目录

一、 问题重述

1.1 问题背景

在物资调运过程中，完成指定点的调运任务是最基本的要求，在完成基本的任务之外，往往有更高的追求，比如如何使总运费最省？怎样才能使得运输时间最短？如何选择运输路径使得运输总距离最短等等。这些更高的追求往往是企业期望达到的目标，为了解决这些类似问题，有必要对物资调运的过程进行数学模型的建立，以期通过模型来理解和分析物资调运的过程，并为其找到解决的方法。现以具体的食品调运案例进行分析研究。

某食品公司有 19 个食品销售点，销售点的地理坐标和每天的需求量见附件。每天凌晨都要从仓库（第 20 号站点）出发将食品运至每个销售点，运送物品后最终返回仓库。现有运送食品的运输车，每台车每日工作 4 小时，运输车重载运费 2 元/吨公里，并且假定街道方向均平行于坐标轴，任意两站点间都可以通过一次拐弯到达。

1.2 问题概述

围绕相关附件和条件要求，研究食品运输车在各仓库间的调度方案，依次提出以下问题:

问题一：若只有一辆载重 100 吨的大型运输车，运输车平均速度为 40 公里 / 小时，每个销售点需要用 20 分钟的时间下货，空载费用 0.6 元/公里。它送完所有食品并回到仓库，求最少需要时间及其对应的总距离，总运费。

问题二：有一种小型运输车，运输车平均速度为 50 公里 / 小时，每个销售点需要用 5 分钟的时间下货，载重为 6 吨，空载费用 0.4 元/公里；要使它们送完所有食品并回到仓库，运输车应如何调度使总体调度效率最高？

