游乐园客流疏导方案设计

**及酒店预定数量预测数学模型的建立**

组员：

**摘要**

本文研究了游乐场客流疏导方案的设计问题，并进行了2016年1至3月酒店预定数量的预测。通过仿真模拟出游客时间利用效率，设计基于游客偏好系数的疏导方案，并通过遗传算法进行优化；通过对2015年酒店全年数据的分析预测2016年1至3月预定数量。

针对问题1，我们建立了游客这一对象，并将之组成组团，更好的符合了实际情况。同时，我们建立了决策算法，对在一定程度上满足了游客意愿的情况下，对游客组团游览提供建议，并为后期对决策算法的修正提供了接口。我们建立了A~J10个游乐项目按一定规则对组团进行托管、作用、交换，最大限度的模拟游客在游乐园中的行动。我们将以上所有决策算法、游乐项目等组合，建立了对该游乐园的完整的仿真模型，并在其中嵌入了各种测试、统计接口，收集了大量的数据。

在仿真的基础上，我们采用遗传算法，通过已有的决策算法修正接口，对决策算法进行仿真、评估、淘汰、变异扩增、再仿真、再评估……。我们通过这一过程，对决策算法进行了多次优化，获取到了一个较优解。

针对问题2，对于酒店预定量的预测，首先图像化数据，适当的将数据进行分类，得到与季节有关的入住关系，然后利用数据的数字特征，考虑到影响预定的各种因素，即机动因素，计划性因素，和节日性因素。找到入住与预定间的关系，而后对于酒店预定情况进行预测。得到线性函数关系，并考虑到其波动性，因而数学模型结果为带状分布。且通过研究发现一二三月份因受年终，春节等因素影响，酒店预定量达到低谷。而其他各月份的预约结果为在一定区间内的震荡分布。

【关键词】**时间利用效率 偏好系数 系统仿真 遗传算法 迭代优化 matlab**

1. **问题背景**

**游乐园客流疏导问题**：在大型游乐场中，园区面积太大、游乐设施数量众多导致游客在游玩过程中会出现走弯路、走回头路、游览顺序选择不恰当等问题。现有的游乐场会在入口处提供纸质版地图，但内容不翔实、不直观，很多仅是简单的位置信息，缺乏路径指引的作用。使游客即便找到了想玩的项目也会因为设施外排队人数过多、等待时间过长而不得不为节约时间而选择其他的项目，造成时间的浪费和园区拥堵。如何避免某些时刻个别游乐设施过于拥挤，而其他设施游客寥寥无几的情况发生，减少游客在寻找和等待上的游玩成本，是一个需要思考、解决的问题。

**游客时间利用效率**：游客的等待时间和总时间的比，是一个能直接体现疏导方案的成效的参数。游客时间利用效率计算公式如下：

游客时间利用效率=

**游客偏好系数**：每个人对游乐场中每个项目都有自己的喜好排序，根据游客的喜好排序，以及给定的一个参数，可计算出其偏好系数。每个游客的偏好系数互不相关。

**数据共享优化决策算法**：基于遗传算法的改进，

**排队论**：或称随机服务系统理论，是通过对服务对象到来及服务时间的统计研究，得出这些数量指标（等待时间、排队长度、忙期长短等）的统计规律，然后根据这些规律来改进服务系统的结构或重新组织被服务对象。

1. **问题重述**

Youth游乐园即将盛大开园，作为本市建有最多过山车的游乐园，受到了青少年的热捧。预计届时园区将迎来每天1万的大客流。如何根据客流情况，及时分流人群，为顾客提供游园线路引导，保障游客的游园体验显得尤为重要。（不要抄原话）

试就园区的整体规划，建立数学模型分析研究下面的问题：

1. 根据Youth乐园的规划图（共设A-J 共10个项目点，游客可沿着图中标出的线路往返下个游乐项目）在保障每位游客体验游乐设施的前提下，建立对每个游乐项目的等候游客进行游览提醒和疏导的模型，以达到游园体验最优。
2. 皇冠假日酒店是游乐园内的酒店，目前已开业，为有需要的游客提供住宿便利。请根据该酒店历史预订数据信息,综合考虑影响房间预定量的主要因素(比如季节,工作日/周末,法定假日,暑期等)建立数学模型。并根据酒店2015年全年预定数据, 预测2016年1月至3月每天预定房间数.
3. **问题分析**

**3.1 问题1的分析**

针对问题1,由生活经验可知，在游乐场游玩时，游乐场提供纸质导览图，即有各个项目大致方位、项目内容等固有信息，游客无法得知各个项目排队人数、等候时间等实时信息，且大多数游客在无导览的情况下盲目随从众人，导致一些项目出现拥挤、等候时间过长的现象，即自由散漫的游览方式的游客时间利用效率是很低的。

为了改进这种状况，我们提出基于游客偏好系数的疏导方案的设计，此方案将综合游乐场的实时数据，并创新性地加入游客偏爱系数，为游客提供实时且个性化的推荐游览方案。

并使用遗传算法对此疏导方案进行迭代优化，以使在一定程度上满足有课偏好的前提下，获得使游客时间利用率最大的最优方案。

问题1的整体思路如图1所示。

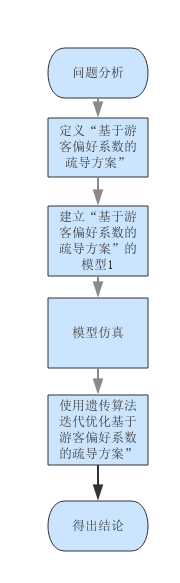


图 1 问题1的思路流程图

**3.2 问题2的分析**

针对问题2，题目给出了2015年整年的房间预定数量，一共6150个数据，大量的数据需要合适的方法来处理。由于酒店预订量与很多因素有关，包括节假日、季节、工作日与周末等，直接寻找客房预订关系，似乎并不能直观的得出有效的结论。而游客的入住行为，受假日，季节，工作日，影响较为显著。并且预定行为受游客入住行为的影响。并且根据一般经验可知，预定行为一般与入住行为呈现三种关系：

1. 第一种为其中机动性因素表现为预定于入住时间相差无几的情况。
2. 在非节假日的情况的入住，称之为计划性入住，一般预定日期与入住日期相差一周至半个月。
3. 最后是假日性因素，基于游客普遍会考虑到节假日，客房比较难以预定，因而会出现提前半个月至一个月的预定情况。

