“拍照赚钱”任务定价方案

摘要

“拍照赚钱”任务定价的确立是一个企业和用户间的双向决策问题，如何制定合理的任务定价方案，在企业的成本和用户的利益之间做出最佳的权衡，是该模式正常运转的最关键因素。

针对问题一，我们对题目中所给数据进行聚类分析。为确定价格的影响因子，我们通过多元线性回归分析、傅立叶幅度灵敏性检验法等方法确定了价格与相关影响因素之间的相关性，并利用Logistic回归与GRNN神经网络的方法得到了改进的价格预测方法。

针对问题二，我们从基本的经济学原理与假设出发，基于等吸引力定价原则，提出了我们的任务定价模型。在此模型中，我们对价格折现系数做出了多种假设，并且对于静态决定的折现系数与动态博弈式决定的折现系数，分别设计出不同的模拟算法求解出相应的任务定价方案，其中，我们依据两层嵌套的遗传算法来优化动态模拟折现系数。

针对问题三，我们首先对于任务打包的效用改进做了理论论证，提出了不同的打包改进准则，并定性分析了它们的适用范围。随后，我们分析了相关量在空间上的统计分布情况，提出将打包策略优化问题转化为最大化任务量，最小化会员迁移量的问题，并在这一模型的基础上，给出了基于voronoi聚类和tsp最短路问题的近似求解算法。

针对问题四，我们首先利用问题二中的定价模型，给出了无打包条件下的任务定价。随后，我们借助第三问中的模拟打包模型，对任务定价再次优化，使得预测成交率得到提升。

关键词：线性规划，神经网络，遗传算法，聚类分析

一、问题重述

1.1 问题背景

随着互联网技术的发展，社会上出现了一种新的赚钱模式——“拍照赚钱”。用户注册成为某APP的会员并在APP上领取相应的拍照任务，完成后便可赚取酬金，同时这些照片为企业提供各种商业检查和信息搜集。与传统的市场调查方式相比，这种新型的调查模式可以大大的节省调查成本，并有效地保证数据的真实性，缩短调查周期，因此受到了很大的欢迎。

在“拍照赚钱”模式中，APP中的任务定价是其核心要素。如果定价不合理，就会使任务无人问津，导致商品检查的失败。因此，在企业的成本和用户的利润之间做好权衡，制定合理的任务定价方案，是“拍照赚钱”模式的关键。

1.2 需要解决的问题

基于以上问题背景以及给出的附件数据，我们需要解决以下问题：

问题一：根据附件一中已结束项目任务的数据，研究其定价规律并分析任务未完成的原因。

问题二：为附件一中的项目设计新的任务定价方案，并和原方案进行比较。

问题三：对于位置比较集中的多个任务，可以将其联合在一起打包发布。在这种情况下，对定价模型进行修改，并分析其对最终任务完成情况的影响。

问题四：利用所建立的任务定价模型，对附件三中的新任务制定合理的任务定价方案，并评价该任务的实施效果。

二、问题分析

2.1 问题一的分析

问题一首先需要我们研究任务的定价规律，因此我们需要找到影响任务定价的主要因素，从而进行回归分析，建立出定价方程。根据附录中所给数据，以及常识判断，我们考虑会员密度、任务密度以及目标任务与所在区域中心的距离三个因素，通过主成分分析确立最终的影响因子，再将这些影响因子进行标准化，建立线性回归模型，便可以得出定价规律。

2.2 问题二的分析

在问题一分析出的未完成原因基础上，问题二要求建立新的定价方案，使任务尽量多的被完成，同时考虑到企业的成本，不能只通过提高定价来吸引会员。首先，我们从企业的角度出发，在不低于用户能接受的最低心理价位，即任务可以被顺利完成的情况下，平台中任务的定价分布越接近最低心理价位，企业所需的成本越低。认为用户的心理价位与他们距任务所在地的距离成正相关，即 $E = E_0 + k_r \cdot r_{ij}$，要想得到成本最低的定价分布，只需要确定系数 k。

2.3 问题三的分析

问题二的定价模型中没有考虑到用户选择优先权和打包对于定价过程的影响。在问题三的求解中，我们首先试图论证打包造成效用改进的机理，根据效用改进的可能机制的不同，我们试图提出各种任务打包优化准则，并定性分析符合准则的任务的可能特点。随后，我们选择一种优化准则，将其用形式化语言建模，并提出初步的求解算法。针对数据量较大的问题，我们又通过近似假设，进一步提出了在时限内可解的有效算法。

2.4 问题四的分析

问题四涉及对于一组新任务的定价问题。对于这个问题，我们可以直接利用问题二中已

经建立的模型进行求解。另一方面，通过初步的数据挖掘，我们发现问题四的任务数目相对于会员数目而言较多，而且任务位点分布情况较为集中、特殊，针对这种情况，我们结合了问题三的算法，提出了考虑打包策略时各任务的定价求解方法。

三、模型假设与约定

1、由于任务所在范围为大城市内，经济发达，因此认为会员完成任务的难度仅与会员距离任务所在地的路程有关，忽略交通状况、地势、海拔等次要因素的影响。

2、假设会员只要到达任务所在位置即可顺利完成任务并得到酬金，任务未完成只因为没有会员到达任务所在地。由于任务只需要会员拍照并上传，无技术难度，因此该假设合理。

3、综合考虑任务定价、任务距会员的距离等因素综合决定的任务吸引力，认为每位用户都会优先选择吸引力大的任务来完成。

四、符号说明及名词定义

符号	说明
P_i	任务 i 的实际定价
E_i	会员对任务 i 的最低心里价位
E_0	基础价格常量
r_{ij}	会员 j 到任务 i 的距离
x_1	标准化后的会员密度
x_2	标准化后的任务密度
x_3	标准化后任务距所在区域的距离（简记为任务-中心距离）
x_4	标准化后的任务定价

五、模型建立与求解

5.1 问题一

5.1.1 数据预处理

首先，将附录一中的位置信息导入到地图中，可以得到所有任务和未完成任务的分布图，

分别如图1，图2所示。从地图中可以看出，任务分布主要集中在广州、佛山、东莞、深圳四个较为发达的城市，其中广州与佛山两地地理位置十分接近，且任务分布情况基本相同，因此我们认为可以将整个任务分布范围划分为三个区域。

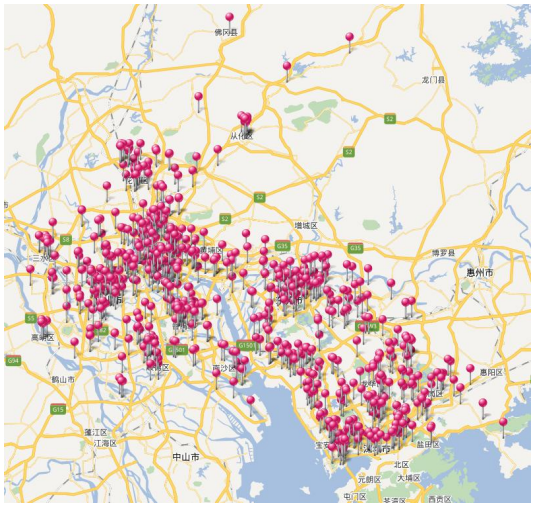


图 1 全部任务的位置分布

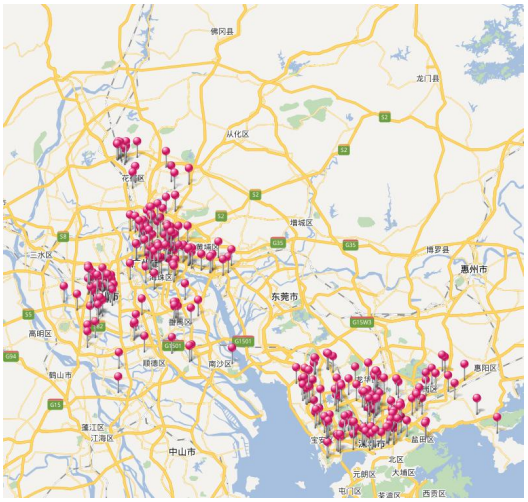


图 2 未完成任务的位置分布

使用K-means算法在MATLAB中进行聚类分析。在此之前，为了保证结果的准确性，需要将距离中心较远的异常点剔除掉。为此我们建立了一个异常处理模型，其基本思想为，在MATLAB中，首先利用dist函数求出每两点之间的距离，建立一个835×835的矩阵，再将矩阵的每一行进行排序；定义常数$\alpha = 0.005$，认为在α分位点之外的值为异常值，得到825组正常数据，我们将以这825组数据作为本论文的参考数据。

利用区域的数据，我们得到了如图3所示结果。其中蓝色区域（区域1）主要为深圳，绿色部分（区域2）主要为东莞，红色区域（区域3）主要覆盖了广州和佛山。这三个区域的中心位置如表1所示。通过聚类分析得到不同区域的中心，可以方便我们衡量不同位置的偏僻程度。

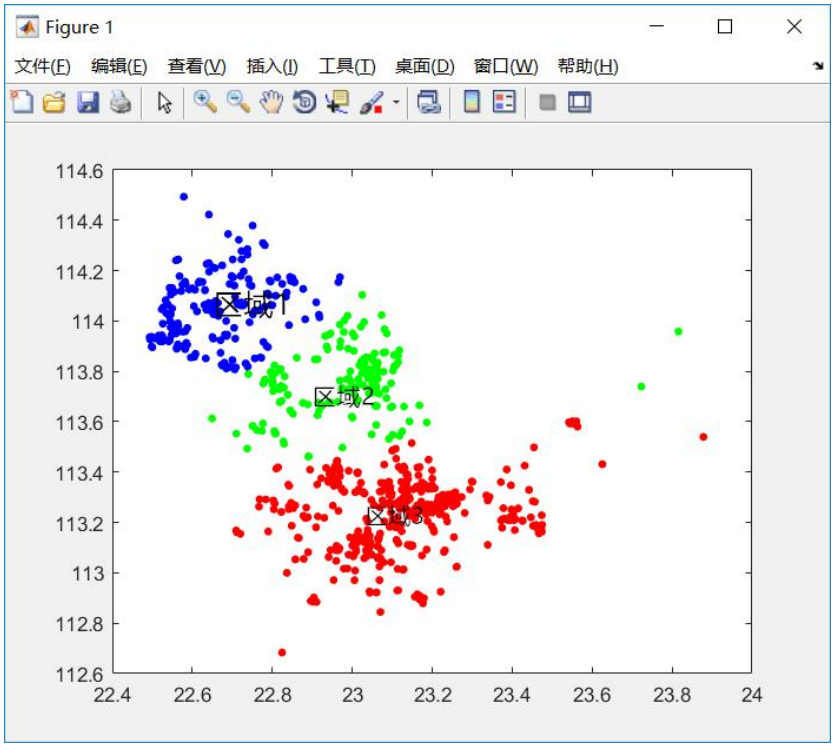


图 3 聚类分析结果

表 1 三个区域中心位置

	区域 1	区域 2	区域 3
经度	22.6676	23.0202	23.1113
纬度	114.0416	113.7245	113.2288

5.1.2 问题一模型的建立

5.1.2.1 影响因子的确立

根据附录中的数据，可以看出任务的定价与任务距中心位置的距离、该任务附近的会员数量和其他任务的数量有关。因此，我们初步认为影响任务定价的因素有三个：会员密度、任务密度及目标任务与所在区域中心的距离。

（1）会员密度

定义某点处的会员密度为以该点为圆心，以 $r=2.5km$ 为半径的圆内会员的数量。利用mathematica统计出每点的会员密度，并分别绘制了三个区域的任务定价与会员密度的关系图，从图中可以看出，任务定价与会员密度大致成反比例关系。