问题三：如果有载重量为 4 吨、6 吨两种运输车，空载费用分别为 0.2、0.4 元/公里，其他条件均相同，又如何安排车辆数和调度方案。

二、 模型假设

(1) 为保证预测结果精确性，假设题目所给出数据真实可信。
(2) 假设重点防控的区域和人群中，发病、死亡人数的增长率比其基数更加重要

三、 符号说明

符号	说明
$X^{(i)}$	人数时间序列
a	发展灰度
u	内生控制灰度

四、 问题一模型的建立与求解

4.1 问题描述与分析

其思维流程图如图 1 所示：

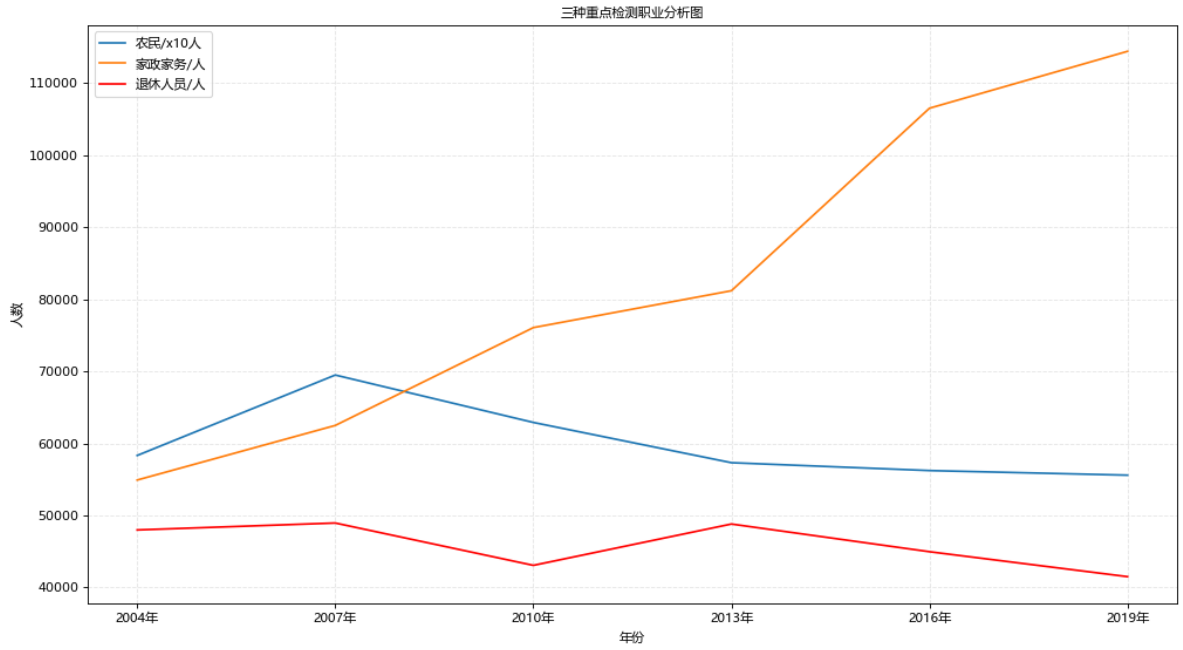

图 **1**　问题一思维流程图

4.2 模型的建立

4.2.1 灰度预测 GM(1,1)

设 2004-2016 年总发病人数为时间序列：

$$X^{(0)} = [x^{(0)}(1), x^{(0)}(2), \cdots, x^{(0)}(13)]$$

其误差状态区间如表 1 所示：

表 **1**　发病人数状态区间划分

状态	E_1	E_2	E_3
残差区间	$[-66389,-22130]$	$(-22130,22130]$	$(22130,66389]$

五、 问题二模型的建立与求解

5.1 问题描述与分析

5.2 模型的建立

5.3 模型的求解

六、 灵敏度分析

七、 模型的评价

7.1 模型的优点

(1) 利用马尔可夫模型改进后的灰度预测值与实际值拟合度更高，波动性保持一致，预测的效果更好。

(2) 针对支持向量回归参数选取，利用灰色关联度筛选合适指标，相较于主观选取指标具有客观性、严谨性。

7.2 模型的缺点

问题一、二中的灰色预测模型只能做短期预测，并不适用于长期预测。

7.3 模型改进

可以通过序列最小优化算法 (Sequential Minimal Optimization,SMO) 作为样本的训练算法，进而建立序列最小优化支持向量回归模型，从而减小算法复杂度，提高算法的求解速度。

参考文献

[1] 张斯嘉, 郭建胜, 钟夫, 等. 基于蝙蝠算法的多目标战备物资调运决策优化 [J]. 火力与指挥控制, 2016, 41(1): 58-61.

[2] 李健, 张文文, 白晓昀, 等. 基于系统动力学的应急物资调运速度影响因素研究 [J]. 系统工程理论与实践, 2015, 35(3): 661-670.

[3] Wang J, Ersoy O K, He M, et al. Multi-offspring genetic algorithm and its application to the traveling salesman problem[J]. Applied Soft Computing, 2016, 43: 415-423.

[4] 陶丽华, 马振楠, 史朋涛, 等. 基于 TSP 问题的动态蚁群遗传算法 [J]. 机械设计与制造, 2019 (12): 39.

附录 A 模型的代码实现

A.1 GATSP–matlab 源代码

```
clear;
w=20;g=100;d=19;%w为种群数，g代数,d维数
G(1:w,1:d)=0;%初始化空间
for i=1:w%初始化
c=randperm(d);
for t=1:20
flag=0;
for t1=1:d-1
for t2=t1+1:d
cl=c;
cl(t1:t2)=cl(t2:-1:t1);
if distan(cl)<distan(c)
c=cl;
flag=1;
end
end
end
if flag==0
G(i,1:d)=c;break
end
end
end
for k=1:g %进入遗传循环
A=G;%预备交叉阵
c=randperm(w);%配对序列
%c=1:w;
for i=1:2:w %交叉
F1=ceil(rand*d);%交叉点1
F2=ceil(rand*d);%交叉点2
while(F1==F2)
F2=ceil(rand*d);
end
if(F1>F2)%交叉地址调序
tem=F1;
F1=F2;
F2=tem;
end
j=0;t=1;%计数标值
while(j~=d+F1-F2-1)%如果剩余基因没完全插入就继续
if(isempty(find(A(c(i),F1:F2)==G(c(i+1),t),1))) %目标基因于交换片段中都不同
j=j+1;
if j<F1 %前半段基因交换
```

```
A(c(i),j)=G(c(i+1),t);
A(c(i+1),t)= G(c(i),j);
else %后半段基因交换
A(c(i),j+F2-F1+1)=G(c(i+1),t);
A(c(i+1),t)=G(c(i),j+F2-F1+1);
end
end
t=t+1;
end
end
by=[];
while isempty(by)
by=find(rand(1,w)<0.3);%变异地址
end
B=G(by,1:d);%预备变异阵
for j=1:length(by)
bw=sort(ceil(rand(1,2)*d));%变异基因节点
B(j,bw(1))=G(j,bw(2));%单点基因交换
B(j,bw(2))=G(j,bw(1));
end
GG=[G;A;B];%GG为选择阵
clear A; clear B;%清除数据防止规格保存
m=size(G,1);%选择阵个体数
long(1:m)=0;%目标函数初始化
for i=1:m%计算函数
long(i)=distan(GG(i,:));
end
[slong,ind]=sort(long(1:m));%目标函数排序
for i=1:w%精英选择
G(i,:)=GG(ind(i),:);
end
clear GG;%清除数据防止规格保存
end
```

