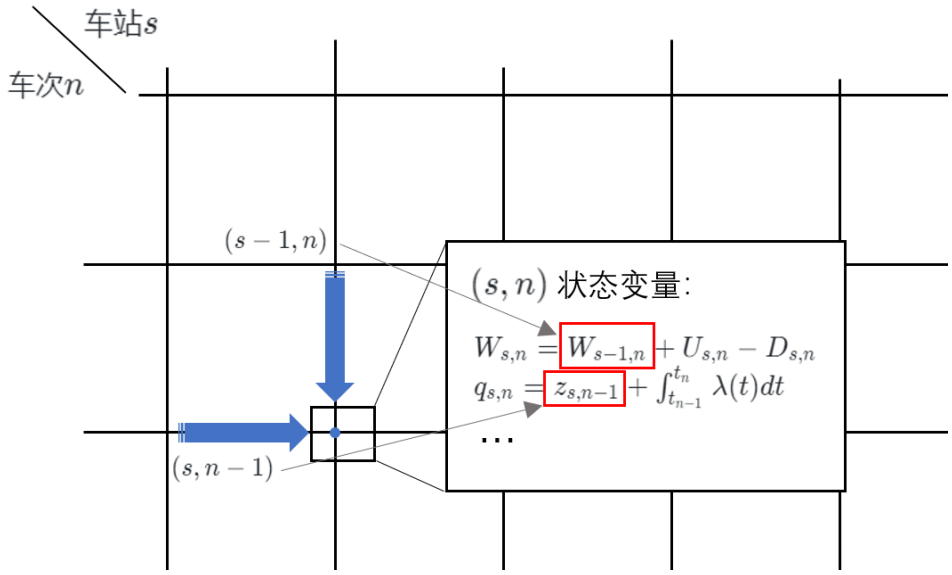

图 5　公交车状态网格迭代示意图

具体状态迭代的公式由上节确定，在此着重说明边界条件的确定。

其中，第一趟车到达各站之前便应该有人在排队。本文认为第一个乘客到达的时间为第一趟车到站的前半个小时，故边界条件为 $z(s,1) = q(s,1) = \int_{t_{s,1}-30}^{t_{s,1}} \lambda(t)dt$.

此外，由于各趟车到达第一个站时车上必然没有人，所以有边界条件 $W_{1,n} = 0$.

综上，结合边界条件和公交车状态网络，在给定一系列发车时间后，便可以计算出任何一趟车在任意一个站点时的状态参数。

5.3 基于帝国竞争算法对优化模型求解

帝国竞争算法 (imperialist competitive algorithm,ICA) 是一种模拟社会政治行为的智能算法，由 Atashpaz-Gargari 等于 2007 年提出, 可以用于解决优化问题。

帝国竞争算法原理

*Step*1 初始化帝国，随机生成一定数量的初始国家, 将成本最小的几个国家定义为殖民国家, 然后根据其自身势力大小为每个殖民国家分配相应数量的殖民地, 构建初始帝国。

*Step*2 同化和革命. 在每个帝国内, 对殖民地执行同化和革命。

*Step*3 殖民国家更新. 比较帝国内的殖民地和殖民国家, 成本最小的国家成为新的殖民国家。

*Step*4 帝国竞争. 根据帝国的总成本, 将最差帝国的最弱殖民地重新分配到新的帝国。

*Step*5 帝国消亡. 如果一个帝国内无任何成员, 则直接剔除这个帝国。

*Step*6 如果终止条件成立, 则搜索结束, 否则转至 *Step*2。

ICA 中种群为所有国家的集合, 每个国家对应问题的一个解，解 $x = (x_1, x_2, ..., x_n) \in R^n$ 中分量 x_i 表示一个国家的文化、语言、经济或宗教等因素, 解的优劣根据其成本

(cost) 决定, 通常要求成本越小, 解越优, 对应的国家势力越大。

初始帝国构建过程中, 殖民国家的归一化成本定义为 $C_k = max\{c_l\} - c_k$, 其中 C_k 表示殖民国家 k 的成本，然后计算殖民国家的势力 $p_k = |\frac{C_k}{\sum_{l=1}^{N_{im}} C_l}|$ 以及每个及每个殖民国家所能分配的殖民地数量. 同化过程可以描述为殖民地沿其自身和殖民国家的连线方向向殖民国家随机移动一段距离 $y\,U(0, \beta \times d)$。其中：d 为殖民地和殖民国家之间的距离；$1 < \beta \leq 2$, 设置 $\beta > 1$ 是为了使殖民地向殖民国家方向移动；$U(a, b)$ 表示区间 (a, b) 上的均匀分布。

革命主要针对少数殖民地进行某种改变, 其作用与 GA 的变异类似, 可以增强探索能力和避免早熟. 帝国竞争是 ICA 的重要步骤, 通过帝国竞争, 势力弱的帝国逐渐失去殖民地, 而势力强的帝国所拥有的殖民地越来越多.