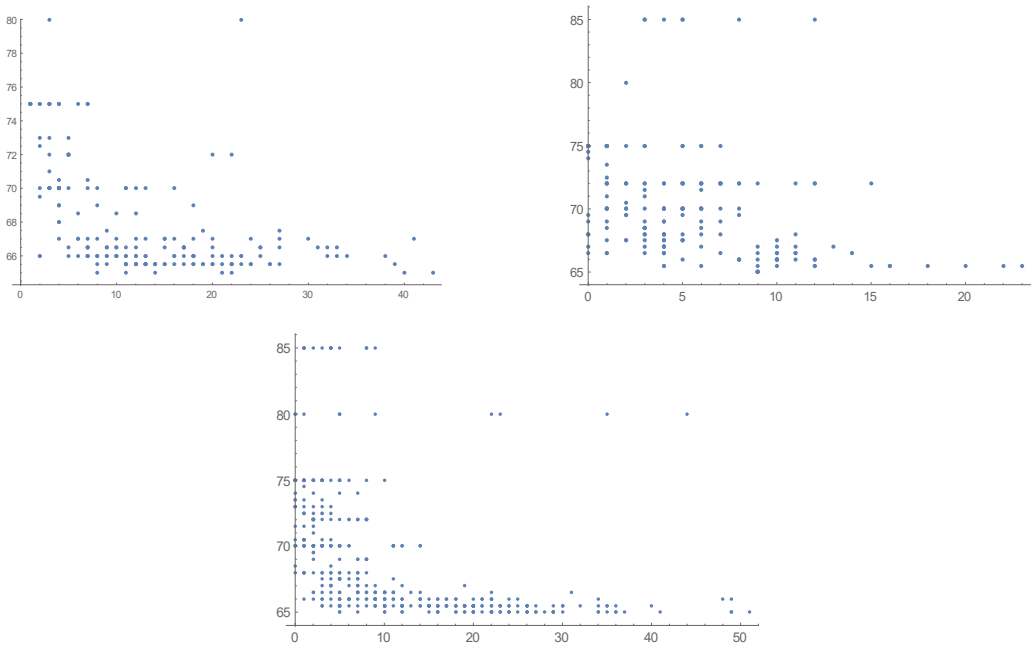


图 4 任务定价与会员密度的关系

（2）任务密度

将整个区域范围分为边长为 c 的张方形网格，定义每个网格内的任务数量为该网格内每一点的任务密度。

我们同样用mathematica检验了任务密度与任务定价的关系，以区域1为例，可见，任务密度与任务定价同样大致成反比例关系。

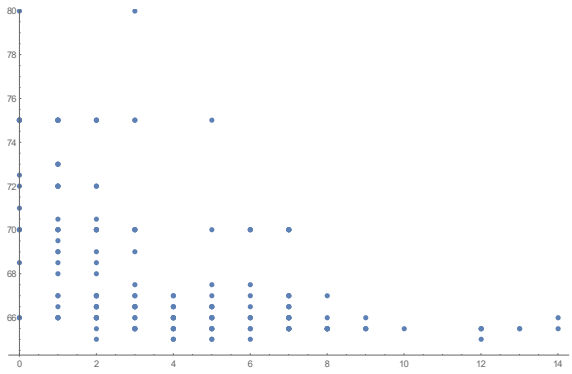


图 5 区域1任务密度与任务定价的关系

（3）目标任务与所在区域中心的距离

为了得到任务与所在区域中心之间的距离，我们首先需要进行经纬度-距离转换。

设任务点的位置可以用有序数对表示为 (a,b)，a 表示经度，b 表示纬度。我们以地心 O 为坐标原点，赤道平面为 xOy 平面，0° 经线所在的平面为 xOz 平面建立三维直角坐标系，可以得到

$$\begin{cases} x = R\cos a\cos b \\ y = R\sin a\cos b \\ z = R\sin b \end{cases} \tag{1}$$

其中 R 为地球半径，$R = 6371.393km$.

根据几何知识，任意两点 $A(a_A,b_A), B(a_B,b_B)$ 间的实际距离为

$$d = R\arccos\left(\frac{\overrightarrow{OA}\cdot\overrightarrow{OB}}{\left|\overrightarrow{OA}\right|\cdot\left|\overrightarrow{OB}\right|}\right) \tag{2}$$

将式（1）带入（2）化简可得：

$$d = R\arccos[\cos(a_A - a_B)\cos b_A\cos a_B + \sin b_A\sin a_B] \tag{3}$$

由图6可知，任务定价与任务-中心距离大致为正比例。

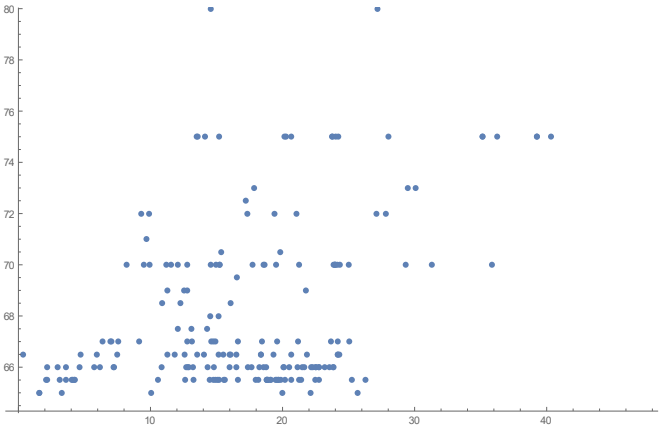


图 6 区域1任务-中心距离与任务定价关系图

5.1.2.2 任务定价规律

首先对以上得到的三个因素进行标准化，得到三个因子分别为x_1, x_2, x_3，在此基础上，我们建立多元线性回归模型来求解定价规律。设基础价格为w_0，则任务定价可以写为

$$P = C_1 x_1 + C_2 x_2 + C_3 x_3 + w_0 \qquad (4)$$

现在我们已知835组x_1, x_2, x_3数据，利用最小二乘法可以拟合出系数C_1, C_2, C_3，得到定价方程为

$$w = 69.6936 + 0.106571 x_1 - 0.128554 x_2 - 0.372981 x_3$$

5.1.2.3 任务未完成原因

认为任务是否被完成与会员密度、任务密度、任务-中心距离和任务定价有关。为了更直观的说明任务未完成原因，我们利用神经网络得到任务完成情况与上述影响因子的关系。

1、神经网络的搭建

综合考虑分类能力、学习速度和逼近能力等方面，我们选择径向基神经网络进行学习和逼近，其基本结构如图7，表达式为

$$y = purelin(b_2 + LW^{2,1} radbas(LW^{1,1} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} - b_1))$$

其中$LW^{1,1}$为输入层到隐藏层的权值矩阵，$LW^{2,1}$为隐藏层到输出层的权值矩阵，b_1为权值向量与输入向量之间的矢量距离偏差，用于调整网络的灵敏度，b_2用于调整最终输出结果，当$y < 0.5$时认为任务未完成，$y \geq 0.5$为任务完成。

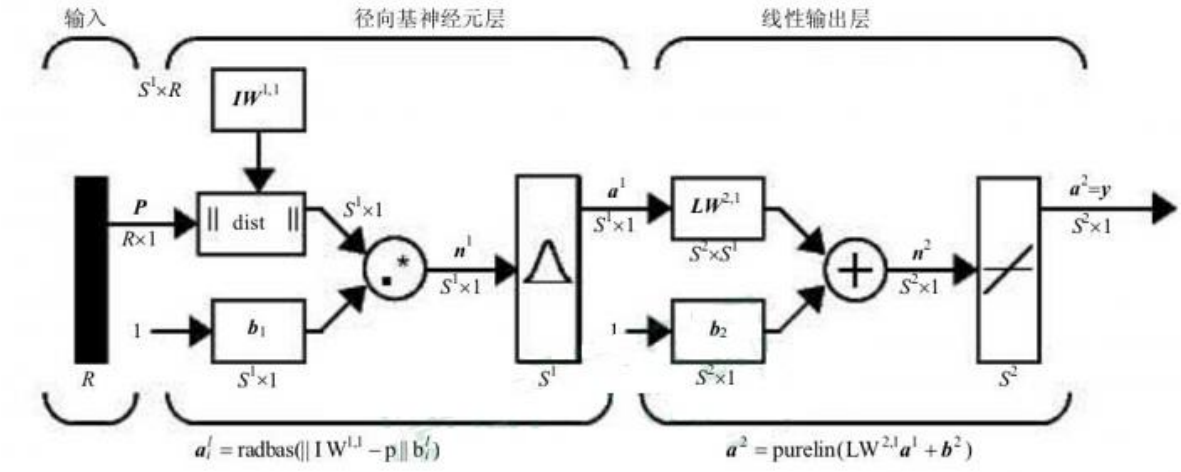


图 7 径向基神经网络结构图

选取附录一中的前500组数据对该网络进行训练，得到了$LW^{1,1}$、$LW^{2,1}$、b_1、b_2的数值，我们用其余数据作为测试数据，结果表明每次测试的正确率在80%左右，神经网络性能良好。

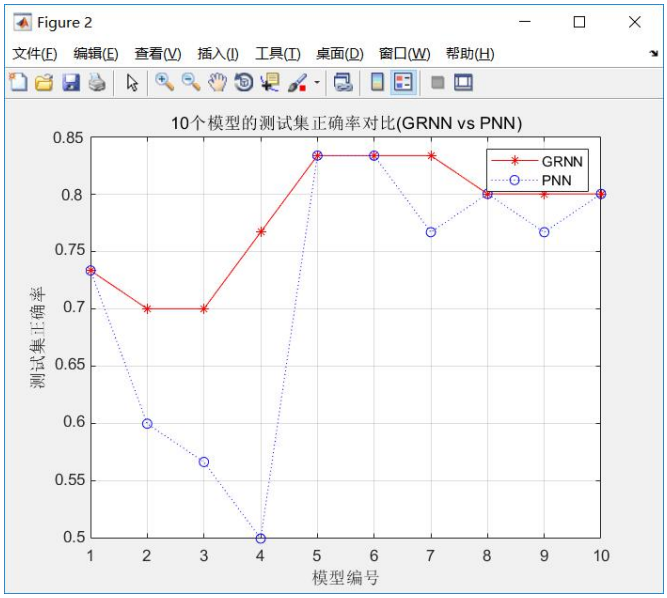


图 8 神经网络测试结果

2、未完成原因分析

由于方程含有四个未知数，难以直接绘制图像，且函数求导结果过于复杂，不便于分析，因此我们分析某一变量时统一取其余变量的平均值，将研究变量由小到大排序，观察输出结果，从而得到未完成原因。

（1）会员密度因素

令 $x_2=3.9188$，$x_3=0.1837$，$x_4=69.0461$，得到的结果如图9所示。从图中可以看出，在任务密度、任务的偏僻程度、定价均比较适宜的情况下，未完成的任务主要集中在会员密度为（70,80）区间内。这表明，当会员密度很低时，由于每个会员周围的任务数量较多，他们有机会同时完成几个任务，对于会员来说，这样的可以大大的节省他们赚取多份酬金的时间成本，而随着任务数量的增加，会员在一定距离内有机会分到的任务数减少，因此许多会员会选择放弃任务，导致任务未完成的情况出现。随着会员的密度继续增加，每项任务都有许多会员可以完成，因此任务被完成的概率又大大提高。

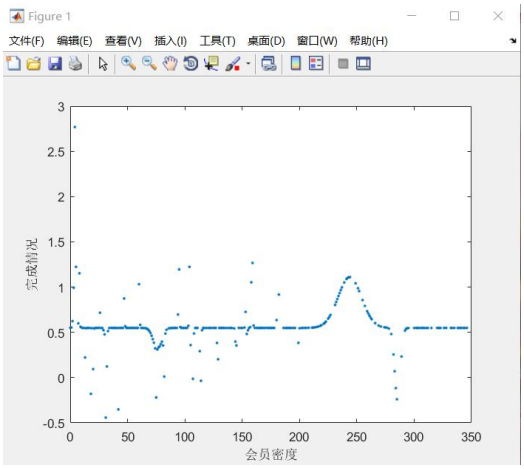


图 9 会员密度与完成情况的关系

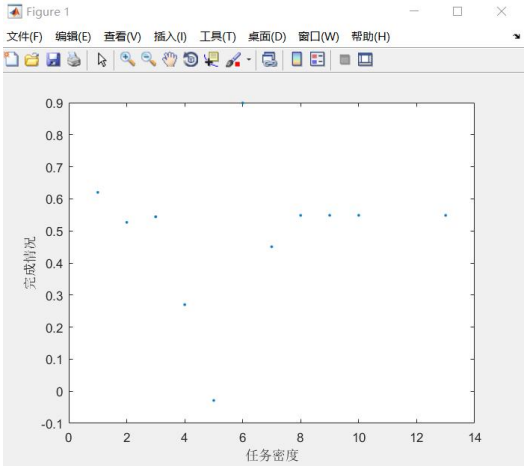


图 10 任务密度与完成情况的关系

（2）任务密度因素

令 $x_1=106.2145$，$x_3=0.1837$，$x_4=69.0461$，即在其他三个因素条件适中的情况下，

得到了任务密度在区间(1,14)内任务的完成情况。与会员密度同理，当任务密度很小时，相当于每个任务周围的会员密度大，因此任务完成率高，当任务密度增加到4、5左右，任务无法被完成；随着密度的继续增加，会员获得多份酬金的时间成本降低，因此任务又可以被完成。

（3）任务-中心距离因素

令$x_1=106.2145$，$x_2=3.9188$，$x_4=69.0461$，从图11中可以清晰地看出，在一定范围内完成情况与距离成反比。当任务位置偏僻时，会员完成任务所需要的时间成本和费用均较高，而我们在此所取的定价为平均值，因此任务完成率迅速下降，而当距离很大时，由于此时任务较少，相当于任务密度小，会员密度大，因此任务完成情况出现好转。