问题整体思路如图2所示：

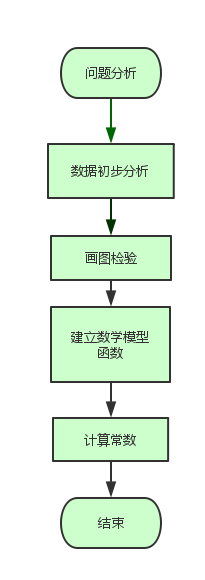


图 2 问题2的思路流程图

1. **符号说明与模型假设**

**4.1符号说明**

问题1符号

|  |  |  |
| --- | --- | --- |
| 序号 | 符号 | 说明 |
| 1 | K1 K2 K3 K4 | 决策算法参数 |
| 2 | n | 总游戏时间 |
| 3 | **QX** | 总入园时间 |
| 4 | **Ngroup** | 组内人数 |

问题2符号

|  |  |  |
| --- | --- | --- |
| 序号 | 符号 | 说明 |
| 1 | α | 机动性因素 |
| 2 | β | 计划性因素 |
| 3 | **γ** | 节日性因素 |

**4.2模型假设**

1. 假设每一个游乐设施在游乐场开始营业的同时开始运行，且不考虑设备损坏造成暂停运行的情况。

2.假设所有游客都以2-7人组团存在，入园后行动共同行动，且都参与决策。

3.假设每一个游客对于各个项目的喜好程度不同，实际运用时由入园时的调查获得。此处为简化模型，游客喜好随机生成且所有项目受欢迎程度相同。

4.假设每一个游客在结束一个项目后接收系统通过分析实时数据得出的推荐游览方案，且完全按系统方案执行。

5.假设各游乐项目之间道路及各项目前排队区的承载能力足够大。

6.游乐园8:00开放，18:00关闭。

1. **模型的建立与求解**

**5.1问题1中的模型建立与求解**

每个游客有既定的偏好顺序。系统将为每组游客提供，由组内每个成员喜好、各项目距离、园内设施实时使用情况综合考量，得出的最优游览建议。我们拟建立以下仿真模型来分析值，并对疏导方案进行遗传优化。

问题1算法流程图如图3所示

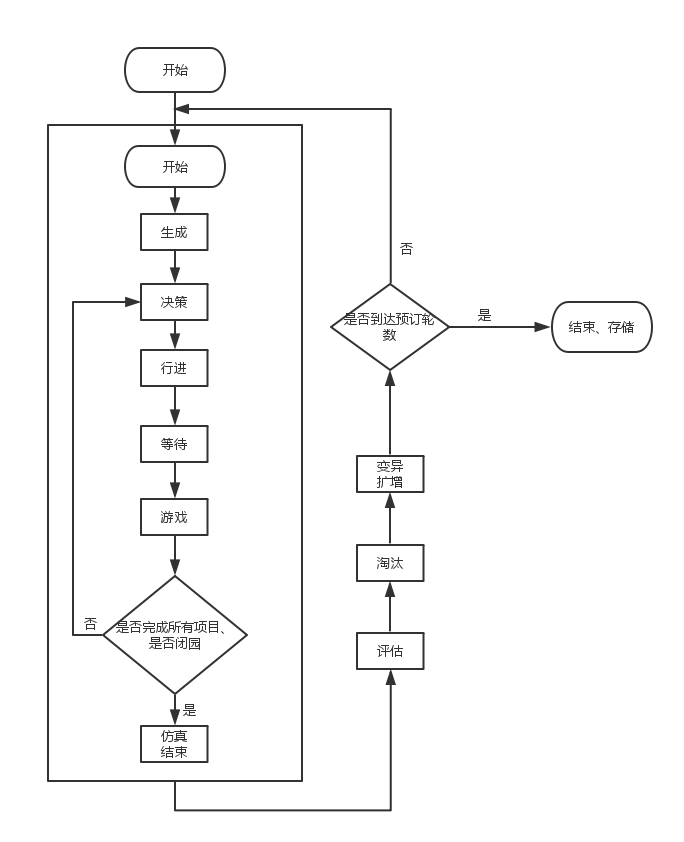
****

图 3 算法流程图

1. **仿真模型的建立**

流程图如下：

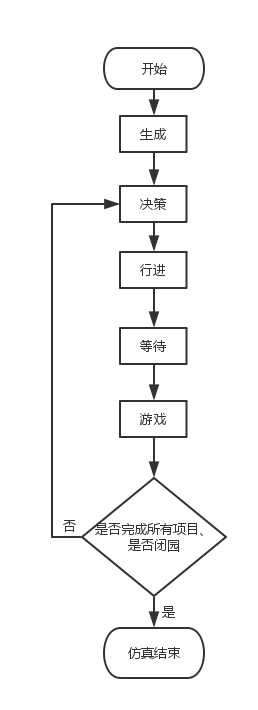
****

图 4 仿真算法流程图

* **生成：**

我们将根据以下模型，每15秒向该游乐园添加一定数量的游客。

经查阅相关论文及资料，我们建立如下每15s进入人数与时间关系。

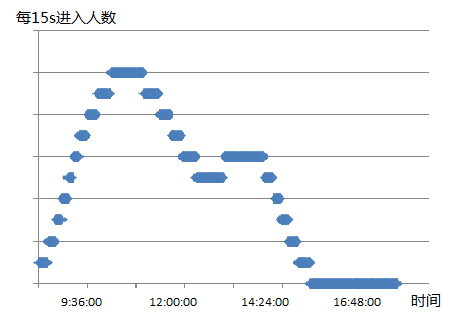


图 5

考虑到进入人数的无规律性，我们将人数进行随机化修正，并将游客分为2-7人组团。各组团人数。经随机化修正后每一次模拟数据都不相同，其中一次模拟中，每15s进入人数与时间关系如下