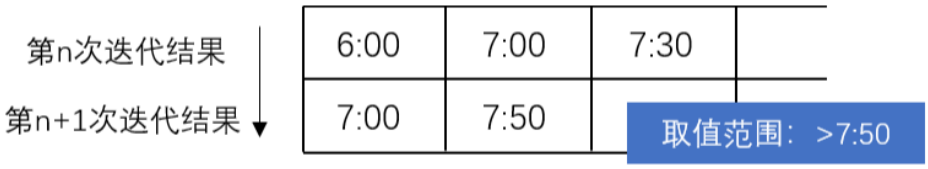

图 6　约束示意图

需要强调，在迭代过程中，显然需要遵守第 $n+1$ 趟车在第 n 趟车之后发车。因此在迭代过程中，需要对每趟车的发车时间进行范围约束。

通过帝国竞争算法我们得出最佳的弹性发车时间组合，图7为发车频数示意图, 表3为具体发车时间表。

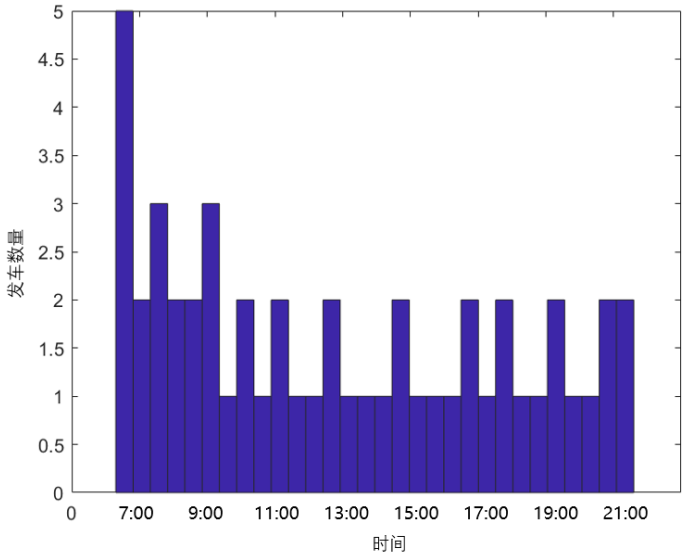

图 7　发车频次示意图

表 2　优化后的弹性发车时间表

发车时间	发车时间	发车时间	发车时间
2014/6/9 6:30	2014/6/9 8:40	2014/6/9 12:15	2014/6/9 16:30
2014/6/9 6:35	2014/6/9 8:50	2014/6/9 12:35	2014/6/9 16:50
2014/6/9 6:45	2014/6/9 9:00	2014/6/9 13:00	2014/6/9 17:10
2014/6/9 6:55	2014/6/9 9:15	2014/6/9 13:20	2014/6/9 17:30
2014/6/9 7:05	2014/6/9 9:30	2014/6/9 13:40	2014/6/9 17:50
2014/6/9 7:15	2014/6/9 9:45	2014/6/9 14:00	2014/6/9 18:10
2014/6/9 7:25	2014/6/9 10:05	2014/6/9 14:25	2014/6/9 18:30
2014/6/9 7:35	2014/6/9 10:25	2014/6/9 14:45	2014/6/9 18:45
2014/6/9 7:45	2014/6/9 10:40	2014/6/9 15:05	2014/6/9 19:00
2014/6/9 7:55	2014/6/9 11:00	2014/6/9 15:20	2014/6/9 19:15
2014/6/9 8:05	2014/6/9 11:20	2014/6/9 15:40	
2014/6/9 8:20	2014/6/9 11:40	2014/6/9 16:00	
2014/6/9 8:30	2014/6/9 11:55	2014/6/9 16:15	

5.4 结果分析

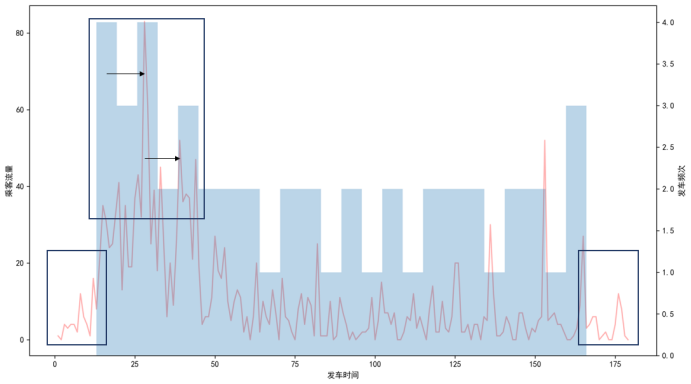

图 8　发车频次与实际客流量对比

观察图8，可以发现对于始发车会在乘客数量到达一定程度时才发车，对应在现实生活中，则为乘客在始发公交车到达之前便在站台排队，因此可以认为与实际情况相符；此外，在末班车发车后，仍然有乘客陆陆续续到站，这部分乘客虽然晚于末班车的发车时间到站，但是存在乘客到达车站 s 时，末班车还在前 $s-1$ 个站，故也与实际情况对应。

此外，关注到客流量高峰期的发车情况，可以发现发车的时间基本上比实际客流高峰提早一段时间，即是否发车取决于后一段时间的客流量。因此客流量预测的准确性将影响到实际运营成本，关于客流量的预测方法将在下章阐述。

6 问题三模型建立与求解

6.1 问题分析

在上一节中，已经得到某一时段发车的情况取决于后一段时间内客流量变化的结论，故必须对客流量进行精准预测，才能保证发车间隔安排合理。然而由于早高峰和晚高峰的存在，使得数据在特定时段上的波动较大，难以用传统的方法对其进行预测。因此，本文拟采用更为灵活的小波神经网络对数据进行预测。

6.2 建立小波神经网络预测模型

本文选用的是以紧致结合性小波神经网络预测模型作为基础，选择 Morlet 小波作为其模型的基函数。

小波神经网络的具体学习步骤

（1）向前传播，计算输出值。隐含层的输入值 h_j 计算公式如公式 (12) 所示：

$$h_j = h(\frac{\sum_{i=1}^{i=m} w_{ij}x_i - b_j}{a_j}), \quad j = 1, 2, 3, ..., m \tag{12}$$

其中，a_j 为小波基函数伸缩因子；b_j 为小波基函数平移因子；h 为小波基函数。小波神经网络的输出值 y_k 计算公式如公式 (6)

$$y_k = \sum_{j=1}^{m} w_{jk}h_j, \quad j = 1, 2, 3, ..., m \tag{13}$$

小波神经网络的输出值 y_k 与理想值 y_k' 之间的误差 $e_k = y_k' - y_k$。误差指标函数 E 为 $E = \frac{1}{2}e_k^2$。