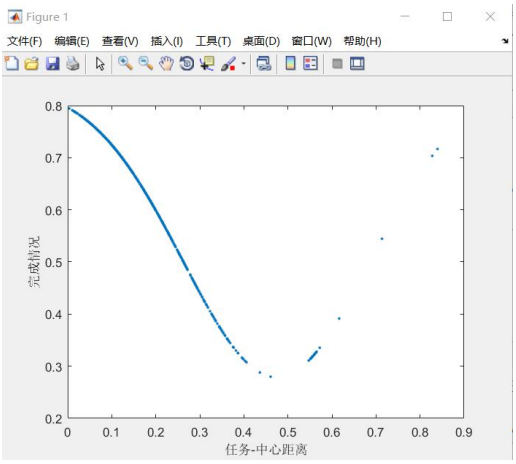


图 11 任务-中心距离与完成情况的关系

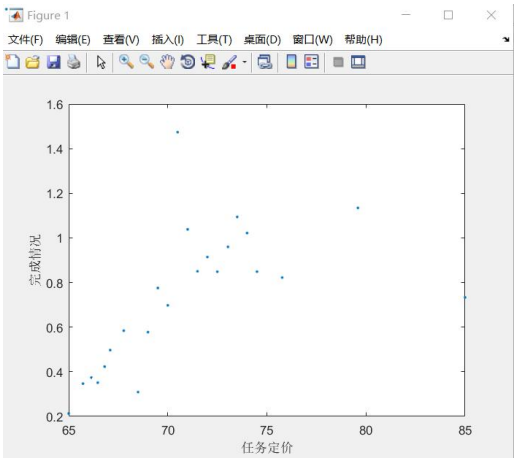


图 12 任务定价与完成情况关系

（4）定价因素

当其余因素一定时，不难理解，一项任务多给出的定价越高，完成度就会越高。当任务的定价低于会员根据自己付出的成本所得到的心理预期定价时，任务便无法被完成。

综上可知，任务未完成与会员密度、任务密度、任务-中心距离以及任务定价四个因素均有关系，如果一项任务的位置较偏僻，定价较低，会员又无法在较近的距离内完成其他任务，则任务很可能无法完成。

5.2 问题二

我们的目标是建立一种新的定价方案，在保证任务完成率最高的情况下，任务定价的总额尽量小，从而兼顾企业的成本和用户的利益。

本模型从企业的角度出发，在定价不低于用户能接受的最低心理价位，即任务可以被顺利完成的情况下，平台中任务的定价分布越接近最低心理价位，企业所需的成本越低，此时得到的定价分布即为我们优化后的定价方案。

5.2.1 模型假设

1、等吸引力“均衡”价格假设：

为避免顾客流失，两个距离相近的商店所出售的高度相似、可完全替代的商品的价格一般也趋于相同，由这一简单的生活经验出发，我们可以引申抽象出商品的等吸引力定价原则。

考虑会员数$m=1$，平台任务数n任意的最简单的情况。

假设平台方对于各任务的完成与否的偏好无差别，考虑平台方作为经济学意义上的“理性人”的决策过程：假设对于每个任务i，其定价对于会员的吸引力为$A_i(P_i)$，那么，如果存在$i(1\le i\le n)$，使得对于$\forall j,1\le j\le n$，$i\ne j$，均有$A_j<A_i$，则根据会员方利益最大化

的原理，如果在定价水平 P_i 下，会员的收益为正，则会员将选择执行任务 i，而完全不会考虑选择其它任务，这样，平台方对于其它任务的定价为无效定价，其带给平台方的效用预期为0，这是一种无效率的价格配置。

显然，当且仅当各任务带给会员的吸引力无差别，即 $A_1(P_1)=A_2(P_2)=\cdots=A_n(P_n)$ 时，所有任务都有机会被会员选择，此时各任务达成了其均衡价格。

类比物理学上对于受力平衡态的表述，我们可以把等吸引力均衡价格假设表述为：各任务达成均衡价格，当且仅当它们对会员造成的吸引力相等。

2、正收益成交假设：

无论是会员方还是平台方，其达成交易的条件是：效用-成本>0。

设平台方因为任务完成而获得的效用为 Q，平台方外包该任务所支付的价格为 P，会员对于完成任务所获报酬的预期底线为 E，则根据正收益成交假设，有成交价格范围：$Q>P>E$。

由于只要收益为正即可达成交易，对于拥有定价权的平台方而言，实现利益最大化的做法是令 $P\to E_+$，因而我们可以认为平台方对任务的理性定价为 E。

在会员数，平台任务数 n 任意的情况下，这意味着平台对所有任务的定价应当为满足吸引力相等条件的最低可能成交价格。

3、距离补偿机制假设：

任务发生位点与会员所在地点之间的距离会造成会员的额外花费，这造成远距离任务对于会员的吸引力的下降。为补贴会员的额外花费，使得各任务对于会员的吸引力相等，因而需要引入距离补偿机制。

假设 $P=E+k_r\cdot r$，其中 E 为任务本身的最低可能成交价格，r 为会员与任务的距离，k_r 为单位距离开销的折现系数。引入距离补偿机制后，可以使得距离会员不等的任务对于会员造成的吸引力相等。

4、预期报酬浮动假设：

会员会在当地的各种经济活动中选择对于自己效益最优者，因而，当地的经济发展水平与物价水平会影响到会员的预期报酬。在本模型中，我们假设 $E=E_0+E_1$，其中 E_0 为任务的基础成本，E_1 为当地经济、物价水平造成的报酬预期溢价。

5、等效距离假设：

任务的特点、会员个人的禀赋等一系列因素会导致任务对于会员的吸引力出现变化，我们假设这些影响因子对于吸引力的影响是线性的。在本模型中，我们只限于考虑任务i对于会员j的难度的影响，我们假设任务对于会员的难度取决于会员本身的特质，并且可以体现为信誉值 k_{cj} 的函数 $D_j(k_{cj})$。在数学上，我们把难度的影响折算为距离的影响，这样，地理距离与难度因素共同形成了等效距离 $\tilde{r}=r+r'(D_j)=r+r'(k_{cj})$。

5.2.2 模型的建立

会员的定价占优区域：

我们先考虑会员数 $m=2$ 的情况。当两个会员A、B在 n 个任务中做出选择的时候，平面区域被 $\tilde{r}_{iA}=\tilde{r}_{iB}$ 划分为2个区域，其中 $\tilde{r}_{iA}=\tilde{r}_{iB}$ 为一条直线或双曲线。当任务落在的区域中时，无论任务定价如何，其对会员A的吸引力恒大于对会员B的吸引力，此时平台方只需根据会员A的报酬预期进行定价即可，因而我们称 $\tilde{r}_{iA}<\tilde{r}_{iB}$ 的区域为会员A的定价占优区域，在此区域内为会员A的买方市场，平台方必须支付A的最低报酬预期方可达成交易。

基于以上假设，我们建立定价模型：

$$P_i=E_0+E_1+k_r\cdot\min_{j\in M}\tilde{r}_{ij}$$

事实上，对于任意任务 i，总存在会员 j_0，使得两者的等效距离 $\tilde{r}_{ij_0}$ 达到最小值。由前

面的分析我们知道，此时任务i对会员j_0的吸引力大于对其它任何会员的吸引力，即i落入了会员j_0的定价占优区域，因而任务i的价格将由$\tilde{r}_{ij_0}$决定。

此外，从平台方的正收益成交假设看，应当有：

$$\sigma_{ij}=1\leftrightarrow P_i=E_0+E_1+k_r\cdot\tilde{r}_{ij}=\min_{j\in M}(E_0+E_1+k_r\cdot\tilde{r}_{ij})\ \text{且}\ P_i\le Q$$

其中Q是平台方的效用。设会员j的失信风险为，根据风险中性定价原理，应当有：

$$Q=Q_0(1-R_j)$$

Q_0为任务成功完成时，平台方获得的效用。

参数估计：

在本模型中，$r'(k_{cj})$按照如下方式确定：

$$r'(k_{cj})=\begin{cases}3,\ k_{cj}<2\\ 2,2\le k_{cj}\le 50\\ 1,50\le k_{cj}<5000\\ 0,\ k_{cj}\ge 5000\end{cases}$$

R_j按照如下方式确定：

$$R_j=\begin{cases}0,3<k_{cj}<2\\ 0.2,2\le k_{cj}<20\\ 0.1,50\le k_{cj}<50\\ 0.05,50\le k_{cj}<400\\ 0.02,400\le k_{cj}<5000\\ 0.01,k_{cj}\ge 5000\end{cases}$$

根据题目给出的任务成交价格的分布范围，我们给出完成任务的基础成本$E_0=60$，聚类A、C分别对应经济发达的广州市和深圳市，我们定义。$E_{1A}=E_{1A}=5$，$E_{1B}=0$。

根据生活经验，我们取每公里距离的价格补贴系数为1.25元/千米，当以经纬度为距离度量单位时，得到$k=130$。

5.2.3 模型的求解

我们采用动态模拟仿真法求解该模型，具体过程如下：

（1）建立数组Available，表示每个会员是否已经选择了任务，0表示已选择，不再考虑匹配新的任务，1表示未选择，可以选择新的任务。初始化时，Available数组各元素均为1。

（2）根据每个会员j的信誉值计算等效距离校正值$r'(D_j)$。

（3）遍历各任务i，计算地理距离r_{ij}，进而计算出等效距离$\tilde{r}_{ij}$。对于每个i，将等效距离向量各分量$\tilde{r}_{ij}$进行排序，按照由小到大的顺序依次检查会员j的Available值，找到第一个Available值为1的会员j。

（4）由$P_i=E_0+E_1+k_r\cdot\tilde{r}_{ij}$确定任务$i$的会员所能接受的最低成交价格，通过检验是否满足平台方成交条件。若满足，将会员j的Available值更新为0，更新总交易数与总交易额的数据。若不满足成交条件，跳至（3）。

（5）重复执行（3）、（4）的步骤，直至完成对i的遍历。输出总交易数和总交易额的数据。

模型求解结果：

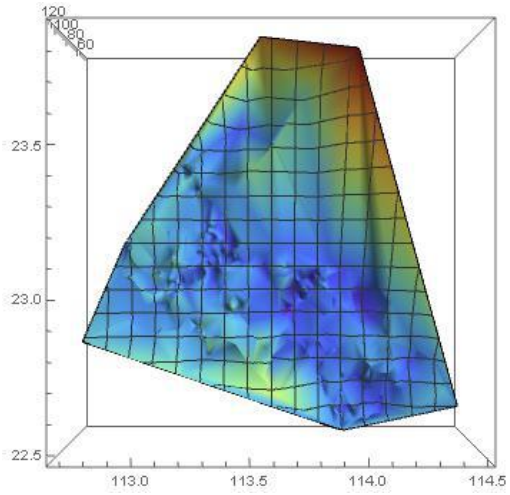


图 13 任务分布求解结果

经计算得，总成交额为60107元，总成交量为774，平均成交价格为77.66元，成交率为92.7%。

5.2.4 模型的改进

在上述讨论中，我们利用生活经验，得到 k 值的估算，进而得到具体定价方案，但由于生活经验的限制，使评判过程具有一定的模糊性。在此，我们对上述定价方案表达式进一步修改优化，引入心理预期价格的概念，将 k 值作为未知数处理，通过计算机的仿真模拟，得到更为精确和高效的定价方案，同时逆向得到了经济学经验值 k 的估计。

对于一个给定的用户 j，在其决定是否选择某项任务 i 时，会将任务的实际价格与最低心理价格 E_{ij} 比较，该心理最低价格与用户完成任务所付出的成本成正相关，且认为模型1中的正收益成交假设仍然成立，即当实际价格低于该心理价格时，用户不会完成该项任务。忽略天气、交通状况等因素的影响，认为用户付出的成本只取决于其距离任务的实际距离 r_{ij}，因此有：

$$E_{ij} = E_0 + r_{ij} * k_r$$

其中 E_0 为任务的基础价格，k_r 为单位距离开销的折现系数。

设任务 i 的实际定价为 P_i，用户会将自己的心理预期价格与实际价格比较并选取性价比最高的任务，故选取 P_i/E_{ij} 最高的任务作为用户的目标任务，为了保证算法的合理性和可行性，我们做了如下约定：