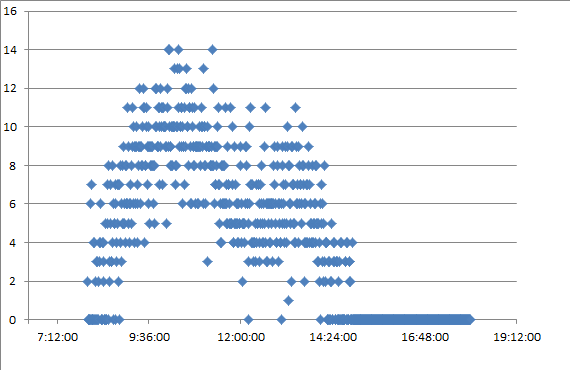


图 6

当天共10000人进入游乐园

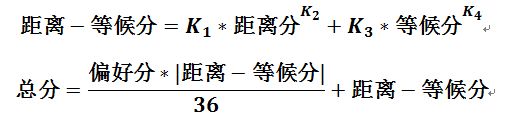
* **决策**

所有决策均以组为单位。通过对未游玩项目打分并比较各项目分数的方式进行决策。总分由偏好分、距离分、等候分组成。

每位游客偏好分由每位游客对该项目排序，计为0~9分。每组偏好分为每个人偏好分的平均数。

距离分由该组现在所在位置与被打分项目的距离决定。

等候分由被打分项目当前排队人数决定。总分由下式决定：



其中**586291294774245244**均为参数，也是下一步我们优化的对象。

获得总分后，得分最高的项目即为该组接下来的目标。

* **项目托管**

决策后，各组的行为即由各项目托管至完成该项目游览。

**行进：**

托管的一环，查阅资料获知，中国人步速为1.5m/s。考虑到游客处于游玩状态，采用1m/s为步数。经过一段时间，当某一组走到目标项目后，该组由“等待”托管。

**等待：**

各组依到达先后顺序排队。每次被等待游玩项目完成一轮后即有若干组按顺序进入“游戏”托管，使游戏人数最多，但不超过该项目容量，且没有被分开的组。

**游戏：**

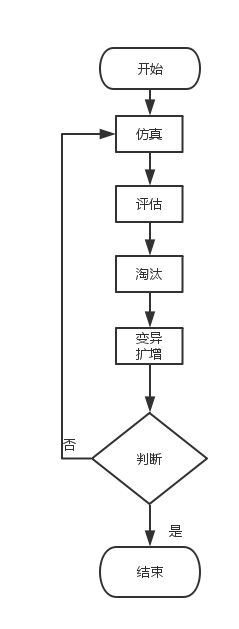
各组进入后即开始计时，达到该项目持续时间后，即释放所托管各组。

* **判断**

对各组游玩情况和当前时间进行评估。若某组已完成所有项目，或已闭园，即将该组结束仿真并存储其数据。否则，其进行下一次决策。

**模型的计算及迭代优化**

为检验及优化模型，我们对模型按所示模型进行了评估，并按遗传算法进行了优化。

****

* **开始**

初始化60个决策算法。决策算法各参数由随机函数给出。

* **仿真**

依照（1）中模型，使用60个决策算法，分别进行仿真。

* **评估**

取依据各决策算法仿真的结果，分别求各项目游客时间利用率的均值。

* **淘汰**

依据各项目时间利用率均值对各决策算法进行从高到低排序，淘汰排名靠后的45个公式，取前15个公式进入下一环节。

* **变异扩增；**

将每个决策算法参数按下式变异，各获得4个新决策算法，变异后为K­i1

506155518811694625

其中Ri由随机数给出，Ri~N（0.1,0）

* **判断**

判断是否迭代所给次数，若达到则结束存储，若未达到即再次仿真。

**5.2问题2中的模型建立与求解**

**1）模型研究对象**

在对不同的时间段的入住关系和预定关系进行分析时，主要分为节假日与工作日两部分考虑其对于入住，预订情况的影响。由于入住受特殊时间段（节假日）的影响比较显著，因而先建立如住房关系曲线。预定是超前于入住的一种行为，并且其可由经验值认为分为三种情况：

1. 入住时间与预定时间相差无几，视其为机动的因素；

2. 预定时间要提前半周至一周，可视其为一种带有计划性的操作；

3. 考虑到：节假日的旅店的房源紧俏。并且认为：有需要住店需要的游客，为了避免夜晚没有房间住，会提前较长时间进行预订。即一周即以上的可能性，视其为一种节假日预定因素。