（2）误差反向修改，根据梯度下降学习法，及逆行权值的不断调整。小波神经网络的权值和小波因子调整公式如公式 (14)

$$\begin{aligned} w_{ij}^{(d+1)} &= w_{ij}^{(d)} + \Delta w_{ij}^{(d+1)} \\ w_{ik}^{(d+1)} &= w_{jk}^{(d)} + \Delta w_{jk}^{(d+1)} \\ a_j^{(d+1)} &= a_j^{(d)} + \Delta a_j^{(d+1)} \\ b_j^{(d+1)} &= b_j^{(d)} + \Delta b_j^{(d+1)} \end{aligned} \tag{14}$$

其中，d 表示训练次数；$\Delta w_{ij}^{(d+1)}$、$\Delta w_{jk}^{(d+1)}$、$\Delta a_j^{(d+1)}$、$\Delta b_j^{(d+1)}$ 分别表示 d+1 次训

练的各数值的调整量，其调整公式如公式 (15)

$$\begin{aligned}\Delta w_{ij}^{(d+1)} &= -\eta\frac{\partial E}{\partial w_{ij}^{(d)}}\\ \Delta w_{ik}^{(d+1)} &= -\eta\frac{\partial E}{\partial w_{jk}^{(d)}}\\ \Delta a_{j}^{(d+1)} &= -\eta\frac{\partial E}{\partial a_{j}^{(d)}}\\ \Delta b_{j}^{(d+1)} &= -\eta\frac{\partial E}{\partial b_{j}^{(d)}}\end{aligned} \tag{15}$$

其中，η 表示小波神经网络的学习速率；E 表示小波神经网络的误差性能指标函数。

为了加快小波神经网络的收敛速度，改进小波神经网络权值和小波因子的调整公式，具体改进如式 (16)

$$\begin{aligned}w_{ij}^{(d+1)} &= w_{ij}^{(d)} + \Delta w_{ij}^{(d+1)} + \alpha(w_{ij}^{(d)} - w_{ij}^{(d-1)})\\ w_{ik}^{(d+1)} &= w_{jk}^{(d)} + \Delta w_{jk}^{(d+1)} + \alpha(w_{jk}^{(d)} - w_{jk}^{(d-1)})\\ a_{j}^{(d+1)} &= a_{j}^{(d)} + \Delta a_{j}^{(d+1)} + \alpha(a_{j}^{(d)} - a_{j}^{(d-1)})\\ b_{j}^{(d+1)} &= b_{j}^{(d)} + \Delta b_{j}^{(d+1)} + \alpha(b_{j}^{(d)} - b_{j}^{(d-1)})\end{aligned} \tag{16}$$

其中，α 为动量因子，$\alpha \in [0,1]$。

小波基函数的选择

小波变换中常被学者使用的基函数有 Haar 小波、MexicanHat 小波等，通过翻阅文献后决定使用 Morlet 小波作为基函数。图9为 Morlet 小波示意图。