• 当 P_i/E_{ij} <1时，任务 i 不会被用户完成。

• 若用户 j 不会去任何一个任务点，则在结果中加入惩罚函数（如+100），使其远离最优解。

• 将用户按照预定任务开始时间进行排序，时间相同时再按照信誉度排序，按此顺序对用户进行任务遍历。

• 用户预定的任务数不可超过其预定任务限额。

设任务的数量为 n，若任务 i 被用户 j 按照上述标准选中，用户 j 距任务 i 距离为 r_{ij}，则总的等效路径和为

$$r = \sum_{i=1}^{n} (r_{ij} + 100 w_i)$$

其中 $w_i=\begin{cases}0, r_{ij}\neq 0\\ 1, r_{ij}=0\end{cases}$，为惩罚函数。

在 k_r 一定的情况下，随机生成不同的定价分布矩阵，找出总路径和最短的矩阵，即为此 k_r 值下的最优定价分布。设此时的定价分布矩阵为 A_k 。

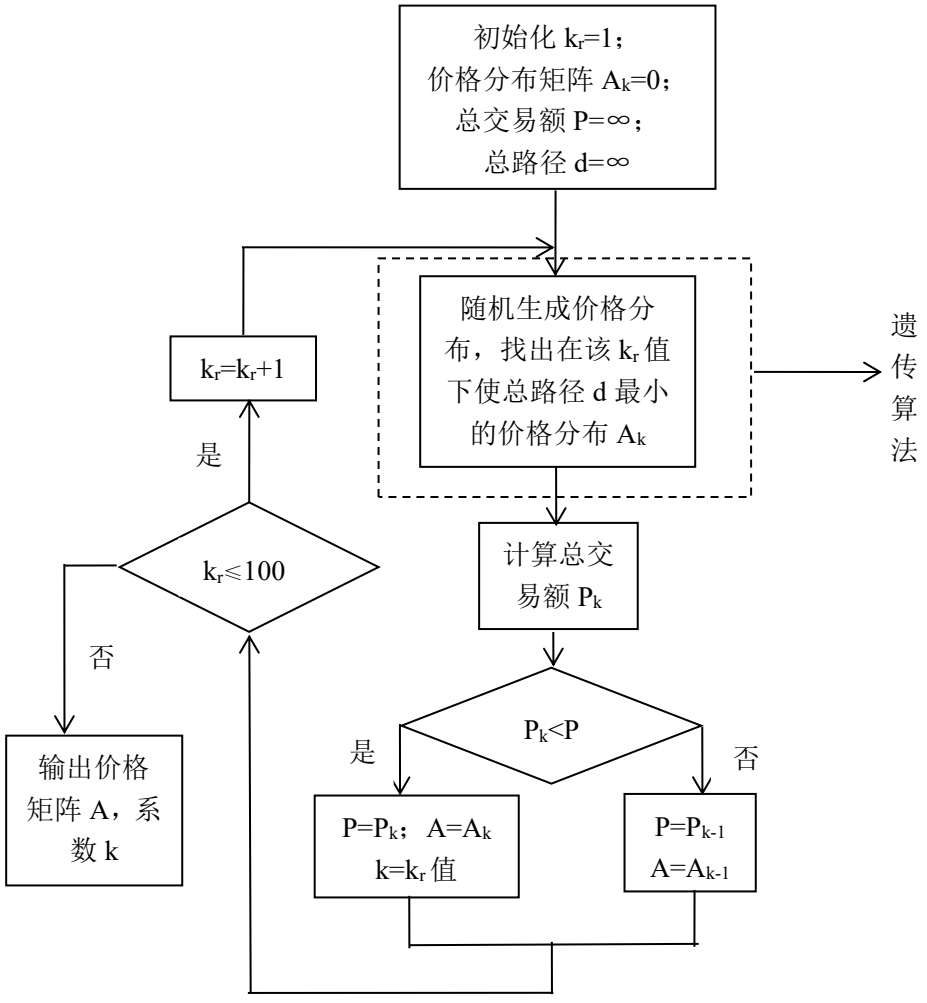
若在某种分布下用户 j 未选择任何一项任务，此时引入的惩罚函数值很大，因此该分布情况几乎不可能被选中，该模型保证了每项任务均可以被完成。

设在 A_k 的任务分布下，任务 i 被用户 j 按照上述标准选中时的任务定价为 P_i，则总交易额 $P_{总k}$ 为

$$P_{总k}=\sum_{i=1}^{n}P_i$$

测试不同的 k_r 值，每个 k_r 值均对应一个交易额 $P_{总k}$，选出使交易额达到最小的 k_r 值，则为在任务均被完成的情况下使企业成本达到最低的 k_r 值，其对应的定价分布矩阵 A_k 为最优的任务定价方案。

我们用MATLAB确定 k 的值和任务的定价分布，求解上述模型。考虑到企业的成本问题，用户的心理预期价格不应过大，因此我们设定 k 的最大值为100，在（1,100）区间内通过对 k 进行穷举确定 k 的最优值，代码流程如下：



在以上算法中，由于价格分布不可能穷举，因此最关键的一步是如何根据上一次得到的总路径调整价格分布矩阵，使下一次得到的总路径长度更接近最短路径。我们采用遗传算法，采用概率化的寻优方法，从而自动获取和指导优化的搜索空间，自适应地调整搜索方向，而不需要确定的规则。

为了测试程序的运行结果，我们令程序输出每个 k 值对应的交易额，结果如下：

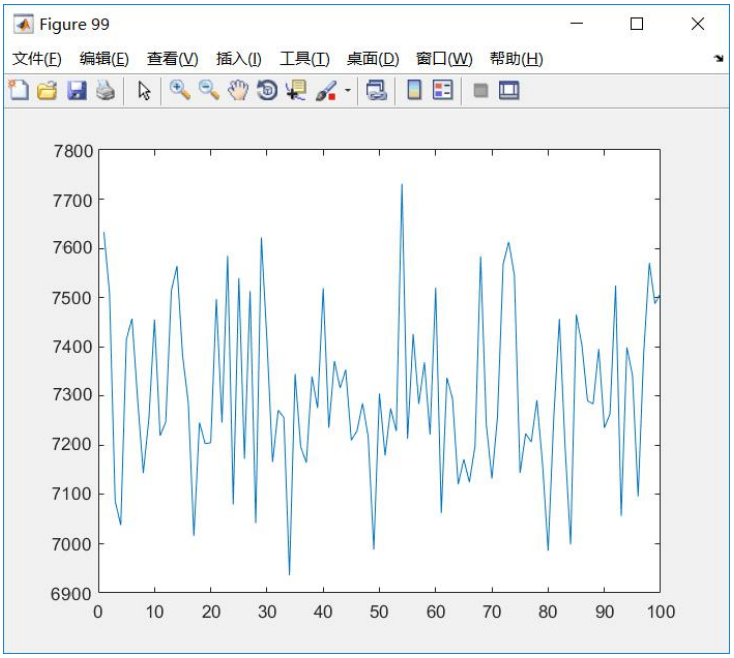


图 14 交易额与 k 值关系

该图像横轴为 k_r 值，纵轴为在总路径最短的定价分布下的交易额。可见，k_r =34时交易额最小，表明 k_r =34时得到的定价分布矩阵为保证任务全部完成的情况下企业成本最低的定价分布，也就是我们所提出的最优定价方案。

5.3 问题三

5.3.1算法一

5.3.1.1 模型的建立

任务打包对于平台与会员双方的效用改进：

考虑任务 $t_1,t_1,\cdots t_n$ 与会员 j 。为了使每个任务 t_i 对于会员 j 具有等吸引力，我们需要任务 t_i 的定价 $P_i = E_0 + E_1 + k_r \cdot (2 \cdot r_{ij} + r^{'}(k_{cj}))$ 。这样，为了n个任务，平台方共需支付费用：

$$P = \sum_{i=1}^{n} P_i = n(E_0 + E_1) + nk_r r^{'}(k_{cj}) + 2nk_r r_{ij}$$

如果将任务 $t_1,t_1,\cdots t_n$ 进行打包处理，则会员实际上可以一次性遍历所有任务位点，这造成实际距离开销的下降。在这种情况下，由正收益成交假设，会员所能够接受的打包任务的最低价格为：

$$P^{'} = n(E_0 + E_1) + nk_r r^{'}(k_{cj}) + k_r \min_{\sigma \in S_n} \left\{ r_{j\sigma(1)} + \sum_{k=1}^{n-1} r_{\sigma(k)\sigma(k+1)} + r_{\sigma(n)j} \right\}$$

这样，平台方和会员方的总的效用改进为 $P - P^{'}$ 。只要平台方的定价满足 $P^{'} \le P_0 \le P$ ，则满足双方的成交条件，这样，效用改进量将在平台方与会员方之间进行重新分配，其中平

台方获得效用增量 $P-P_0$，会员方获得效用增量 $P_0-P^{'}$。

5.3.1.2 模型的求解

1、优化准则的提出：

（1）任务距离就近原则：

假如 n 个距离较近的任务可近似认为排布在一个半径为 r 的圆周上，会员距离该圆周圆心的距离为 R。我们考虑两种极端情况下打包策略对于距离开销下降的近似估计。则当 $R>>r$ 时，可近似计算，认为

$$r_{j\sigma(1)}+\sum_{k=1}^{n-1}r_{\sigma(k)\sigma(k+1)}+r_{\sigma(n)j}\approx 2R+2\pi r$$

因而将 n 个任务打包带来的效用改进

$$P-P^{'}=k_r[2(n-1)R-2\pi r]$$

当 $R\approx 0$ 时，类似地，可以近似认为打包前地理距离开销为 $2nr$，打包后地理距离开销为 $2r+2\pi r$，因而效用改进 $P-P^{'}\approx 2k_r r(n-1-\pi)$。实际上，通过仿真验证也表明，当用户与各任务距离都比较近时，打包规模 $n\geq 5$ 时，效用改进一般为正值；当 $n=4$ 时，打包策略有时能够带来效用改进，有时则不能。

综上所述，将各任务按照距离就近的原则打包，往往能够节省价格构成中的距离开销，从而带来效用的改善。

（2）任务成交可能性搭配原则：

假设对于会员 j，存在任务 i_1，i_2。分开定价时，有

$$P_{i2}=E+2k_r r_{i_2 j}>Q_0(1-R_j)-E-2k_r r_{i_1 j}$$

则任务 i_2 为保证对于会员具有等吸引力，其定价补偿过高，超出了平台方的承受能力，此时，只有任务 i_1 达到成交条件，平台方的效用为

$$Q=Q_0(1-R_j)-E-2k_r r_{i_1 j}$$

将任务 i_1，i_2 打包定价时，会员可接受的最低定价下降为

$$P^{'}=2E+k_r(r_{i_1 j}+r_{i_1 i_2}+r_{i_2 j})$$

如果打包策略制定得当，则 $P^{'}<2Q_0(1-R_j)$，打包任务满足双方成交条件，平台方获得效用 $Q^{*}=2Q^{0}(1-R_j)-2E-k_r(r_{i_1 j}+r_{i_1 i_2}+r_{i_2 j})$

通过简单的观察可知 $Q<Q^{*}<2Q$。更为一般地，我们可以归纳出这样一条规律，即引入得当的打包策略后，平台方获得的效用总量增加了，其代价是，对于全部成交任务而言，每个任务带来的平均效用量有所减少。

通过进一步观察，我们发现 $P_{i_1}+P_{i_2}-P^{'}=k_r\left(r_{i_1 j}+r_{i_2 j}-r_{i_1 i_2}\right)$，即只有打包节省的距离开销 $r_{i_1 j}+r_{i_2 j}-r_{i_1 i_2}$ 较为可观的时候，打包策略（2）才有可能有效。通过定性分析，我们发现打包策略（2）可能发生在以下两种情况：

<1>任务 i_1,i_2 距离会员都较远，但任务 i_1,i_2 之间的距离很近。

<2>会员、任务 i_1,i_2 三者两两之间都有一定的距离，该距离不太远也不太近。

考虑三角不等式的等号成立条件，容易知道，另有两种情况明显不适用于策略（2）：

<1>一个任务距离会员很近，另一个任务距离会员很远。

<2>两个任务对于会员而言处于相反的对角线方向。

（3）任务距离远近搭配原则：

由打包策略（2）的讨论我们知道，如果仅仅从节省距离成本的角度看，一远一近两个任务的搭配打包策略未必是经济的，然而考虑到平台方的综合诉求，如开拓遥远地区的市场、扩大影响力，平衡兼顾各地区的市场发展和用户满意度，等等，则我们仍然可以提出任务按