因此我们根据2015年的入住关系图，初步拟合2016年的入住关系。然后找出入住与预定之间存在的函数关系，建立数学模型。

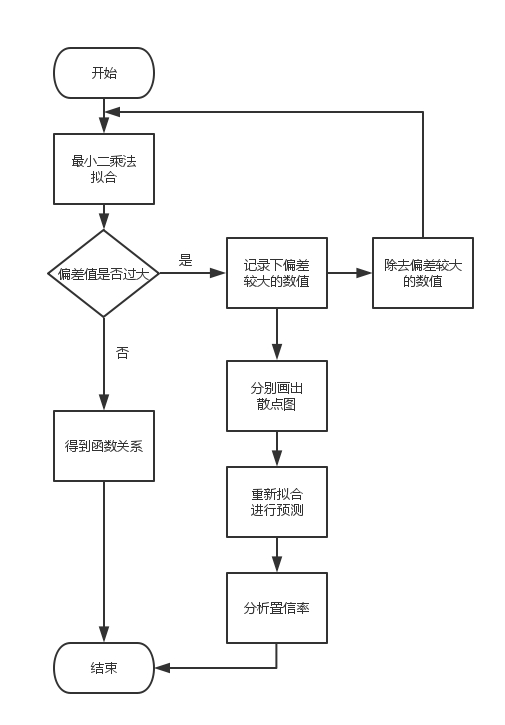


图 2 入住堆积图

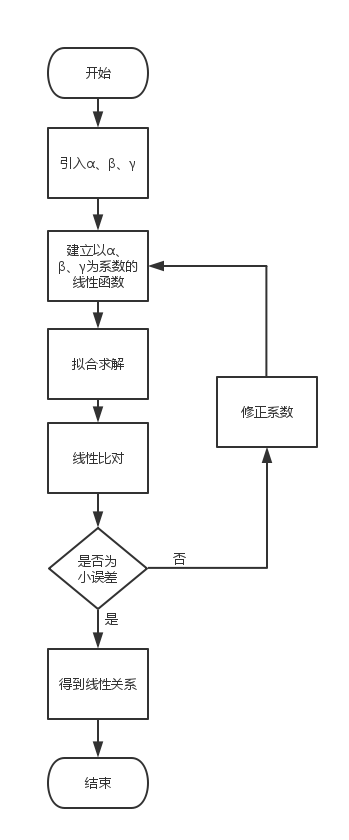


图 3 入住关系图

**2）模型的选择**

**1.**为了消除入住过程中每一天的不稳定性的偏差，采用总和计数的图表，进行数据的拟合。并且考虑到这种增长关系的线性性。因而对图像进行线性函数。且经分析，图像应改为一过原点的曲线，因此可以写为y=kx的形式。

**2.**对于入住与预定的函数关系f（α，β，γ，g入住）可基本表示为f=α\*g入住1+β\*g入住2+γ\*g入住3的形式，通过对于函数的拟合进行求解。

模型的建立

**3）模型的求解**

* **数据拟合**

直接画出入住-日期的散点图：

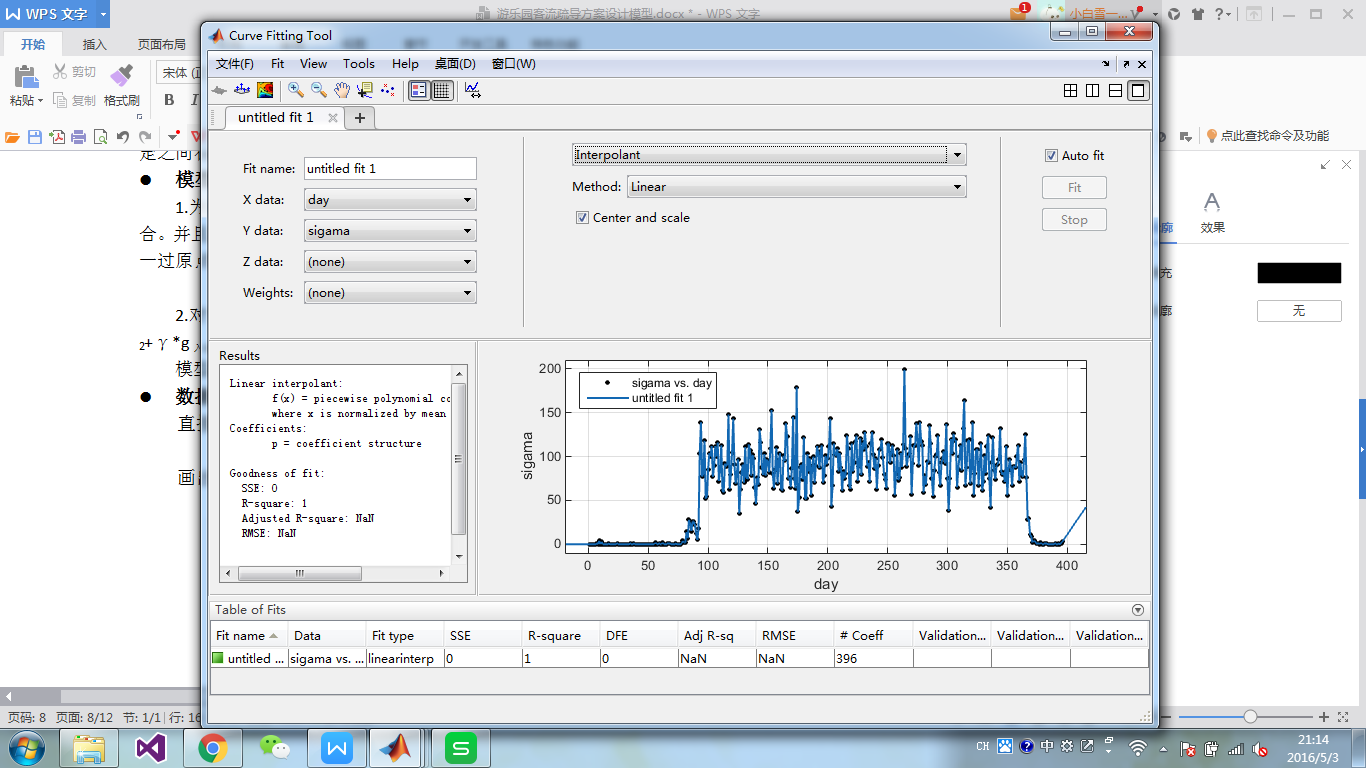


图 5

可以发现在前90几天，数据几乎没有，并且散点图中无法清晰的反应函数关系，因而想到了入住的堆积图，用以消除误差；

画出入住函数的堆积图：

图 6

发现数据中15年的一月至三月呈直线状，而16年一月的入住后期也呈现出水平直线状，拟合出来的函数会出现较大的误差。

因而将数据进行处理：将一至三月的数据，另外记录，找寻函数关系，而四月至十二月进行函数拟合：（拟合结果如图）

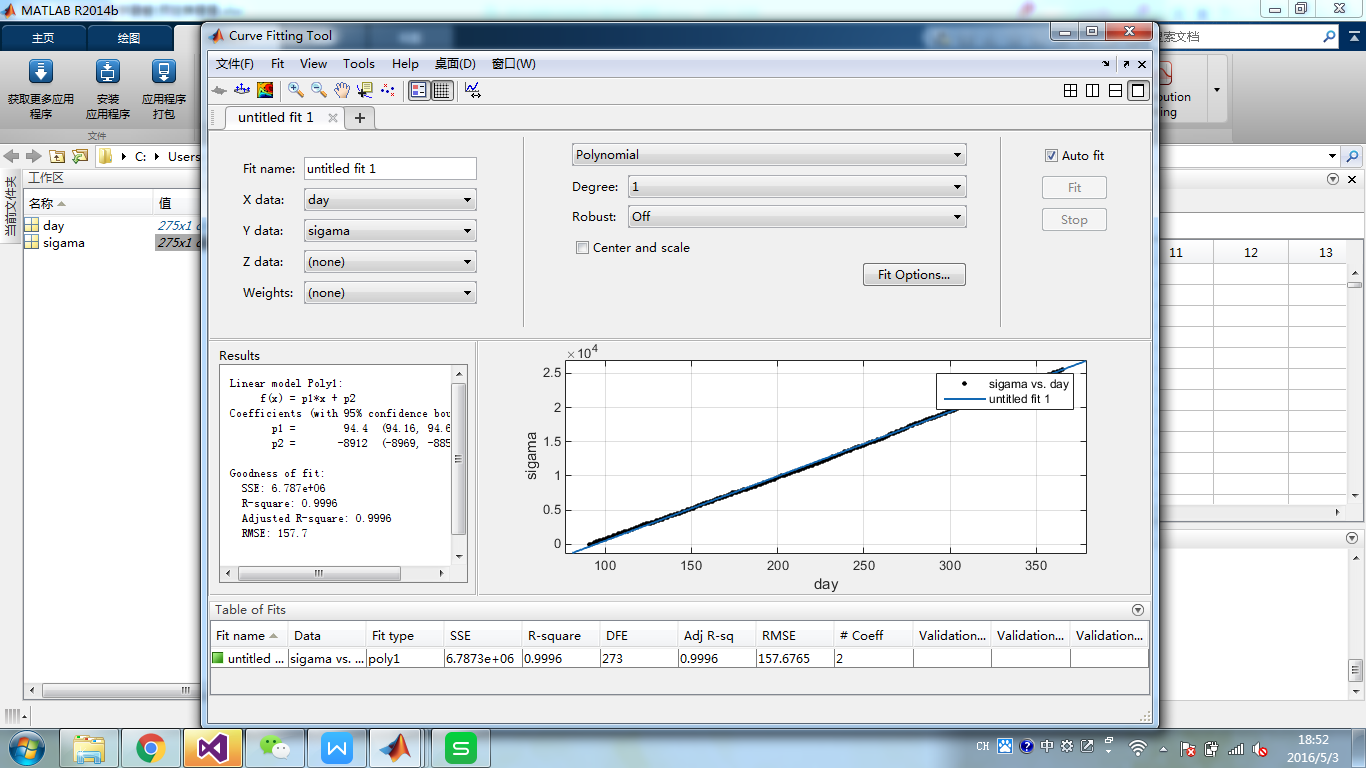


图 7

得到拟合参数如下：

Linear model Poly1:

f(x) = p1\*x + p2

Coefficients (with 95% confidence bounds):

p1 = 94.4 (94.16, 94.64)

p2 = -8912 (-8969, -8855)