A.2 MGATSP–matlab 源代码

```
clear;
for pp=6:13 %6:13
for ppp=1:100
n=pp;w=20;g=100;d=19+n-1;%n为车数，w为种群数，g代数,d维数
G(1:w,1:d)=0;%初始化空间
for i=1:w%初始化
c=randperm(d);
for t=1:20
flag=0;
```

```
for t1=1:d-1
for t2=t1+1:d
cl=c;
cl(t1:t2)=cl(t2:-1:t1);
if price(cl)<price(c)
c=cl;
flag=1;
end
end
end
if flag==0
G(i,1:d)=c;break
end
end
end
for k=1:g %进入遗传循环
A=G;%预备交叉阵
c=randperm(w);%配对序列
%c=1:w;
for i=1:2:w %交叉
F1=ceil(rand*d);%交叉点1
F2=ceil(rand*d);%交叉点2
while(F1==F2)
F2=ceil(rand*d);
end
if(F1>F2)%交叉地址调序
tem=F1;
F1=F2;
F2=tem;
end
j=0;t=1;%计数标值
while(j~=d+F1-F2-1)%如果剩余基因没完全插入就继续
if(isempty(find(A(c(i),F1:F2)==G(c(i+1),t),1))) %目标基因于交换片段中都不同
j=j+1;
if j<F1 %前半段基因交换
A(c(i),j)=G(c(i+1),t);
A(c(i+1),t)= G(c(i),j);
else %后半段基因交换
A(c(i),j+F2-F1+1)=G(c(i+1),t);
A(c(i+1),t)=G(c(i),j+F2-F1+1);
end
end
t=t+1;
end
end
by=[];
while isempty(by)
```

```
by=find(rand(1,w)<0.3);%变异地址
end
B=G(by,1:d);%预备变异阵
for j=1:length(by)
bw=sort(ceil(rand(1,2)*d));%变异基因节点
B(j,bw(1))=G(j,bw(2));%单点基因交换
B(j,bw(2))=G(j,bw(1));
end
GG=[G;A;B];%GG为选择阵
clear A; clear B;%清除数据防止规格保存
m=size(G,1);%选择阵个体数
long(1:m)=0;%目标函数初始化
for i=1:m%计算函数
long(i)=price(GG(i,:));
end
[slong,ind]=sort(long(1:m));%目标函数排序
for i=1:w%精英选择
G(i,:)=GG(ind(i),:);
end
clear GG;%清除数据防止规格保存
end
result(pp-5,ppp)=long(1);
XXX(pp-5,ppp,1:d)=G(1,1:d);
end
end
```

A.3 distan–matlab 源代码

```
function f=distan(X)
n=size(X,2);
a=[3 2
1  5
5  4
4  7
0  8
3  11
7  9
9  6
10 2
14 0
2  16
6  18
11 17
15 12
19 9
```

```
22 5
21 0
27 9
15 19];
f=sum(abs(a(X(1),:)-10));%距离值初始化
for i=1:n-1%计算距离和
f=f+sum(abs(a(X(i+1),:)-a(X(i),:)));
end
f=f+sum(abs(a(X(n),:)-10));%头尾固定
```