照距离远近搭配打包的原则。一般来说，按照此策略打包考虑的不完全是单笔交易的经济效益，因而这需要我们重新定义平台方的综合效用，然后进行模型分析。

（4）成交量最大化、迁移量最小化原则：

原题中给出的关于会员、任务与交易情况的数据都是离散的，实际上，为了便于数据的分析与挖掘，我们可以通过适当的方式，将离散数据连续化，构建连续型的模型。

对于任意一点(x_0,y_0)，我们定义如下集合：

$$T(x_0,y_0)=\{t_i(x_i,y_i)||x_i-x_0|<\varepsilon,|y_i-y_0|<\varepsilon\}$$

$$T_0(x_0,y_0)=\left\{t_i(x_i,y_i)\middle||x_i-x_0|<\varepsilon,|y_i-y_0|<\varepsilon且\exists j\in M,s.t.\sigma_{ij}=1\right\}$$

$$M(x_0,y_0)=\{m_i(x_i,y_i)||x_i-x_0|<\varepsilon,|y_i-y_0|<\varepsilon\}$$

其中$\mathrm{T}(x_0,y_0)$表示经度为x_0，纬度为y_0的位点的 ε −方邻域内的任务t_i的集合，$T_0(x_0,y_0)$表示经度为x_0，纬度为y_0的位点的 ε −方邻域内的成交任务t_i的集合，$\mathrm{M}(x_0,y_0)$表示经度为x_0，纬度为y_0的位点的 ε −方邻域内的成员m_i的集合。

我们定义如下函数，从宏观、统计的角度表征会员与任务的数据信息：

$$t_d(x_0,y_0)=|T(x_0,y_0)|$$

$$m_d(x_0,y_0)=|M(x_0,y_0)|$$

$$\mathrm{c}(x_0,y_0)=\sum_{m_i\in M(x_0,y_0)}c_i(m_i)$$

$$P(x_0,y_0)=\begin{cases}\dfrac{1}{|T(x_0,y_0)|}\displaystyle\sum_{t_i\in T(x_0,y_0)}P_{t_i},|T(x_0,y_0)|>0\text{ 时}\\ 0,|T(x_0,y_0)|=0\text{ 时}\end{cases}$$

$$r_d(x_0,y_0)=\begin{cases}\dfrac{|T_0(x_0,y_0)|}{|T(x_0,y_0)|},|T(x_0,y_0)|>0\text{ 时}\\ 0,|T(x_0,y_0)|=0\text{ 时}\end{cases}$$

其中t_d表示任务密度，m_d表示会员密度，c表示周边地区的运力密度，P表示周边地区平均定价，r_d表示周边地区平均成交率。

以下是根据题目中的数据得到的各地区周边任务密度（左）、平均成交率（中）、平均定价（右）的函数图像：

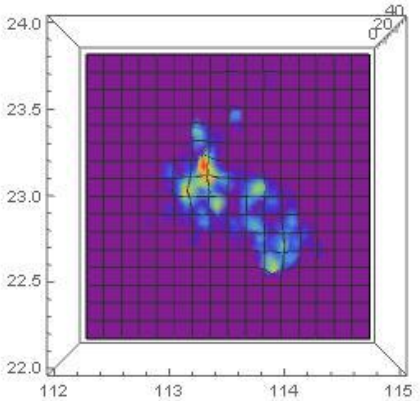


图 14　周边地区任务密度

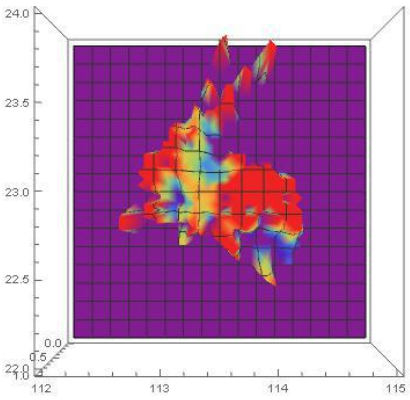


图 15　平均成交率

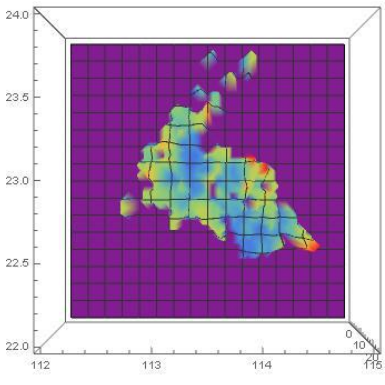


图 16　平均定价

从生活经验出发，我们可以做出假设，认为成交任务的密度与所在地区的任务平均定价和运力密度成正比，即：

$$t_d(x_0,y_0)r_d(x_0,y_0)=kP(x_0,y_0)c(x_0,y_0)$$

实际上，我们可以赋予系数 k 这样一个实际意义，即大量运力涌入某地区时，k 值增大，

使得在同等定价与运力密度的条件下，成交任务的密度大大增加。

我们绘制 k 的函数图像，得到：

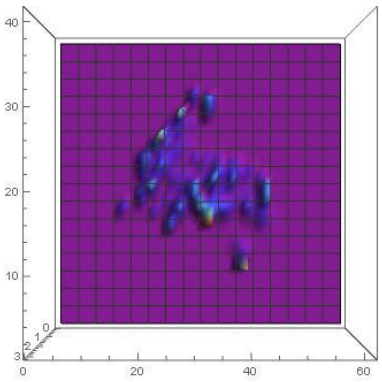


图17 系数k

从图像中可以看出，大部分地区的运力迁入率都限制在一个比较小的数值之内，只有个别地区出现了会员运力大量迁入的情况。根据距离补偿机制假设，用户为完成任务而发生的位置迁移量越大，平台方支付的费用越高，因而在我们的模型中，可以把最大化成交任务量、最小化会员迁移量作为打包策略的制定依据。

具体而言，我们可以把会员看作运力的承载者，把任务打包并与会员匹配的过程看作是对于运力的调度过程，会员与平台方的成交价格条件可以视为调度的可行性约束条件，我们的打包决策就是找到一个运力调度方案，使得在可调度条件的约束下，实现总调度量最大，总迁移量最小。

（5）熵增原则：

我们知道，平台的支付价格取决于运力与任务分布的匹配程度。从直觉上看，会员与任务的分布混杂越均匀，系统的熵值越大，平台的支付成本越小。打包可以看作调整任务分布的一种方式，我们可以尝试度量打包策略对于系统熵值的影响，以熵增原则求解最优打包策略。

2、具体求解过程

下面，我们聚焦于原则（4），用形式化语言建立模型。

定义1：称会员$m_i(\tau_i,k_{ci})(i=1,2..n)$的排列 $Q(m_1,m_2,..,m_n)=m_{q_1}m_{q_2}...m_{q_n}$为会员决策序列，如果对于$\forall i<j,\tau(m_i)\le\tau(m_j)$且$\forall\tau(m_i)=\tau(m_j)$，若$i<j$则$k_{ci}\ge k_{cj}$。

在上述定义中，$\tau_i,\tau(m_i)$表示会员m_i的预定任务开始时间，k_{ci}表示会员的信誉值。

定义2：打包策略s为任务集T的分划，即$s=\{T_1,T_2,..,T_k\}$，使得$\cup_{i=1}^{k}T_i=T$且$T_i\cap T_j=\emptyset,\forall 1\le i,j\le k,i\ne j$。记全部可能的打包策略构成打包策略空间S。

定义3：定义$\varphi=\varphi_1\cdot\varphi_2\cdot...\cdot\varphi_n$为任务分配方案，其中映射$\varphi_i$:

$$\varphi_i:M\to\tilde{s},\tilde{s}\text{为s的子集}$$

$$m_i\to\{T_{i_1},T_{i_2},...,T_{i_l}\}$$

为对于会员m_i的任务包分配映射。根据会员按照时序与信誉值的优先选择约束，应当有$i<j$时，与φ_j相比，φ_i作用在会员决策序列的排序靠前的会员元素上。整体的任务分配方案φ即为对于每个会员的任务分配映射φ_i的复合映射。

在以上定义的基础上，模型可表述为：

确定$s\in S$以及$\varphi(M,T,S)$，使得：

$$\max\sum_{T_i\in P(M,T,S,\varphi)}|T_i|$$

$$\min\sum_{i\in M}r_{ij}$$

其中：

$$P(M,T,S,\varphi)=\left\{T_i\middle|T_i\in S,\exists j,s.t.\sigma_{ij}=1,|T_i|=n_i,s.t.n_i(E_0+E_1)+k_r(r_{ij}+r')\le n_iQ_0(1-R_j)\right\}$$

为全部成交任务包的集合。

$$r_{ij}=\min_{a_0,a_1\cdots a_n\text{为遍历任务点的路径}}\left\{\left|M_j a_0\right|+\left|M_j a_n\right|+\sum_{k=1}^{n}\left|a_{k-1}a_k\right|\right\}$$

运力约束条件为：$\sum_{T_i\in\varphi_i(m_i)}|T_i|\le C_i$

C_i为会员运力约束，在本模型中，我们认为会员每天实际的运力是具有上限C_{max}的，因而有：$C_i=min\{C_{max},L_i\}$，其中L_i为会员i的配额限制，在本模型中我们取每日运力上限为常数$C_{max}=5$。

最后，引入会员优选准则：

分配给会员m_i的任务包$\varphi_{M,T,S}(m_i)=s_0$，s_0为s的子集，$s_0=\{T_1,T_2,..,T_{m_i}\}$使得$\frac{|s_0|}{\sum_i r_{ij}}$达到最大。其中$|s_0|=\sum_{i=1}^{m_i}|T_i|$为会员完成的总任务量，$\sum_i r_{ij}$为会员的总位置迁移量。

基于上述模型，我们可以设计动态决策模拟的算法进行近似求解。运用贪心算法的思想，在会员决策序列Q中,按照时序优先级与信誉值优先级从高到低的顺序遍历每个会员。对于每个会员，令其在运力约束下，选择任务量与迁移量的比值最大的任务组合，将其作为分配给该会员的任务包，按照这种方式完成遍历，即可得到总体的任务打包分配方案以及优化后的成交量与成交额。

5.3.2算法二

5.3.2.1 模型的建立

在问题 3 中指出，由于任务过于集中，附近的用户会出现相互竞争的现象，对此设计合理的打包模型。

1、一次打包模型：

我们引入 voronoi 图的概念，首先将用户进行聚类分析，分成 50 个中心区域，将各个区域内的用户看成一个聚类中心。将 835 个任务的 GPS 位置进行规划，分别落入 47 个 voronoi 域中。

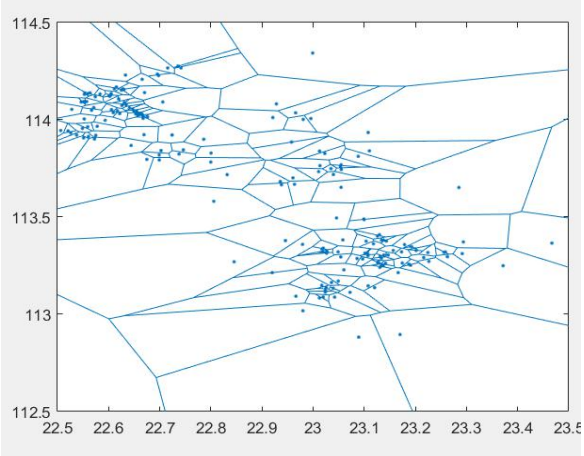


图 18 无边界 voronoi 图

引入任务点的坐标分布，有边界 voronoi 图如下所示：

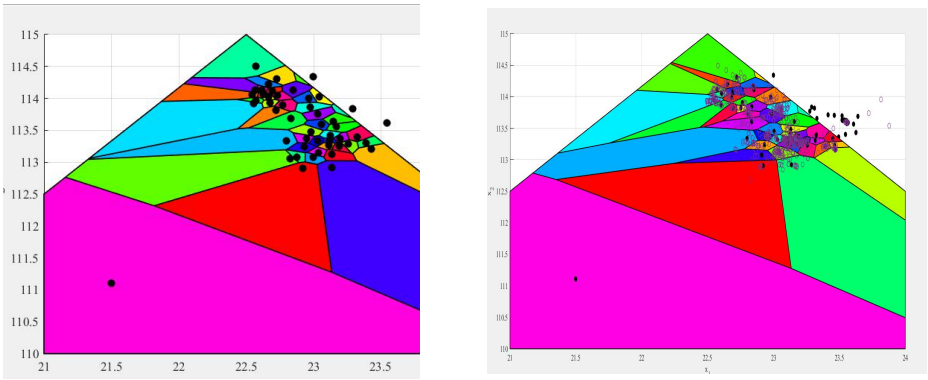


图 19 有边界 voronoi 图

2、二次打包模型：

我们对 835 个任务具体的 GPS 位置进行规划，分别落入 47 个 voronoi 域中。但在实际生活中，我们有必要将一个 voronoi 域中的所有任务做进一步的细化打包处理，通过商家角度和用户角度，我们可以得到如下的打包原则：

（1）从商家角度，依据使商家获益最大的原则，我们容易想到“捆绑”销售方法，利用第二问中的“心理预期价格/实际价格”作为会员心理性价比的判断准则，在同一个 voronoi 区域中，将性价比相对较高的和性价比相对较低的进行打包处理，使得商家获益最大。