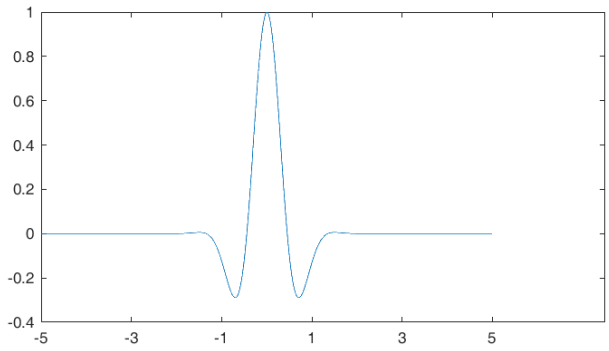

图 9　Morlet 小波示意图

6.3 基于小波神经网络客流量预测模型的求解

基于小波神经网络的预测主要可以分为两个阶段：训练阶段和预测阶段。图10为小波审计为了预测的流程示意图。

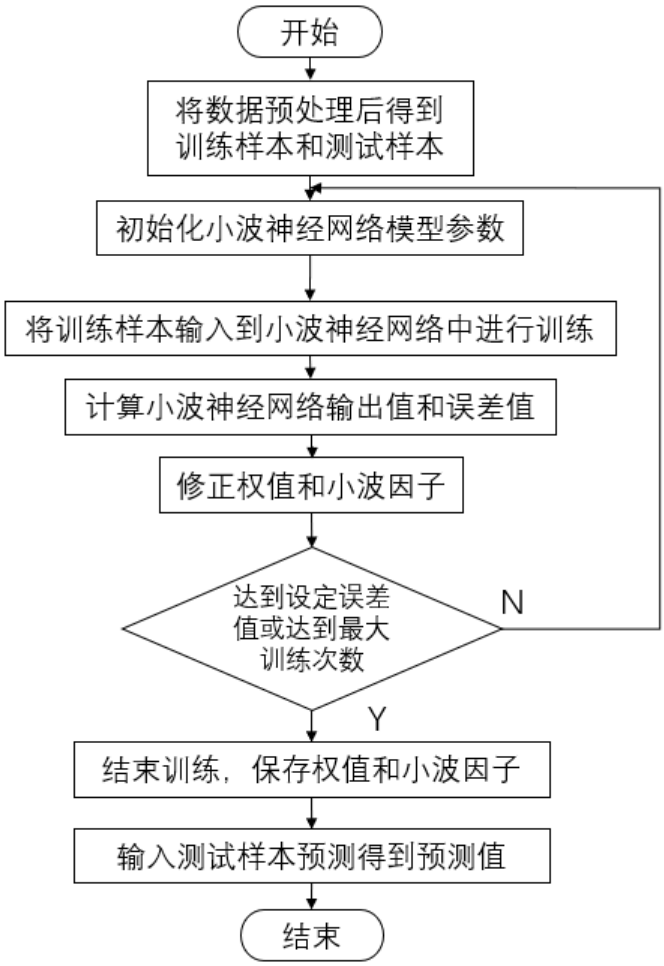

图 10　小波神经网络预测流程图

具体操作步骤如下：

（1）对原始客流量数据进行预处理；

（2）初始化小波神经网络模型中的参数，如权值和小波因子等；

（3）输入训练数据进行训练；

（4）得出输出值和误差值；

（5）运用梯度下降算法调整各项参数值；

（6）判断是否达到了设定的误差值或是否达到最大训练次数，如果达到，则结束训练；反之，则继续执行第三步操作。

（7）训练阶段结束，保存小波神经网络的权值和小波因子；

（8）输入测试数据样本进行客流量的预测；

（9）评价结果性能的优劣性。

最后通过 matlab 求解，我们得出如下结果，如图11：

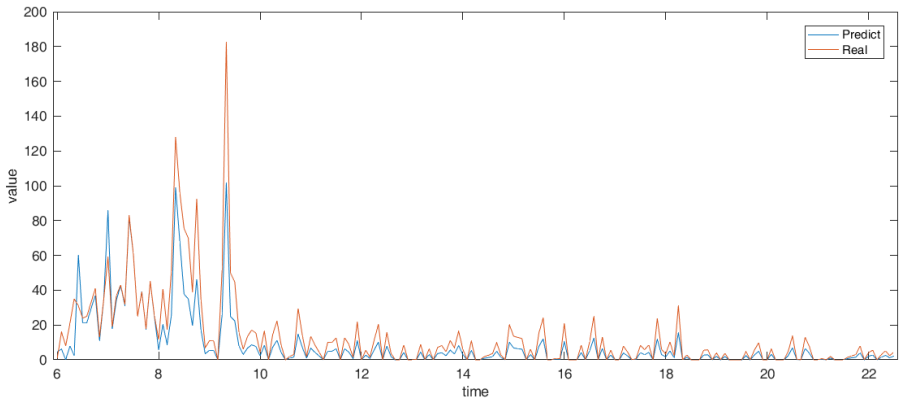

图 11　客流量预测示意图

6.4 结果分析

我们将预测的结果和先前客流量的平均数做对比形成为图13的误差示意图。

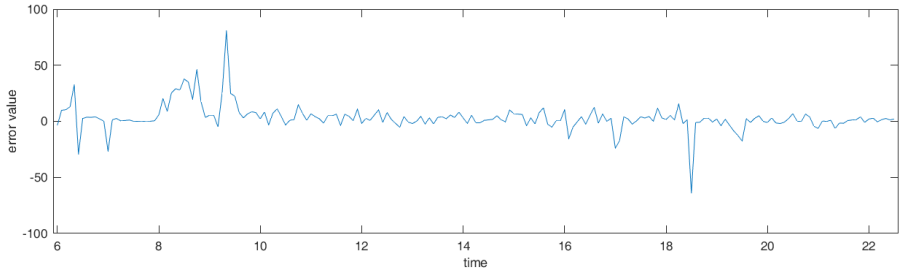

图 12　误差分析示意图

根据误差分析图，可以认为客流量的预测数据基本准确。

7 问题四的模型与求解

7.1 问题分析

在上文中，本文已经就公交车的弹性发车间隔在理论层面上进行了讨论，对于问题四则需要分析方案在实际运行过程中的有效性。考虑到实际工作中发车间隔需要事先预定，本文拟通过将问题三中的预测结果作为发车间隔调整依据，并将此发车间隔方案运用至实际客流量中。之后再针对实际客流量进行后验分析，获取理论最优的发车间隔。最后对比两方案的成本，进而分析本文设计的弹性发车间隔调整方案在实际运行中的有效性。