附录 B 数据可视化的实现

B.1 第一问画图–python 源代码

```
from pylab import *
mpl.rcParams['font.sans-serif'] = ['SimHei']

dict = {"1":[3,2], "2":[1,5], "3":[5,4],"4":[4,7], "5":[0,8],"6":[3,11],"7":[7,9],
"8":[9,6],"9":[10,2], "10":[14,0],"11":[2,16], "12":[6,18],"13":[11,17],"14":[15,12],
"15":[19,9],"16":[22,5], "17":[21,0],"18":[27,9], "19":[15,19],"20":[10,10],}
x_axis_data = []
y_axis_data = []
road = [20,8,3,4,5,2,1,9,10,17,16,18,15,14,19,13,12,11,6,7,20]
x_tem = []
y_tem = []
for i in range(len(road)):
x = str(road[i])
print(dict[x])
x_axis_data.append(dict[x][0])
y_axis_data.append(dict[x][1])

try:
x_tem.append(dict[str(road[i+1])][0])
y_tem.append(dict[str(road[i])][1])
except:
pass

x_ = []
y_ = []

for i in range(len(x_tem)):
x_.append(x_axis_data[i])
y_.append(y_axis_data[i])
x_.append(x_tem[i])
y_.append(y_tem[i])
```

```
x_.append(x_axis_data[i+1])
y_.append(y_axis_data[i+1])

plt.plot(x_, y_, 'ro-', color='#4169E1', alpha=0.8, label='路径')

for x, y in zip(x_axis_data, y_axis_data):
plt.text(x, y+0.3, '({},{})'.format(x,y), ha='center', va='bottom', fontsize=10.5)

# plt.legend(loc="road")
plt.xlabel('X轴/km')
plt.ylabel('Y轴/km')

# plt.show()
plt.savefig('demo.jpg') # 保存该图片
```

B.2 第二问画图–python 源代码

```
from pylab import *
mpl.rcParams['font.sans-serif'] = ['SimHei']
# import matplotlib.pyplot as plt
import numpy
import matplotlib.colors as colors
import matplotlib.cm as cmx

dicts = {"1":[3,2], "2":[1,5], "3":[5,4],"4":[4,7], "5":[0,8],"6":[3,11],"7":[7,9],
"8":[9,6],"9":[10,2], "10":[14,0],"11":[2,16], "12":[6,18],"13":[11,17],"14":[15,12],
"15":[19,9],"16":[22,5], "17":[21,0],"18":[27,9], "19":[15,19],"0":[10,10],}
x_axis_data = []
y_axis_data = []

cars = [14,0,18,15,0,10,9,0,17,16,0,12,11,0,4,2,3,8,0,5,1,0,19,0,6,7,0,13]

c_ = []
x = [0]
for j in range(len(cars)):
x.append(cars[j])
if cars[j]==0:
c_.append(x)
x = [0]

print(c_)
cmap = plt.cm.jet
cNorm = colors.Normalize(vmin=0, vmax=len(c_))
scalarMap = cmx.ScalarMappable(norm=cNorm, cmap=cmap)
```

```
for fff in range(len(c_)):
############
x_axis_data = []
y_axis_data = []
road = c_[fff]
x_tem = []
y_tem = []
for i in range(len(road)):
x = str(road[i])
x_axis_data.append(dicts[x][0])
y_axis_data.append(dicts[x][1])

try:
x_tem.append(dicts[str(road[i + 1])][0])
y_tem.append(dicts[str(road[i])][1])
except:
pass

x_ = []
y_ = []

for i in range(len(x_tem)):
x_.append(x_axis_data[i])
y_.append(y_axis_data[i])
x_.append(x_tem[i])
y_.append(y_tem[i])



colorVal = scalarMap.to_rgba(fff)
x_.append(x_axis_data[i + 1])
y_.append(y_axis_data[i + 1])
plt.plot(x_, y_, 'o-', alpha=0.8)
for i in range(0,len(x_)-1):
plt.arrow(x_[i], y_[i], x_[i+1] - x_[i], y_[i+1] - y_[i],
length_includes_head=True, head_width=0.3, lw=2,
color=colorVal)

for x, y in zip(x_axis_data, y_axis_data):
plt.text(x, y + 0.3, '({},{})'.format(x, y),)

plt.xlabel('X轴/km')
plt.ylabel('Y轴/km')

# plt.show()
plt.savefig('demo.jpg') # 保存该图片
```