（2）从用户角度，以用户拥有更高便捷体验和任务完成度为原则，我们选择“使得用户节省最多的距离成本”为原则，将能使用户节省最多距离量的任务进行打包。

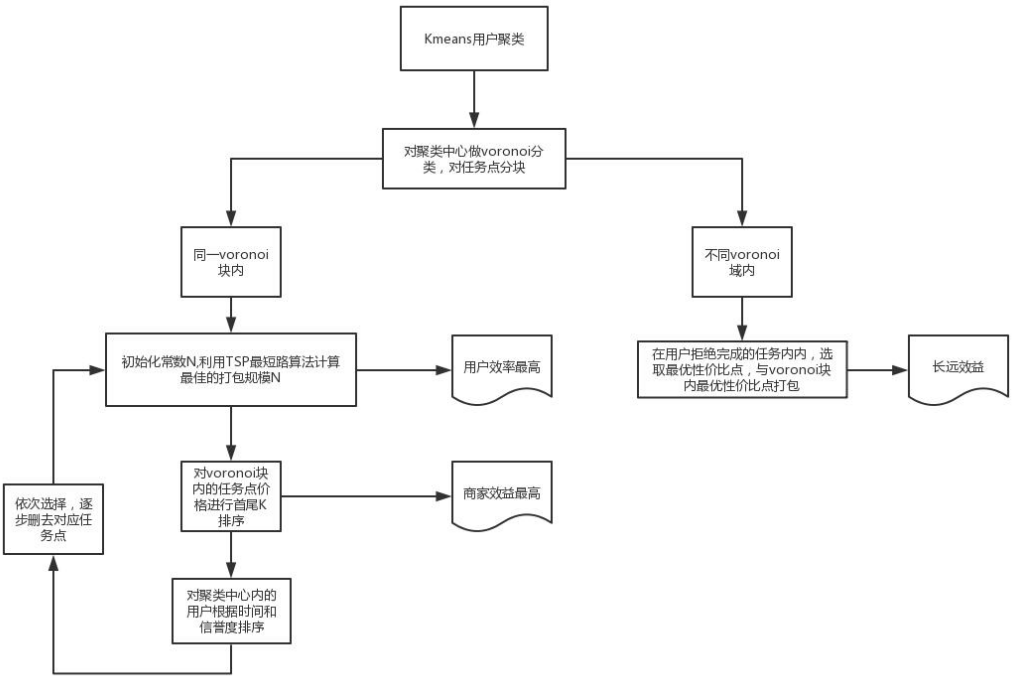
（3）从长远效益角度，为了增强“拍照赚钱”活动的影响力，我们采用如下两种促销方案：

• 递推式分析未完成和已完成的任务坐标布局，在两者的交界域进行动态打包，起到牵引作用。

• 对距离远的和距离近的任务进行整体打包，从整体上提高用户的活跃度。

5.3.2.2 模型的建立

在此，我们综合考虑上述三条打包原则，结合第二问中思路手段，设计了完整的实现流程：在对用户进行聚类分析后，引用 voronoi 图对其活跃范围进行分类，接着对在同一 voronoi 块内的任务和不同区块的任务实践具体的打包原则。具体算法流程图如下：



5.3.2.3 模型结果分析

1、在上述算法流程中，第一次打包后的结果如下，不同颜色代表不同的任务包：

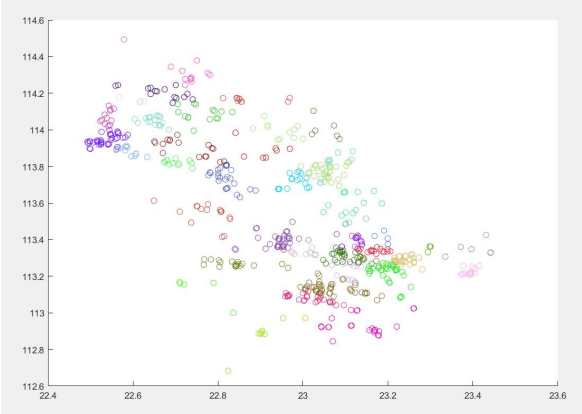


图 20 第一次打包

2、二次打包：

第一步：确定具体的打包规模：

我们根据 TSP 算法逐步优化，对给定的 voronoi 区块使用随机 K 值进行模拟，比较在最短路的原则约束下，任务打包能带给用户的距离节省量（采用几何中心模拟用户在随机移动下的平均位置）。选取其中节省量最大的作为最优 K 值。在此我们以第 10 个聚类中心为例：

（1）第 10 个 voronoi 域内的任务散点分布

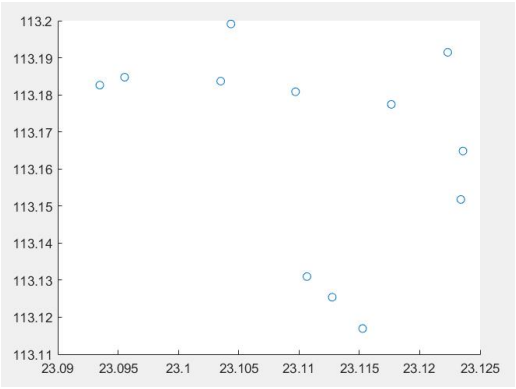


图 21 第 10 个 voronoi 域内的任务散点分布

（2）随机初始化

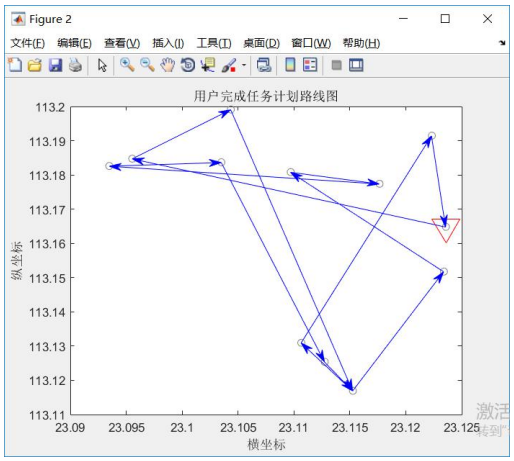


图 22 随机初始化结果

（3）TSP 算法进化过程

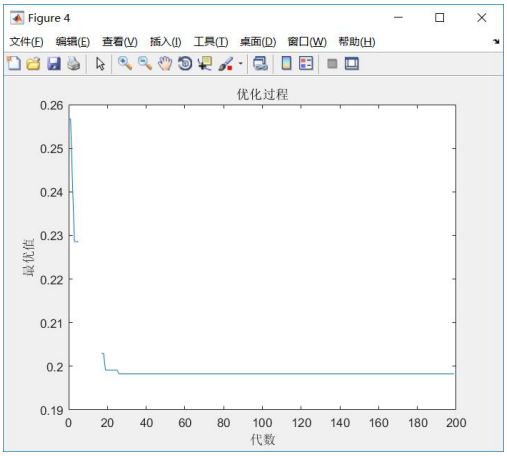


图 23 TSP 算法进化过程

（4）优化结果：

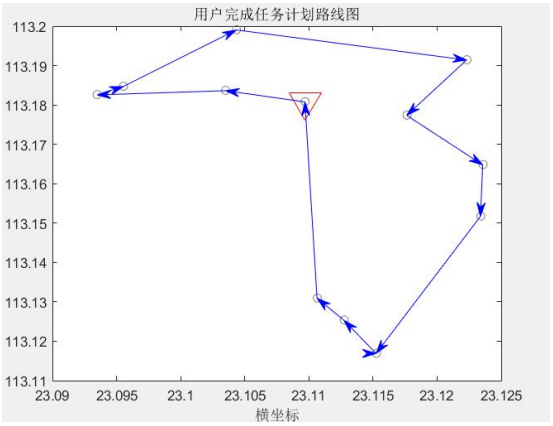


图 24 优化结果

（5）优化距离量

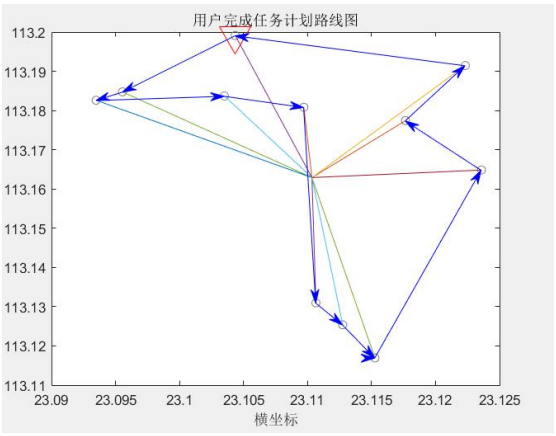


图 25 优化距离量

比较距离量，在第 10 个聚类中心，K=12 情形下，节省距离量为 0.2314.

（6）搜索后的最优 K 值

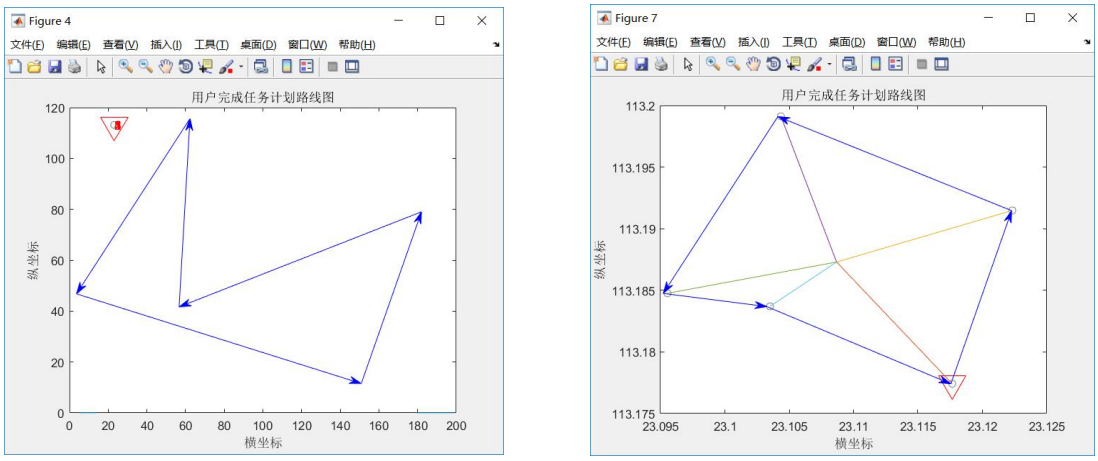


图 26 搜索后的最优 K 值

第二步：确定打包规模后，确定定价匹配方案：

在第 10 个 voronoi 域，确定打包规模是 5，进一步确定首尾的定价匹配算法，易得到最后的定价匹配策略：

表 2 定价匹配策略

23.1176526030152	113.177420842311	66.5000000000000
23.1223276296192	113.191485208337	67
23.1043540831342	113.199121080051	66.5000000000000
23.0955424458077	113.184764535666	75
23.1035079563968	113.183697797873	80

第三步: 从长远效益出发，选取在某特定用户拒绝的任务数组内的最优性价比点，与 voronoi 块内的最优性价比点进行打包。

第四步：具体结果显示

我们只给出了传出的样本打包示例，详细结果显示建附录

表 3 部分样本打包示例

593	0	561	666	543	798	658	104	384	188	17	119	216	244	173
586	0	562	664	573	603	683	93	197	402	357	110	434	224	304

5.4 问题四

5.4.1 利用问题二解决问题：

问题四给出了新的任务定位信息，要求我们根据前述模型与数据分析方法，给出新任务的定价方案，并评价其实施效果。

首先，我们可以对新任务的定位信息进行 kMeans 聚类分析，得到聚类散点图如下：

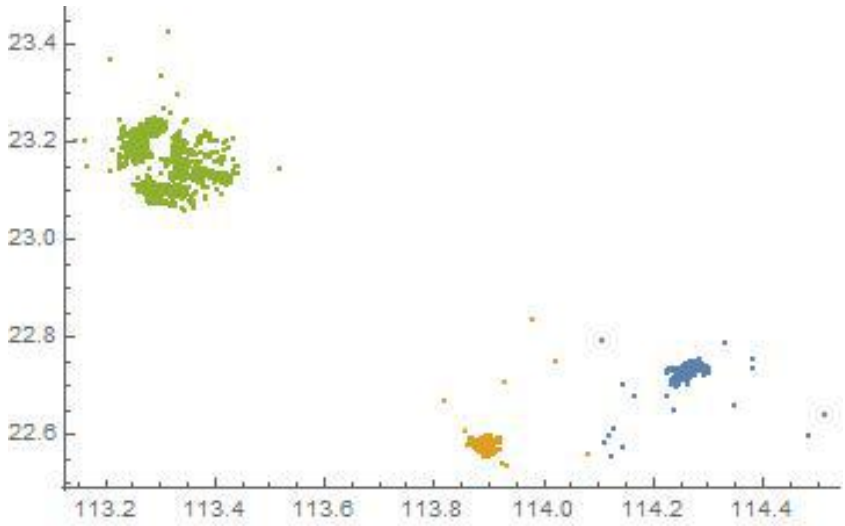


图 27 聚类分析结果

针对以上任务聚类，我们可以分别考虑无打包和任务打包后的定价策略。

在无打包的情况下，依据问题二建立的模型，调用相应算法，得到三个任务聚类地区各自的定价分布图像：

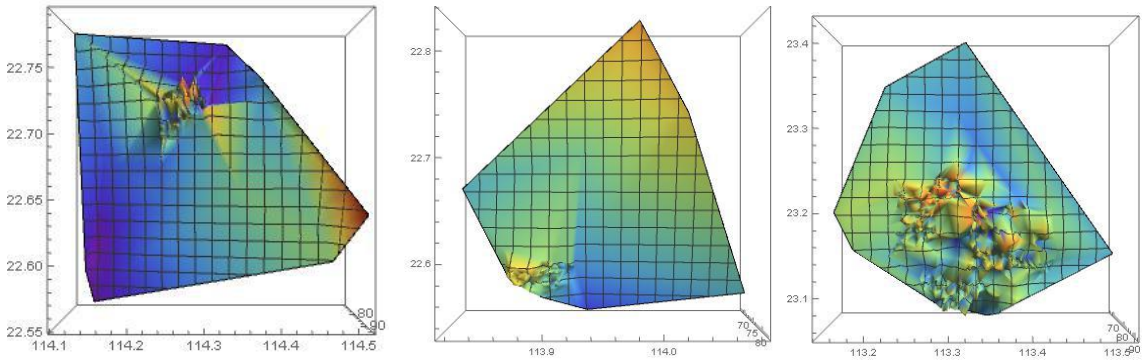


图 28 三个聚类区域的定价分布图像

总成交额为 118773 元，总成交量为 1506，平均成交价格为 78.87 元，成交率为 72.89%.