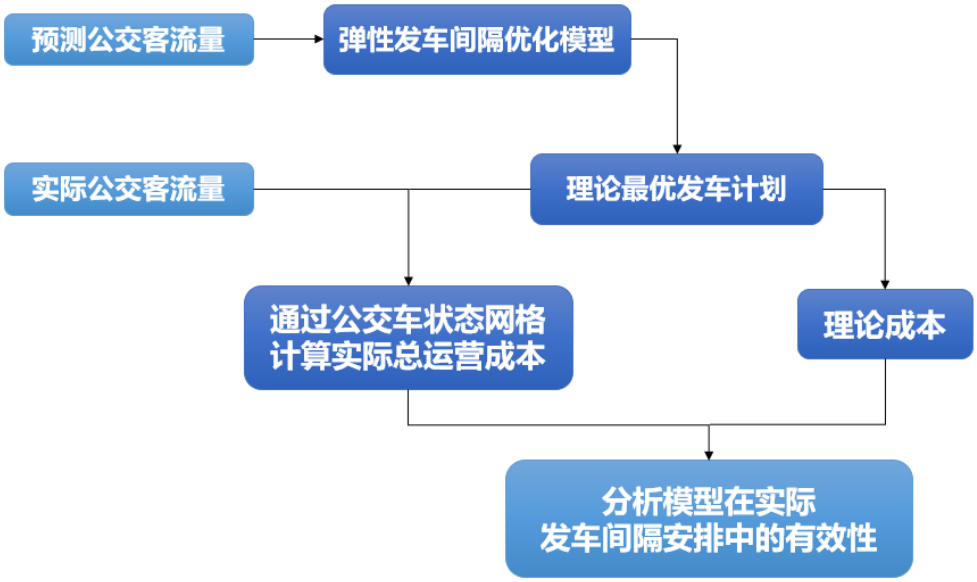

图 13　问题四思路示意图

7.2 弹性发车间隔模型实际运行验证

首先，将问题三中计算得到的客流量预测值代入到问题二的弹性发车间隔优化模型中，得到实际发车方案以及理论运营成本：

表 3　发车时间方案

发车时间	发车时间	发车时间	发车时间
2014/6/9 6:25	2014/6/9 8:35	2014/6/9 12:45	2014/6/9 17:35
2014/6/9 6:30	2014/6/9 8:50	2014/6/9 13:05	2014/6/9 17:55
2014/6/9 6:35	2014/6/9 9:00	2014/6/9 13:25	2014/6/9 18:15
2014/6/9 6:40	2014/6/9 9:10	2014/6/9 13:50	2014/6/9 18:35
2014/6/9 6:45	2014/6/9 9:25	2014/6/9 14:15	2014/6/9 18:55
2014/6/9 6:55	2014/6/9 9:40	2014/6/9 14:40	2014/6/9 19:10
2014/6/9 7:05	2014/6/9 10:00	2014/6/9 15:00	2014/6/9 19:25
2014/6/9 7:15	2014/6/9 10:20	2014/6/9 15:20	2014/6/9 20:45
2014/6/9 7:25	2014/6/9 10:40	2014/6/9 15:40	2014/6/9 21:05
2014/6/9 7:35	2014/6/9 11:00	2014/6/9 15:55	2014/6/9 21:20
2014/6/9 7:45	2014/6/9 11:15	2014/6/9 16:10	2014/6/9 21:35
2014/6/9 7:55	2014/6/9 11:30	2014/6/9 16:25	2014/6/9 21:55
2014/6/9 8:05	2014/6/9 11:45	2014/6/9 16:45	2014/6/9 22:15
2014/6/9 8:15	2014/6/9 12:00	2014/6/9 17:05	
2014/6/9 8:25	2014/6/9 12:20	2014/6/9 17:20	

其中理论运营成本为 37294 元。

将上述发车方案代入到公交车状态网络中，可以得到下述到站时间表格：

表 4　发车状态

6:25	6:30	…	7:15
6:30	6:35	…	7:20
6:35	6:40	…	7:25
6:40	6:45	…	7:30
6:45	6:50	…	7:35
6:55	7:00	…	7:45
…	…		…
22:15	22:20	…	23:10

并基于此对每个节点的状态指标进行迭代运算，进而可以计算出此方案的净运营成本为 51690 元。

表 5　成本汇总

	实际客流理论运营成本	基于预测的实际客流运营成本
成本	43726	51690

显然，由于预测存在一定误差，使得运营成本比理论最低运营成本结果提高了 15.4%，但基本在可接受范围内。

8 稳健性和敏感性分析

8.1 敏感性分析

最优化模型求解中常用灵敏度分析来研究原始数据发生变化时最优解的稳定性。本文通过改变单位运营的成本和 w_1, w_2 来探究最优解的稳定性, 本文在默认参数附近百分之五的范围内以等步长进行搜索。结果如表6和表7：

表 6　单位时间运营成本变化

单位时间运营成本变化	总成本变化
150	43726
147.5	49544
145	47874
152.5	47964
155	50959

表 7　等待成本权值与运营成本权值变化

w_1, w_2	总成本变化
1,1	43726
0.9,1.1	47213
0.8,1.2	38806
1.1,0.9	55165
1.2,0.8	57683

通过改变参数的数值，我们可以看到最优化的总成本变化不大，由此可知，本文的弹性发车间隔优化模型的敏感度较低。

8.2 稳健性分析

当改变某些参数，考察数据变化模型求解的影响。本文给客流量数据加一个白噪声，将处理过后的数据带入原模型，考察解的变换趋势。如图14

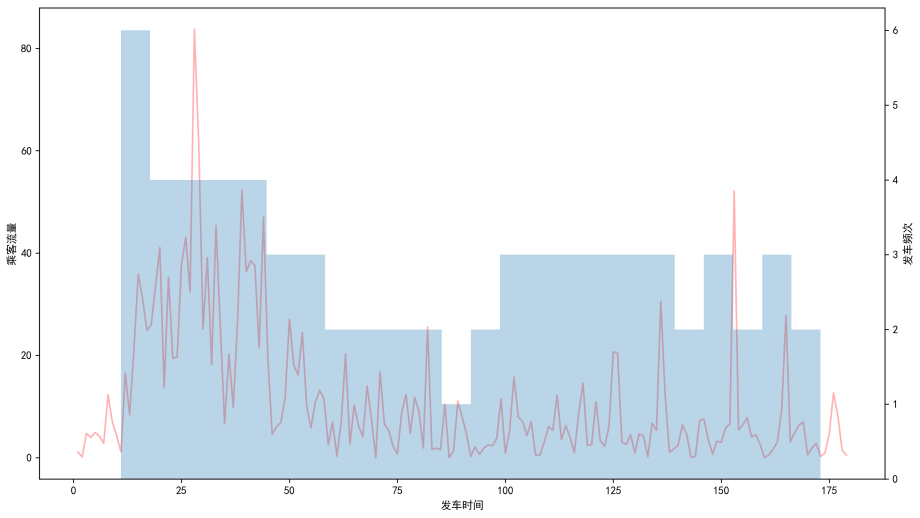

图 14　加噪声之后的示意图

观察加了噪声后的模型求解图，可以观察到噪声对公交车发车间隔的影响很小，证明了此模型具有较好的稳健性。

9 模型总结与评价

9.1 模型优点

1. 运用帝国竞争算法进行求解，该算法收敛速度快、收敛精度较高，具有较强的全局收敛性。

2. 通过借鉴网格法的思想引入了公交车状态网络，考察了每个节点的状态，便于后续分析。

3. 通过目标函数的设计确保净运营成本不超过阈值的前提下，同时考虑乘客的等待时间和运营开销，使结果能够保证两者的利益。

9.2 模型缺点

1. 由于数据获取的限制，本问仅使用了深圳市 2014 年 6 月 9 日的深圳通刷卡数据，使结果存在一定局限性。

2. 考虑到模型的复杂程度，本问未对高峰期的堵塞导致的到站时间延误的情况进行讨论。

9.3 模型改进

在实际运营的过程中，还需要对公交车线路之间的竞争、突发大型事件的临时车辆调度方案等内容进行讨论。并且在学术界中，对于客流量高峰的讨论主要集中在瓶颈模型 Vickery W 在《Congestion theory and transport investment》中，故将来时间允许，可以考虑获得更精确的数据，从更全面的角度，结合瓶颈模型的相关理论进行分析。