Goodness of fit:

SSE: 6.787e+06

R-square: 0.9996

Adjusted R-square: 0.9996

RMSE: 157.7

观察数据，发现1、2、3月的入住情况并无特殊的函数关系，且量少、数据小，几乎可以认为其预定情况为处在某一区间的随机分布。因而直接研究其预订模型，直接预测2016年1、2、3月的预定情况：

* **一月份预定预测：**

根据一月份的数据，画出散点图：

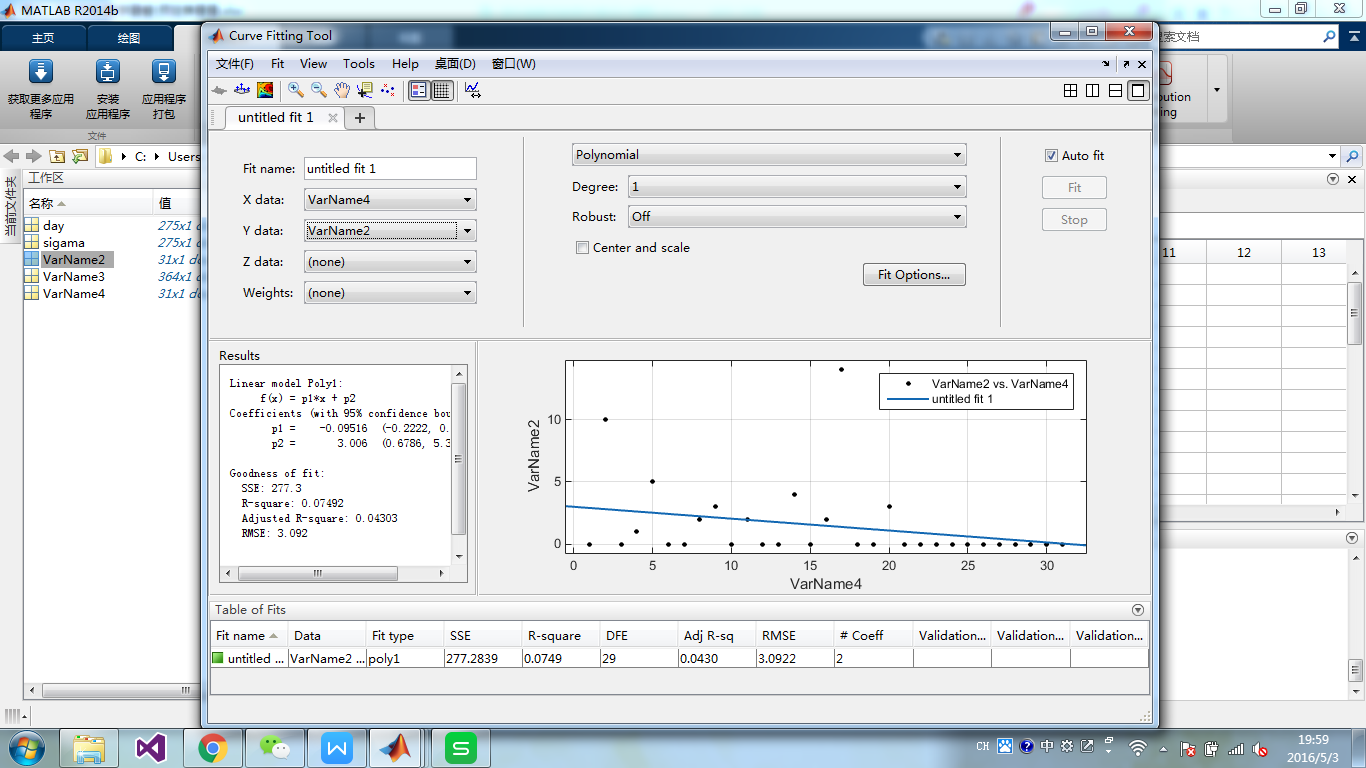


图 8

函数关系：f(x)=-0.95+3

E=1.5 S=3.1 E±S=[0,5] 落在区间上的概率：93.5%

(考虑到房间数为整数，因而区间四舍五入取整数)