在模型二的算法框架下，距离会员特别近的任务被优先匹配，得到低成交价，由于新发布的任务过于集中，导致任务区域内的会员运力被迅速耗尽，这时平台方必须为外部迁入的用户运力支付高价，这就是上图中低价区域与高价区域交错分布的原因。

5.4.2 利用问题三解决问题

在我们使用问题二的定价算法框架解决问题时，出现了如上图所示的高低价格交替分布的问题，为解决问题，我们引入第三问的模拟打包模型进行辅助优化，得到的打包模型如图：

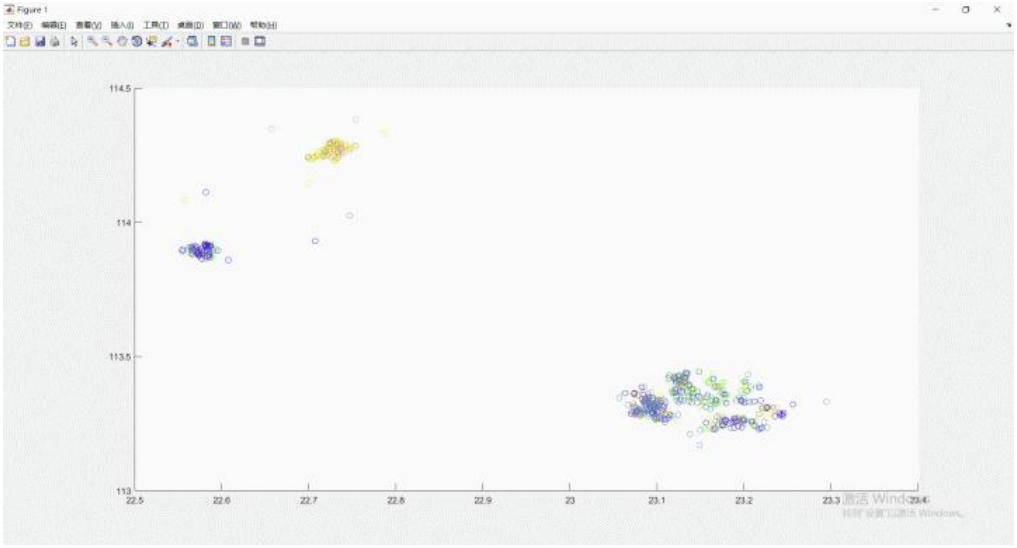


图 29 打包模型

在此基础上，将被打包的任务组当做整体任务继续使用第二问的算法框架进行分析，得的模拟效果如下，成交率为 89.32%。

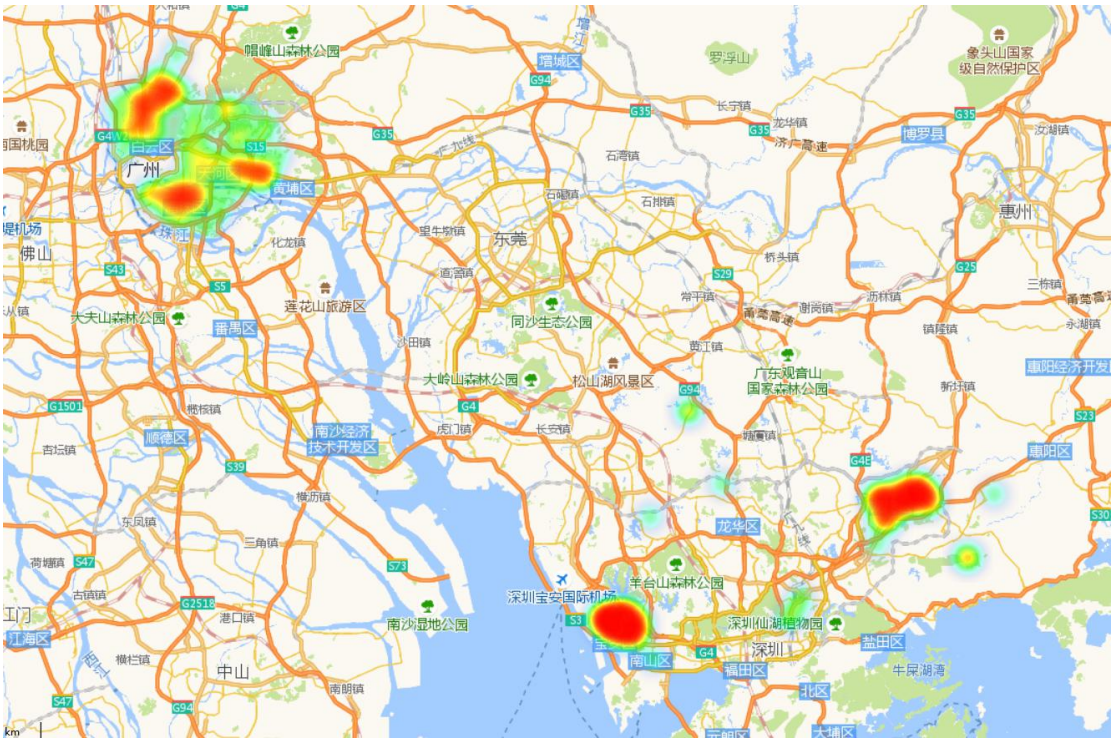


图 30 引入打包模型后的定价热力图

六、模型评价

6.1 优点

•在分析问题一任务未完成原因时，利用径向基神经网络得到了任务完成情况与四个影响因子的具体函数关系，并绘制出了任务完成情况与每个因素的关系图，分析结果有说服力。

•在问题二中，我们分别从企业和用户的角度出发得到了两种不同的算法，这两种算法都在保证任务完成度的情况下使企业的成本最低，得到的定价方案都显著优于原方案。

6.2 缺点

•由于我们利用神经网络训练得到的函数结构较为复杂，因此我们只分析了在其余影响因素适宜的情况下某一因素对任务完成情况的影响，而没有讨论不同因素之间的相互作用。

• 问题二中的优化算法由于利用神经网络，因此运行效率较低。

七、灵敏性分析

1、系统灵敏性分析：

在问题一和三中，我们同时运用了kmeans聚类分析。由于聚类函数SSE的非凸性，不能保证找到全局最优解，只能确保局部最优解。但是可以重复执行几次kmeans，选取SSE最小的一次作为最终的聚类结果。

在问题二的模拟优化方案中，我们采用了基于局部最优的遗传算法，易陷入局部最优解，对初值的依赖性较强，不够稳定。为提高系统的稳定性，我们可以将遗传算法与模拟退火算法相结合，利用模拟退火算法较强的局部搜索能力，使搜索过程避免陷入局部最优解，来弥补遗传算法的局部搜索能力差，并结合其总体搜索能力较强的特点，互相取长补短，则有可能开发出性能优良的新的全局搜索算法。

这种算法的伪代码如下：
这种算法的伪代码实现如下：
Step 1：进化代数计数器初始化：t←0。
Step 2：随机产生初始种群P(t)。
Step 3：评价种群P(t)的适应值。
Step 4：个体交叉操作：P’(t) ←Crossover[P(t)]。
Step 5：个体变异操作：P”(t) ←Mutation[P’(t)]。
Step 6：个体模拟退火操作：P”’(t) ← SimulatedAnnealing[P”(t)]。
Step 7：评价种群P”’(t)的适应值。
Step 8；个体选择、复制操作：P(t+1) ←Reproduction[P(t)U P”’(t)]。
Step 9：终止条件判断。若不满足终止条件，则：t←t+1，转到Step4，继续进化过程：若满足终止条件，则：输出当前最优个体，算法结束。

2、 函数灵敏性分析

在第一问中，我们首先采用了多元线性回归的模型去拟合实际定价，考虑到此模型与之后更为精确的神经网络模型在函数表达上有较大偏差，直接根据其多元回归的系数分析灵敏性并不准确，我们采用傅立叶幅度灵敏性检验法，我们在超平面上将线性回归函数转换为向量函数，进行傅立叶变换后分别计算各个元素的偏差和方差，得到会员密度，任务密度，任务距质心距离3个分量的灵敏性贡献率为34%，48%，18%，质心距离影响因子所占比重并不大，与根据系数大小的直观分析结果不同。

八、模型推广

本模型考虑了价格与多种影响因素的相关性，考虑了定价机理与买卖双方的动态决策、博弈过程，该模型可以推广用以解释各种多买家、多卖家同时动态决策的均衡价格形成机制。

在第三问的解决过程中，我们将允许打包策略时的定价问题转化为资源的分配与最优调度问题，因而该问题的求解过程也可用在一般的资源规划与调度问题上。

在本模型中，我们考虑了会员方、平台方的博弈过程。如果适当增加对于监管方决策方式的刻画，则本模型还可以用来解释在宏观调控下各种商品与服务的定价过程。

九、参考文献

无

十、附录

异常检测源代码：

```
% Data1=x1(1:825,1);%经纬度
% Data2=y1(1:825,2);
Data11=xlsread('fujian1');
Data1=Data11(2:826,1);
% Data22=xlsread('fujian2');
Data2=Data11(2:826,2);
```

```
Data=[Data1 Data2];
Data=Data';
dd=dist(Data);   %求距离矩阵
ndd=sort(dd,2);  %排序
p=0.005;        %异常比例
n=825;
k=round((1-p)*n);%%P 由用户自己定，
d=dd(:,k);%取第 K 列
D=quantile(d,1-p);
num=p*n;
outlier=zeros(2,825);
for i=1:n
    [row,col]=find(dd(i,:)>D);
    if sum(row)>num*50
        outlier(:,i)= Data(:,i);
    end
end
%  outlier=data(:,outlier);

% plot(Data(1,:),Data(2,:),'b','Markersize',14)
% hold on
% plot(outlier(1,:),outlier(2,:),'ro');
% diyige=(outlier(1,:))';
% dierge=(outlier(2,:))';
% scatter(diyige,dierge,'.')
number=0;
for m=1:825
    if outlier(1,m)~=0
%        scatter(outlier(1,m),outlier(2,m),'o')
       number=number+1;
    end
end


  temp=[];


  for m=1:825
      if outlier(1,m)~=0
         temp=[temp outlier(:,m)];
      end
  end
%   scatter(temp(1,:),temp(2,:),'.')
```

```
% plot(22.7117,114.4227,'.');
plot(temp(1,:),temp(2,:),'.')

temp
```