参考文献

[1] 陆百川, 何相嶬, 刁素素, 等. 城市外围非高峰时段多线路柔性公交协调调度研究 [J]. 公路交通科技,2020,37(05):131-139+158.

[2] 汤月华. 基于 GPS 数据的公交站点区间行程时间分布与可靠性分析 [D]. 浙江大学,2015.

[3] 曹全新, 王武宏, 沈中杰. 基于多目标规划的公共交通调度算法与仿真 [J]. 交通科技与经济,2004(02):48-50.

[4] 李泰照. 基于改进蚁群的混合小波神经网络短时交通流预测 [D]. 东华理工大学,2019.

[5] 牛帅. 基于公交客流分布特性的弹性发车间隔优化研究 [D]. 大连交通大学,2018.

[6] 尚华艳, 王闪, 黄海军, 等. 基于活动的瓶颈模型: 公交枢纽晚高峰居民通勤研究 [J]. 系统工程理论与实践,2020,40(03):679-690

[7] 张蕴琦. 基于客流动态特性的公交调度优化方法研究 [D]. 大连交通大学,2019.

[8] William S. Vickrey. Congestion Theory and Transport Investment. 1969, 59(2):251-260.

[9] 田双双. 基于站点需求弹性的多模式公交频率分配研究 [D]. 长安大学,2017.

[10] 雷德明, 操三强, 李明. 求解约束优化问题的新型帝国竞争算法 [J]. 控制与决策,2019,34(08):1663-1671.

附录 A 代码

A.1 帝国竞争 matlab 源程序

```
function TheEmpire =
    AssimilateColonies(TheEmpire,AlgorithmParams,ProblemParams)

% for i = 1:numel(Imperialists)
%     Imperialists{i}.Number_of_Colonies_matrix =
    [Imperialists{i}.Number_of_Colonies_matrix
    Imperialists{i}.Number_of_Colonies];
%
%     Imperialists_cost_matrix (i) = Imperialists{i}.cost_just_by_itself;
%
%     Imperialists_position_matrix(i,:) = Imperialists{i}.position;

NumOfColonies = size(TheEmpire.ColoniesPosition,1);

Vector =
    repmat(TheEmpire.ImperialistPosition,NumOfColonies,1)-TheEmpire.ColoniesPosition;

% 这里怎么没用上偏移角度？？
TheEmpire.ColoniesPosition = TheEmpire.ColoniesPosition +
    AlgorithmParams.AssimilationCoefficient * rand(size(Vector)) .* Vector;
TheEmpire.ColoniesPosition = round(TheEmpire.ColoniesPosition);
TheEmpire.ColoniesPosition = sort(TheEmpire.ColoniesPosition,2);
MinVarMatrix = repmat(ProblemParams.VarMin,NumOfColonies,1);
MaxVarMatrix = repmat(ProblemParams.VarMax,NumOfColonies,1);

% 这一步是将非解空间的解拉回边界
TheEmpire.ColoniesPosition=max(TheEmpire.ColoniesPosition,MinVarMatrix);
TheEmpire.ColoniesPosition=min(TheEmpire.ColoniesPosition,MaxVarMatrix);
```