* **二月份预定预测：**

根据二月份数据，画出散点图：

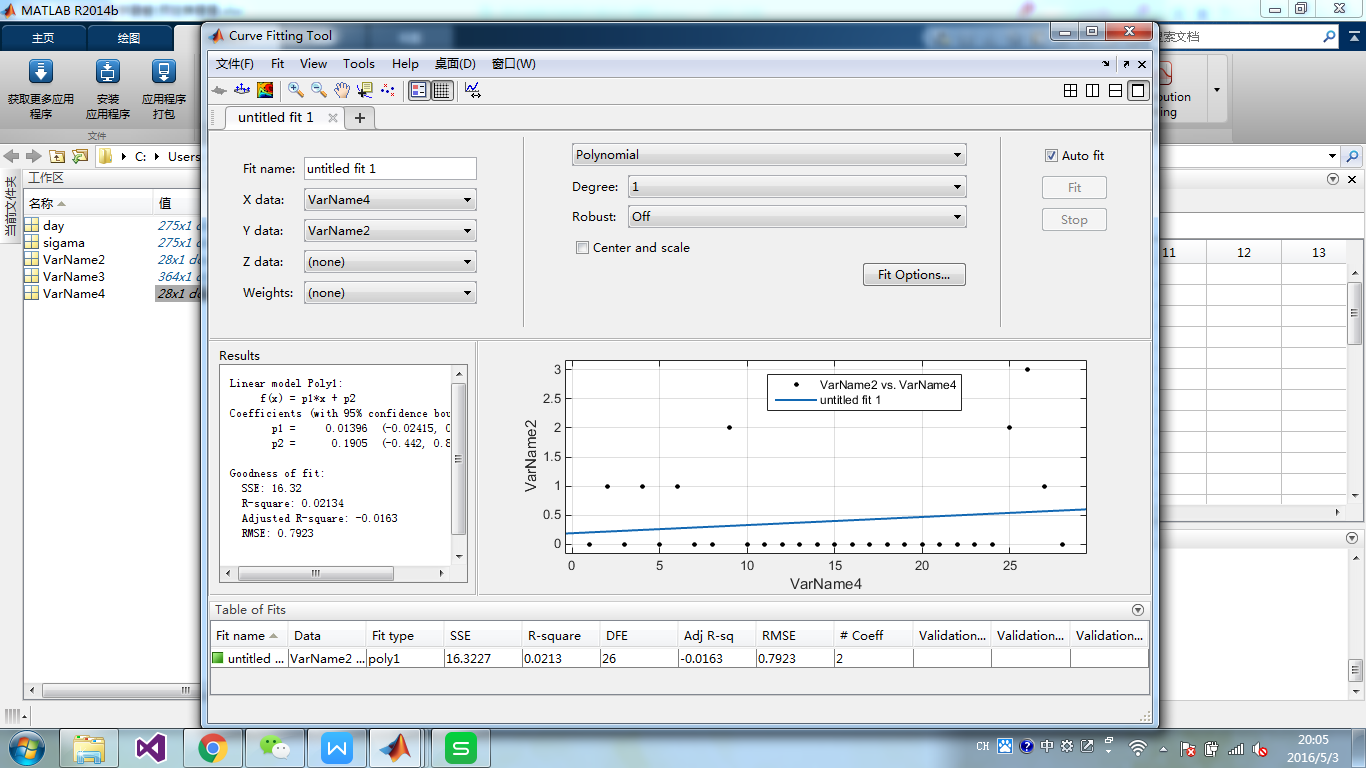


图 9

函数几乎取在0-1附近，因而只分析其特征值，列出区间

E=0.4 S=0.78 E±S=[0,1] 落在区间上的个概率：92.85%

* **三月份预定预测：**

根据二月份数据，画出散点图：

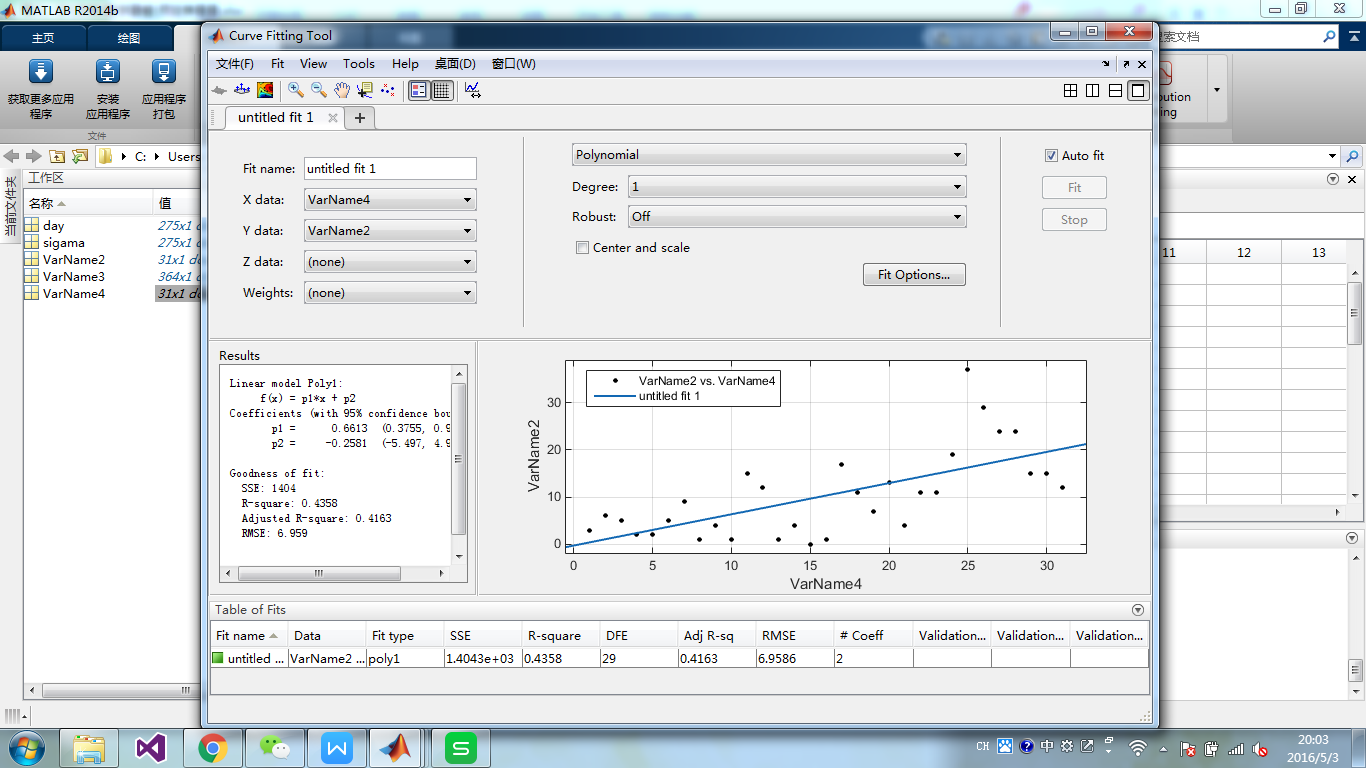


图 10

函数关系：F(x)=0.66x-0.26

E=10 S=9 E±S=[1,19] 落在区间上的概率：84%

由于三月份的数据波动幅度越来越大，因而造成的方差也比较大，所以需要结合区间和函数关系，综合估计其预定情况。也就是说，可以视三月份的预定情况，为一三角形区域。

* **普通情况下的预测模型：**

预测模型数学函数f（α，β，γ，g入住）可基本表示为f=α\*g入住1+β\*g入住2+γ\*g入住3

由于所有的入住行为都需要预约（其中直接入住的情况为，提前0天预约）。所以预约的均值应与入住的均值相同，而节假日因素，影响的是预测值的方差。预测函数又可以写为：

E(入住)±μ\*S（入住）的形式。即如三月份数据的形式，其为一根f函数外包罗的长方形面积：如图

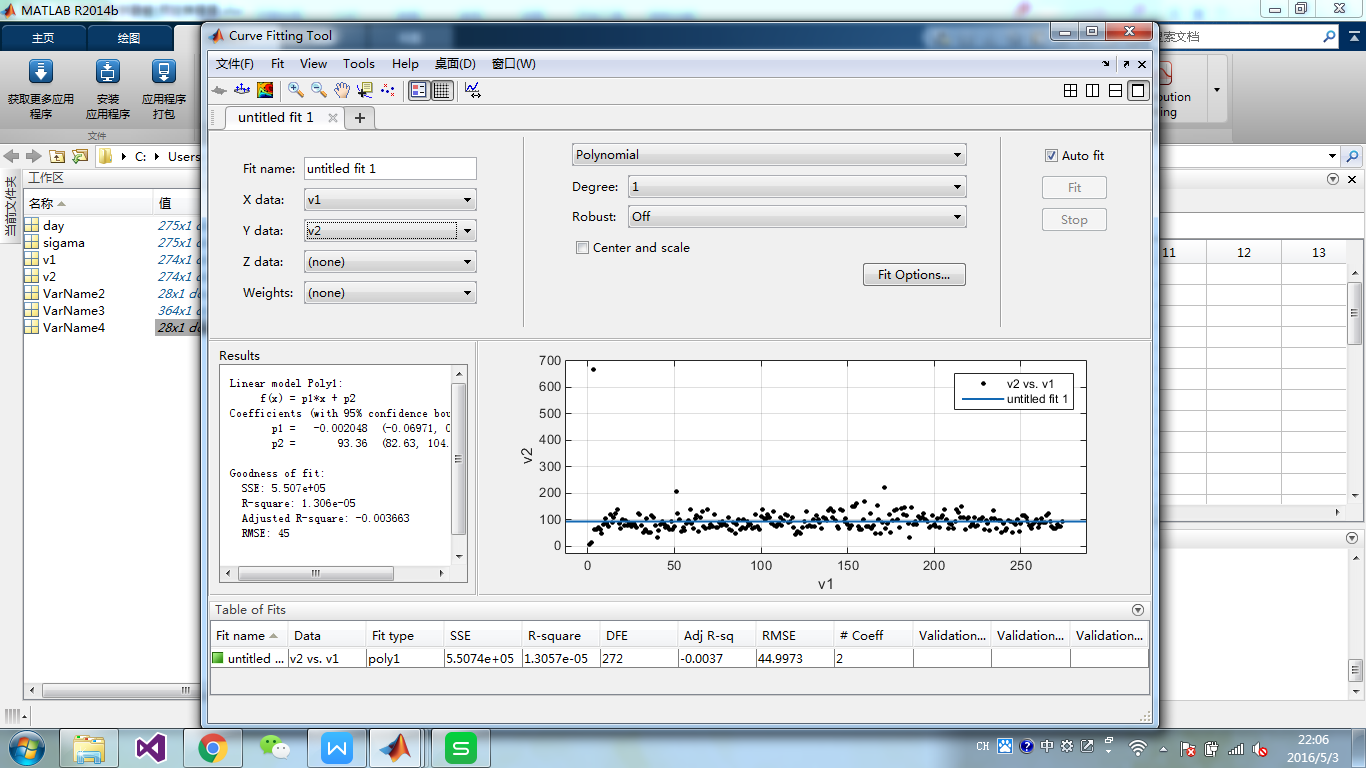


图 11

**六、**  **模型结果分析**

**6.1 问题1结果分析**

依据模型仿真，下午入园的游客将无法完成所有项目，符合事实情况。

由于本组所使用笔记本电脑计算能力所限，我们只进行了50轮优化，取每次优化后，时间利用率最高的算法，取平均时间利用率列表如下：

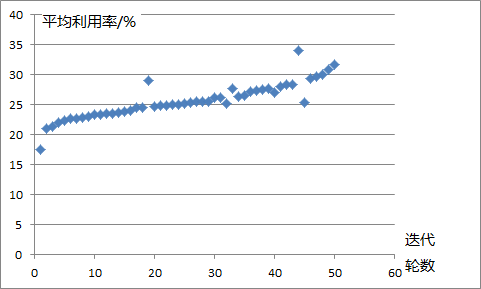
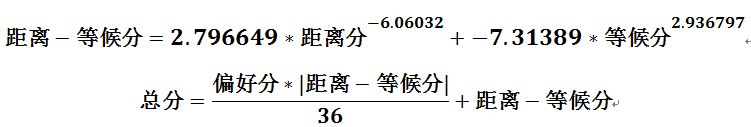


图 12

所获得相对最优算法如下：

****

由表中可见，经过迭代其时间利用率逐渐提高，所获得算法逐渐趋向最优，若能使用更大运算能力的计算机，迭代更多轮数，我们将可获得更优化的解决方案。