A.2 main matlab 源程序

```
function R=BenchmarkFunction(x)
global num;
%%%%%%%%%%todo时间在:的索引之外num
mu = 0.25;
A = 10;
V = 60; % 车载容量
```

```
w1=1;
w2=1;
for solution = 1:size(x, 1) % 对第一个解
   bus_start_time = x(solution, :);
   % 对第一辆车
   % 第一个站
   D(1,1) = 0;
   coming = 0;
   for i = 1:bus_start_time(1)
      coming = coming+mu*num(i);
   end
   Q(1,1)=coming;
   if Q(1,1) > V
      U(1,1) = V;
      W(1,1) = V;
   else
      U(1,1) = coming;
      W(1,1) = coming;
   end
   Z(1,1)=Q(1,1)-U(1,1);
   % 对个站2-9
   for i = 2:A-1
      D(1,i) = W(1,i-1)*2/(2+A-i);
      coming = 0;
      for j = 1:bus_start_time(1)+i-1
         coming = coming+mu*num(j);
      end
      Q(1,i) = coming;
      if V-W(1,i-1)+D(1,i) >= Q(1,i)
         U(1,i) = Q(1,i);
         W(1,i) = W(1,i-1)-D(1,i)+Q(1,i);
      else
         U(1,i) = V-W(1,i-1)+D(1,i);
         W(1,i) = V;
      end
      Z(1,i)=Q(1,i)-U(1,i);
   end
   % 对第个站10
   D(1,10)=W(1,A-1);
   Q(1,10)=0;
   U(1,10)=0;
   W(1,10)=0;
   Z(1,10)=0;
```

```
% 对后面的车
for i=2:size(x, 2)
   % 第一个站
   D(i,1)=0;
   coming = 0;
   for j = bus_start_time(i-1)+1:bus_start_time(i)
      coming = coming+mu*num(j);
   end
   Q(i,1)=coming+Z(i-1,1);
   if Q(i,1) > V
      U(i,1)=V;
      W(i,1)=V;
   else
      U(i,1)=Q(i,1);
      W(i,1)=Q(i,1);
   end
   Z(i,1)=Q(i,1)-U(i,1);
   %对个站2-9
   for j = 2:A-1
      D(i,j) = W(i,j-1)*2/(2+A-j);
      coming = 0;
      for jj = bus_start_time(i-1)+j:bus_start_time(i)+j-1
         coming = coming + mu*num(jj);
      end
      Q(i,j) = Z(i-1,j)+coming;
      if V-W(i,j-1)+D(i,j) < Q(i,j)
         U(i,j)=V-W(i,j-1)+D(i,j);
         W(i,j)=V;
      else
         U(i,j)=Q(i,j);
         W(i,j)=W(i,j-1)-D(i,j)+U(i,j);
      end
      Z(i,j)=Q(i,j)-U(i,j);
   end
   % 对第个站10
   D(i,A)=W(i,A-1);
   Q(i,A)=0;
   U(i,A)=0;
   W(i,A)=0;
   Z(i,A)=0;
end
T(solution) = 0;
for i=2:size(x,2)
```

```
        for s=1:A
            T(solution) =
                T(solution)+(Z(i-1,s)+5*Q(i,s))*(bus_start_time(i)-bus_start_time(i-1)).^2/2;
        end
    end
    T(solution) = T(solution)+5*(Z(1,s)+Q(1,s))*(bus_start_time(1)-1)/2;
    T(solution) =
        T(solution)+5*(Z(1,s)+Q(1,s))*(209-bus_start_time(size(x,2)))/2;
    %T
    C(solution)=0;
    for i = 1:209
        C(solution) = C(solution)-mu*num(i);
    end
    C(solution) = C(solution) + size(x,2)*150/60*10*5;
    R(solution) = w1*T(solution)*30/60+w2*C(solution);
  end
end
```

附录 B 代码

B.1 帝国竞争 matlab 源程序

```
  function TheEmpire =
      AssimilateColonies(TheEmpire,AlgorithmParams,ProblemParams)

  % for i = 1:numel(Imperialists)
  %     Imperialists{i}.Number_of_Colonies_matrix =
      [Imperialists{i}.Number_of_Colonies_matrix
      Imperialists{i}.Number_of_Colonies];
  %
  %     Imperialists_cost_matrix (i) = Imperialists{i}.cost_just_by_itself;
  %
  %     Imperialists_position_matrix(i,:) = Imperialists{i}.position;

  NumOfColonies = size(TheEmpire.ColoniesPosition,1);

  Vector =
      repmat(TheEmpire.ImperialistPosition,NumOfColonies,1)-TheEmpire.ColoniesPosition;

  % 这里怎么没用上偏移角度？？
  TheEmpire.ColoniesPosition = TheEmpire.ColoniesPosition +
      AlgorithmParams.AssimilationCoefficient * rand(size(Vector)) .* Vector;
```

```
TheEmpire.ColoniesPosition = round(TheEmpire.ColoniesPosition);
TheEmpire.ColoniesPosition = sort(TheEmpire.ColoniesPosition,2);
MinVarMatrix = repmat(ProblemParams.VarMin,NumOfColonies,1);
MaxVarMatrix = repmat(ProblemParams.VarMax,NumOfColonies,1);

% 这一步是将非解空间的解拉回边界
TheEmpire.ColoniesPosition=max(TheEmpire.ColoniesPosition,MinVarMatrix);
TheEmpire.ColoniesPosition=min(TheEmpire.ColoniesPosition,MaxVarMatrix);
```