**6.2 问题2结果分析**

通过对数据图像化处理分析，先将有明显差异的数据分类，然后根据各类数据的特征，直观的选择拟合方式，找出相应的线性表达式。而后根据数据的数字特征，即平均值和标准差，构造函数浮动区间。通过一线一面，将预定情况表现在函数图像一定的区域内。虽然得到的是一个区间量，但是预测的精准度并不会因此而下降。比起每天的预定数值预测，更有操作性。所得到的预测结果落在一个范围值内，有效的避免了数据的波动对于预测结果的影响。虽然没有考虑到偶然发生的小概率事件对于预测结果出现的较大范围影响，但是根据历史数据，得到的数据范围，足够以预测普通情况下，房客随机的预定行为。因而模型做出的结果预测是可行的，有效的。

**八、参考文献**

[1] 张聪慧.杨明.ZHANG Cong-hui.YANG Ming 贝叶斯动态模型在煤矿事故预测中的应用研究 [期刊论文]-中国安全生产科学技术 2014(z1)

[2] 徐继承.王婷.黄水平..XU Jicheng.WANG Ting.HUANG Shuiping 基于后验预测分布的贝叶斯 模型评价及其在霍乱传染数据中的应用 [期刊论文]-郑州大学学报（医学版） 2015(2)

[3] 郑天翔 基于单步协调控制的大型游乐场游客智能导览系统的建模与仿真 [期刊论文]-暨南学报（哲学社会科学版） 2015(10)

[4] 2015年全国大学生数学建模竞赛优秀论文（“月上柳梢头，人约黄昏后”数学模型的建立）

**附录：部分代码**

仿真

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace 游乐园模拟

{

class Park

{

//参量确定

private PeopleIn TodayPeopel = new PeopleIn();

public facility[] ParkFacility = new facility[11];

public decision DecisionHere = new decision();

//建立类

public Park()

{

……

}

//15时间进程

public void TimeRun\_15s(int Time\_15s)

{

//进入决策区

//入园

DecisionHere.DecisionIn(TodayPeopel.fIn(Time\_15s));