B.2 第四问 matlab 源程序

```
close all
clc; clear
global num;
[num, txt] = xlsread("../data2.xlsx");
for i = 1:50
   num(end+1)=0;
end
%% Problem Statement
ProblemParams.CostFuncName = 'BenchmarkFunction'; % You should state the
    name of your cost function here.
ProblemParams.CostFuncExtraParams = '';
ProblemParams.NPar = 60;                    % Number of optimization
    variables of your objective function. "NPar" is the dimention of the
    optimization problem.
ProblemParams.VarMin = 1;                   % Lower limit of the
    optimization parameters. You can state the limit in two ways. 1) 2)
ProblemParams.VarMax = 209;                 % Lower limit of the
    optimization parameters. You can state the limit in two ways. 1) 2)


ProblemParams.SearchSpaceSize = ProblemParams.VarMax - ProblemParams.VarMin;

%% Algorithmic Parameter Setting
AlgorithmParams.NumOfCountries = 200;        % Number of initial countries.
AlgorithmParams.NumOfInitialImperialists = 8; % Number of Initial
    Imperialists.
AlgorithmParams.NumOfAllColonies = AlgorithmParams.NumOfCountries -
    AlgorithmParams.NumOfInitialImperialists;
AlgorithmParams.NumOfDecades = 2000;
AlgorithmParams.RevolutionRate = 0.3;        % Revolution is the process in
    which the socio-political characteristics of a country change suddenly.
```

```
AlgorithmParams.AssimilationCoefficient = 2; % In the original paper
    assimilation coefficient is shown by "beta".
AlgorithmParams.AssimilationAngleCoefficient = .5; % In the original paper
    assimilation angle coefficient is shown by "gama".
AlgorithmParams.Zeta = 0.02;                % Total Cost of Empire = Cost
    of Imperialist + Zeta * mean(Cost of All Colonies);
AlgorithmParams.DampRatio = 0.99;
AlgorithmParams.StopIfJustOneEmpire = false;  % Use "true" to stop the
    algorithm when just one empire is remaining. Use "false" to continue the
    algorithm.
AlgorithmParams.UnitingThreshold = 0.02;     % The percent of Search Space
    Size, which enables the uniting process of two Empires.

zarib = 1.05;                   % **** Zarib is used to prevent the weakest
    impire to have a probability equal to zero
alpha = 0.1;                    % **** alpha is a number in the interval of
    [0 1] but alpha<<1. alpha denotes the importance of mean minimum compare
    to the global mimimum.

%% Display Setting
DisplayParams.PlotEmpires = false; % "true" to plot. "false" to cancel
    ploting.
if DisplayParams.PlotEmpires
   DisplayParams.EmpiresFigureHandle = figure('Name','Plot of
       Empires','NumberTitle','off');
   DisplayParams.EmpiresAxisHandle = axes;
end

DisplayParams.PlotCost = false; % "true" to plot. "false"
if DisplayParams.PlotCost
   DisplayParams.CostFigureHandle = figure('Name','Plot of Minimum and Mean
       Costs','NumberTitle','off');
   DisplayParams.CostAxisHandle = axes;
end

ColorMatrix = [1 0  0  ; 0 1  0   ; 0  0 1   ; 1  1  0  ; 1  0 1   ; 0 1  1
      ; 1 1 1     ;
          0.5 0.5 0.5; 0 0.5 0.5 ; 0.5 0 0.5 ; 0.5 0.5 0 ; 0.5 0 0 ; 0 0.5
              0 ; 0 0 0.5 ;
          1   0.5 1 ; 0.1*[1 1 1]; 0.2*[1 1 1]; 0.3*[1 1 1]; 0.4*[1 1 1];
              0.5*[1 1 1]; 0.6*[1 1 1]];
DisplayParams.ColorMatrix = [ColorMatrix ; sqrt(ColorMatrix)];
```

```
DisplayParams.AxisMargin.Min = ProblemParams.VarMin;
DisplayParams.AxisMargin.Max = ProblemParams.VarMax;

%% Creation of Initial Empires
InitialCountries = GenerateNewCountry(AlgorithmParams.NumOfCountries ,
    ProblemParams);

% Calculates the cost of each country. The less the cost is, the more is
    the power.
if isempty(ProblemParams.CostFuncExtraParams)
   InitialCost = feval(ProblemParams.CostFuncName,InitialCountries);
else
   InitialCost =
       feval(ProblemParams.CostFuncName,InitialCountries,ProblemParams.CostFuncExtraParams
end
[InitialCost,SortInd] = sort(InitialCost);                % Sort the cost
    in assending order. The best countries will be in higher places
InitialCost = InitialCost';
InitialCountries = InitialCountries(SortInd,:);           % Sort the
    population with respect to their cost.

Empires =
    CreateInitialEmpires(InitialCountries,InitialCost,AlgorithmParams,
    ProblemParams);

%% Main Loop
MinimumCost = repmat(nan,AlgorithmParams.NumOfDecades,1);
MeanCost = repmat(nan,AlgorithmParams.NumOfDecades,1);

if DisplayParams.PlotCost
   axes(DisplayParams.CostAxisHandle);
   if any(findall(0)==DisplayParams.CostFigureHandle)
      h_MinCostPlot=plot(MinimumCost,'r','LineWidth',1.5,'YDataSource','MinimumCost');
      hold on;
      h_MeanCostPlot=plot(MeanCost,'k:','LineWidth',1.5,'YDataSource','MeanCost');
      hold off;
      pause(0.05);
   end
end

for Decade = 1:AlgorithmParams.NumOfDecades
   AlgorithmParams.RevolutionRate = AlgorithmParams.DampRatio *
       AlgorithmParams.RevolutionRate;
```

```
Remained = AlgorithmParams.NumOfDecades - Decade;
for ii = 1:numel(Empires)
   %% Assimilation; Movement of Colonies Toward Imperialists
       (Assimilation Policy)
   Empires(ii) =
       AssimilateColonies(Empires(ii),AlgorithmParams,ProblemParams);

   %% Revolution; A Sudden Change in the Socio-Political Characteristics
   % 该殖民地改革操作可能导致势力较强的殖民地丢失，导致寻优精度降低
   Empires(ii) =
       RevolveColonies(Empires(ii),AlgorithmParams,ProblemParams);

   %% New Cost Evaluation
   if isempty(ProblemParams.CostFuncExtraParams)
      Empires(ii).ColoniesCost =
          feval(ProblemParams.CostFuncName,Empires(ii).ColoniesPosition);
      Empires(ii).ColoniesCost = Empires(ii).ColoniesCost';
   else
      Empires(ii).ColoniesCost =
          feval(ProblemParams.CostFuncName,Empires(ii).ColoniesPosition,ProblemParams.Co
   end

   %% Empire Possession (****** Power Possession, Empire Possession)
   Empires(ii) = PossesEmpire(Empires(ii));

   %% Computation of Total Cost for Empires
   Empires(ii).TotalCost = Empires(ii).ImperialistCost +
       AlgorithmParams.Zeta * mean(Empires(ii).ColoniesCost);

end

%% Uniting Similiar Empires
Empires = UniteSimilarEmpires(Empires,AlgorithmParams,ProblemParams);

%% Imperialistic Competition
Empires = ImperialisticCompetition(Empires);

if numel(Empires) == 1 && AlgorithmParams.StopIfJustOneEmpire
   break
end

%% Displaying the Results
```

```
    DisplayEmpires(Empires,AlgorithmParams,ProblemParams,DisplayParams);

    ImerialistCosts = [Empires.ImperialistCost];
    MinimumCost(Decade) = min(ImerialistCosts);
    MeanCost(Decade) = mean(ImerialistCosts);

    if DisplayParams.PlotCost
       refreshdata(h_MinCostPlot);
       refreshdata(h_MeanCostPlot);
       drawnow;
       pause(0.01);
    end

end % End of Algorithm
MinimumCost(end)
```