//为所有group加计时间

//决策区

for (group current = DecisionHere.LineOfDecision.head; current != null; current = current.next)

{

current.counter++;

current.CounterPerFacility++;

}

//facility 内部

for (int i = 0; i < 10; i++)

{

//等待区

for (group current = ParkFacility[i].WaitingLine.head; current != null; current = current.next)

{

current.counter++;

current.CounterPerFacility++;

}

for (group current = ParkFacility[i].PlayingLine.head; current != null; current = current.next) //游玩区

{

current.counter++;

current.CounterPerFacility++;

current.counterPlaying++;

}

//行进区

for (group current = ParkFacility[i].ApprochingLine.head; current != null; current = current.next)

{

current.counter++;

current.CounterPerFacility++;

}

}

for (int i = 0; i < 10; i++)//结束游戏的

{

if (ParkFacility[i].PlayingLine.head != null)

{

if (ParkFacility[i].PlayingLine.head.CounterPerFacility == ParkFacility[i].LastTime)

{

DecisionHere.DecisionIn(ParkFacility[i].PlayingLine);

}

}

}

//决策

DecisionHere.MakeDecision(ParkFacility);

ParkFacility[10].ToApproachingLine(DecisionHere.LineTo[10]); //特定项目处理

for (int i = 0; i < 10; i++)

{

ParkFacility[i].ToApproachingLine(DecisionHere.LineTo[i]);

ParkFacility[i].ToWaitingLine(ParkFacility[i].ApprochingLine);

ParkFacility[i].ToPlayingLine(ParkFacility[i].WaitingLine);

}

}

}

}

迭代优化

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace 游乐园模拟

{

public partial class Form1 : Form

{

//初始化参量

private Park youth = new Park();

private Random OutRandom = new Random();

private randgaussian OutRandGaussian;

//数据导入excel

private bool DataSetToExcel(bool isShowExcle, int[] a)

{

//建立Excel对象

Microsoft.Office.Interop.Excel.Application excel = new Microsoft.Office.Interop.Excel.Application();

excel.Application.Workbooks.Add(true);

excel.Visible = isShowExcle;//是否打开该Excel文件

//填充数据

int j = 1;

int k = 1;

group current = youth.ParkFacility[10].ApprochingLine.head;

for (int i = 0; current != null; i++, current = current.next)

{

if (true)

{

excel.Cells[j++, k] = current.n;

if (false)

{

k++;

j = 1;

for (int n = 0; n < 10; n++)

{

excel.Cells[j, k + n] = current.Gr[0].preference[n];

}

j++;

}

}

}

return true;

}

public Form1()

{

InitializeComponent();

}

//优化对象

const int Larger=50;

const int Smaller=5;

private decision[] DecisionTesting = new decision[Larger];

private decision[] DecisionTested = new decision[Smaller];

//繁衍

private decision[] produce(decision[] seed, randgaussian InRand)

{

//待输出区

decision[] result = new decision[Larger];

for (int i = 0; i < Larger; i++)

{

result[i] = new decision();

}

for (int i = 0; i < Smaller; i++)

{

//使用第i个种子

for (int j = 0; j < (Larger/Smaller); j++)

{

double K1 = InRand.rand() + seed[i].K1;

double K2 = InRand.rand() + seed[i].K2;

double K3 = InRand.rand() + seed[i].K3;

double K4 = InRand.rand() + seed[i].K4;

result[(Larger / Smaller) \* i + j].ChangeK(K1, K2, K3, K4);

}

}

return result;

}

//优选

private decision[] Optimization(decision[] DesiotionTesting)

{

decision[] result = new decision[Smaller];

for (int i = 0; i < Smaller; i++)

{

for (int j = Larger-2 - i; j >= 0; j--)

{

if (DecisionTesting[i].ScoreDecision < DecisionTesting[i + 1].ScoreDecision)

{

decision room = DecisionTesting[i];

DecisionTesting[i] = DecisionTesting[i + 1];

DecisionTesting[i + 1] = room;

}

}

result[i] = DecisionTesting[i];

}

return result;

}

private void btBetter\_Click(object sender, EventArgs e)

{

for (int i = 0; i < Smaller; i++)

{

DecisionTested[i] = youth.DecisionHere;

}

//建立Excel对象

Microsoft.Office.Interop.Excel.Application excel = new Microsoft.Office.Interop.Excel.Application();

excel.Application.Workbooks.Add(true);

excel.Visible = true;//是否打开该Excel文件

//填充位确定

int Xi = 1;

int Yi = 1;

//仿真

for (int RevolutionI = 0; RevolutionI < 10; RevolutionI++)

{

if (RevolutionI == 0 || RevolutionI == 1)

{

OutRandGaussian = new randgaussian((RevolutionI + 1) \* OutRandom.Next(-987, 99999), OutRandom.Next(-988, RevolutionI), 3, 0);

}

else

{

OutRandGaussian = new randgaussian((RevolutionI + 1) \* OutRandom.Next(-987, 99999), OutRandom.Next(-988, RevolutionI), 0.1, 0);

}

//第RevolutionI次迭代

//产生Larger个待仿真数据

DecisionTesting = produce(DecisionTested, OutRandGaussian);

for (int TestI = 0; TestI < Larger; TestI++)

{

//第TestI个decision仿真

youth = new Park();

youth.DecisionHere = DecisionTesting[TestI];

for (int Time\_15 = 0; Time\_15 < 2400; Time\_15++)

{

youth.TimeRun\_15s(Time\_15);

}

//第TestI个decision评分

double FuckI = 0;

for (group current = youth.ParkFacility[10].ApprochingLine.head; current!=null; current = current.next)

{

double before = youth.DecisionHere.ScoreDecision \* FuckI;

double ScoreCurrent = (int)current.counterPlaying + (int)current.counter;

youth.DecisionHere.ScoreDecision = (before + ScoreCurrent) / (FuckI+1.0);

FuckI += 1.0;

}

}

//优选

DecisionTested = Optimization(DecisionTesting);

//填充入excel

for (int PutI = 0; PutI < Smaller; PutI++)

{

//填充第PutI个数据

excel.Cells[Xi, Yi] = DecisionTested[PutI].K1;

excel.Cells[Xi, Yi + 1] = DecisionTested[PutI].K2;

excel.Cells[Xi, Yi + 2] = DecisionTested[PutI].K3;

excel.Cells[Xi, Yi + 3] = DecisionTested[PutI].K4;

excel.Cells[Xi, Yi + 4] = DecisionTested[PutI].ScoreDecision;

//填充位顺移

Xi++;

Yi = 1;

}

}

}

}

}