第一问源代码：

聚类分析：

```
%%获取经纬度
% [number,txt,raw]=xlsread('fujian1.xls');
A=xlsread('fujian1.xls');
X=[A(2:1:826,1) A(2:1:826,2)];
opts=statset('Display','final');

%开始调用 kmeans 函数
%X  保存的 K 个中心城市的 N*P 数据矩阵
%Idx   N*1，N 个点的聚类标号
%Ctrs  K*P 矩阵，储存 K 个质心的坐标
%SumD  1*K 的向量，所有点到这个点的距离之和
%D  N*K   N 个点每个点到质心的距离

[Idx,Ctrs,SumD,D]=kmeans(X,3,'Replicates',1,'Options',opts);

plot(X(Idx==1,1),X(Idx==1,2),'r.','MarkerSize',14)
hold on
plot(X(Idx==2,1),X(Idx==2,2),'b.','MarkerSize',14)
plot(X(Idx==3,1),X(Idx==3,2),'g.','MarkerSize',14)
  M=xlsread('fujian1.xls');
 Q1=[M(2:1:826,1)];
 Q2=[M(2:1:826,2)];

 for i=1:825
    if Idx(i)==1
       juli(i)=sqrt((Q1(i)-23.0095)^2+(Q2(i)-113.7245)^2);
    end
    if Idx(i)==2
       juli(i)=sqrt((Q1(i)-23.1113)^2+(Q2(i)-113.2288)^2);
    end
    if Idx(i)==3
       juli(i)=sqrt((Q1(i)-22.6676)^2+(Q2(i)-114.0416)^2);
    end

 End
```

网格化处理

```
function[Num]=wangge2()
data=xlsread('fujian2');
x1=data(1:length(data(:,2)),1);
y1=data(1:length(data(:,2)),2);

data1=xlsread('fujian1');
x11=data1(1:835,1);
y11=data1(1:835,2);

scatter(x1,y1,'.')
max1=max(x1);
min1=min(x1);
max2=max(y1);
min2=min(y1);
d1=(max1-min1)/500;
d2=(max2-min2)/500;
grid on
hold on
[x,y]=meshgrid(min1:d1:max1,min2:d2:max2);
plot(x,y,'k',x',y','k');
axis tight
Num=[]
number=0;
for m=1:length(x1)
    for i=0:500
        for j=0:500
            %%找到每一个任务，对应的网格区域

            temp=0;

            for p=1:length(x11)
                if
(x11(p)>=(min1+i*d1))&&(x11(p)<(min1+(i+1)*d1))&&(y11(p)>=(min2+j
*d2))&&(y11(p)<(min2+(j+1)*d2))
                    temp=1;
                end
            end



if(x1(m)>=(min1+i*d1))&&(x1(m)<(min1+(i+1)*d1))&&(y1(m)>=(min2+j*
d2))&&(y1(m)<(min2+(j+1)*d2)&&temp==1)
```

```
                %开始计数，先检查有没有任务点



                for t=1:length(x1)

if(x1(t)>=(min1+i*d1))&&(x1(t)<(min1+(i+1)*d1))&&(y1(t)>=(min2+j*
d2))&&(y1(t)<(min2+(j+1)*d2))
                        number=number+1;
                    end
                end
                Num=[Num,number];
                number=0;
            end
        end
    end
end
end
```

多元回归：

```
a1=load('huiyuanmidu.txt');
x1=a1(1,:);
x1=x1';
a2=load('renwumidu.txt');
x2=a2([1],:)
x2=x2';
a3=load('zhixinjuli.txt');
x3=a3([1],:);
x3=x3';
y1=xlsread('fujian1.xls');
y=y1(2:826,3);
zonghe=([x1 x2 x3])'
ym=xlsread('fujian1.xls');
target=(ym(2:826,4))'
X=[ones(length(y),1),x1,x2,x3];

[b,bint,r,rint,stats]=regress(y,X);
b;bint;stats;
rcoplot(r,rint)


% tx=[230.1,37.8,69.2];
b2=[b(2),b(3),b(4)];
ty=b(1)+b2*tx';
```

```
ty;
```

第二问源代码：

遗传算法主函数：

```
%% 清空环境
% clc
% clear
profile on
global k
global sum2
sum2=[];
for k=1:1:100
%% 遗传算法参数
maxgen=3;                          %进化代数
sizepop=2;                         %种群规模
pcross=[0.7];                      %交叉概率
pmutation=[0.01];
%变异概率
lenchrom=[];
bound=[];
for i=1:3
    lenchrom=[lenchrom 1];
% lenchrom=[1 1];                  %变量字串长度
% bound=[-5 5;-5 5];               %变量范围
   bound=[bound;65 85];
end

%% 个体初始化
individuals=struct('fitness',zeros(1,sizepop), 'chrom',[]);  %种群结构体
avgfitness=[];                                                %种群平均适应度
bestfitness=[];                                               %种群最佳适应度
bestchrom=[];                                                 %适应度最好染色体
% 初始化种群

for i=1:sizepop
    individuals.chrom(i,:)=Code(lenchrom,bound);       %随机产生个体
    x=individuals.chrom(i,:);

%    for k=1:1:3
```

```
    individuals.fitness(i)=funtesting(x,k)


%     end
    plot(i,individuals.fitness(i),'o')%个体适应度
end

%找最好的染色体
[bestfitness bestindex]=min(individuals.fitness);
bestchrom=individuals.chrom(bestindex,:);  %最好的染色体
avgfitness=sum(individuals.fitness)/sizepop; %染色体的平均适应度
% 记录每一代进化中最好的适应度和平均适应度
trace=[];

%% 进化开始
for i=1:maxgen

     % 选择操作
     individuals=Select(individuals,sizepop);
     avgfitness=sum(individuals.fitness)/sizepop;
     % 交叉操作

individuals.chrom=Cross(pcross,lenchrom,individuals.chrom,sizepop
,bound);
     % 变异操作

individuals.chrom=Mutation(pmutation,lenchrom,individuals.chrom,s
izepop,[i maxgen],bound);

    % 计算适应度
    for j=1:sizepop
        x=individuals.chrom(j,:);
        individuals.fitness(j)=funtesting(x,k)
    end

  %找到最小和最大适应度的染色体及它们在种群中的位置
    [newbestfitness,newbestindex]=min(individuals.fitness);
    [worestfitness,worestindex]=max(individuals.fitness);
    % 代替上一次进化中最好的染色体
    if bestfitness>newbestfitness
        bestfitness=newbestfitness;
        bestchrom=individuals.chrom(newbestindex,:);
    end
    individuals.chrom(worestindex,:)=bestchrom;
```

```
    individuals.fitness(worestindex)=bestfitness;

    avgfitness=sum(individuals.fitness)/sizepop;

    trace=[trace;avgfitness bestfitness]; %记录每一代进化中最好的适应度
和平均适应度
end
%进化结束
%% 结果显示
[r c]=size(trace);
figure
plot([1:r]',trace(:,1),'r-',[1:r]',trace(:,2),'b--');
title(['函数值曲线  ' '终止代数＝' num2str(maxgen)],'fontsize',12);
xlabel('进化代数','fontsize',12);ylabel('函数值','fontsize',12);
% legend('各代平均值','各代最佳值','fontsize',12);
ylim([-0.5 1000])
disp('函数值                  变量');
% 窗口显示
disp([bestfitness x]);
profile viewer
sum2(k)=x(1)+x(2)+x(3)
end
find(sum2==max(sum2))
```

调用函数：

```
function y = funtesting(x,k)
 bianliang1=xlsread('fujian1');%(需要随机化任务的 GPS)
%jiage=bianliang1(2:801,3);%任务价格需要随机化
% x=rand([799 1])*20+65;
renwuGPS1=bianliang1(2:4,1);  %任务的 gps
renwuGPS2=bianliang1(2:4,2);  %任务的 GPS
 bianliang2=xlsread('fujian2(更改)');%（不需要随机化会员的 GPS）
huiyuanGPS1=bianliang2(2:100,1);%会员的 GPS
huiyuanGPS2=bianliang2(2:100,2);%会员的 GPS
renwuxiane=bianliang2(2:100,3);
renwuGPS=[huiyuanGPS1 huiyuanGPS2];
y=0;
%首先进行排序，对时间排名靠前的，信用度排名靠前的依次靠前。
% for i=1:1876
for i=1:99
    %     k=10;
    xinlijiage=[];
    xinlibijiao=[];
    %     for j=1:799
```

```
    temp=3;
    for j=1:3
        if renwuxiane(j)~=0

xinlijiage(j)=65+k*((huiyuanGPS1(i)-renwuGPS1(j))^2+(huiyuanGPS2(
i)-renwuGPS2(j))^2);

          xinlibijiao(j)=xinlijiage(j)/x(j);

          if xinlibijiao(j)<1
             xinlibijiao(j)=xinlibijiao(j)-10000;
             temp=temp-1;
          end   %使得心理预期小于 1 的不参与考虑

          [row,col]=find(xinlibijiao==max(xinlibijiao));

y=y+sqrt((huiyuanGPS1(col)-renwuGPS1(col)).*(huiyuanGPS1(col)-ren
wuGPS1(col))+(huiyuanGPS2(col)-renwuGPS2(col)).*((huiyuanGPS2(col
)-renwuGPS2(col))));
          renwuxiane(col)=renwuxiane(col)-1;
       if temp==0
          y=y+100;
       end
       end  %对没有完成任何任务欲望的会员加上惩罚函数
    end
end
```

第三问源代码：

```
daoru11=xlsread('fujian2(更改)')
yonghuGPS1=daoru11(100:1001,1);
yonghuGPS2=daoru11(100:1001,2);
Options.plot=1; %设置 1 表示画出维诺图
v=[21 110;21 112.5;22.5 115;24 112.5;24 110];
P = polytope(v); %生成边界
Options.pbound=P;
%axis square;
yonghuGPSS=[yonghuGPS1 yonghuGPS2];
[idx,yonghuGPS]=kmeans(yonghuGPSS,50)
%根据用户中心给出的 5 个中心点

Pn=mpt_voronoi([yonghuGPS(:,1) yonghuGPS(:,2)],Options);
% V = extreme(Pn(2)) %这里的 V 就是第一个多边形的顶点序列
% voronoi(X,Y);
```

```
xlim([21 24])
ylim([110 115])

% V=[];
newjilu=cell(50,1)  %5个多边形
daoru22=xlsread('fujian1');
renwuGPS1=daoru22(1:835,1);
renwuGPS2=daoru22(1:835,2);
renwujiage=daoru22(1:835,3);

temp=[];
for i=1:1:835

    for j=1:1:47  %5个多边形
        temp=extreme(Pn(j));

in=inpolygon(renwuGPS1(i),renwuGPS2(i),temp(:,1),temp(:,2));
        if in==1
            newjilu{j}=[newjilu{j} i];
            break
        end



        %         [hang lie]=size(temp);
        %
        %         for k=1:1:hang
        %
        %             V=[V extreme(Pn(i))];
        %         end
    end
end
% scatter(renwuGPS1,renwuGPS2)
%首先将性价比最高的和性价比最低的一起打包
%聚类中心的心里预期价格是65+54*平均距离
dabao=cell(50,1);
% dabao=zeros(50,2);
xingjiabi=cell(50,1);
ercidabao=zeros(50,2);
for i=1:1:50
    yuqijiage=[];
    if length(newjilu{i})==0
        continue
    end
```

```
    for j=1:1:length(newjilu{i})     %513 的序号值对应 515 行
        %
distance=abs(renwuGPS1(newjilu{j})-yonghuGPS(i,1)).*abs(renwuGPS1
(newjilu{j})-yonghuGPS(i,1))
        %
+abs(renwuGPS2(newjilu{j})-yonghuGPS(i,2)).*abs(renwuGPS2(newjilu
{j})-yonghuGPS(i,2));
distance=(renwuGPS1(newjilu{i}(j))-yonghuGPS(i,1)).^2+(renwuGPS1(
newjilu{i}(j))-yonghuGPS(i,1)).^2
        distance
        %确定了心里预期价格的表达式
        xingjiabi{i}=[xingjiabi{i}
distance/((renwujiage(newjilu{i}(j))))];
        %
yuqijiage(j)=(65+54*distance)/(renwujiage(newjilu{i}(j)));
        %
dabao{i}=[find(xingjiabi{i,:}==max(cell2mat(xingjiabi{i})))]
        dabao{i}=[max((xingjiabi{i})) min((xingjiabi{i}))]
ercidabao(i,:)=[newjilu{i}(find(xingjiabi{i}==max(xingjiabi{i})))
newjilu{i}(find(xingjiabi{i}==min(xingjiabi{i}))) ];
    end
    %    dabao{i}{1}=[find(xingjiabi(j)==max(yuqijiage))
find(yuqijiage(j)==min(yuqijiage))];
    %
dabao{i}=[find(xingjiabi{i,:}==max(cell2mat(xingjiabi{i})))]
end
%%打包过程已经解决
% zhuanhua = [newjilu{:}];
% zhuanhua = reshape(zhuanhua,length(zhuanhua)/3,3);
% xlswrite('duru1.xls',zhuanhua);



%%
 daoru1=xlsread('newfujian2(paixu)')
daoru2=xlsread('fujian1')
renwuGPS1=daoru2(2:31,1)
renwuGPS2=daoru2(2:31,2)
X=[renwuGPS1 renwuGPS2]

figure;
plot(X(:,1),X(:,2),'.');
title 'Randomly Generated Data';
opts = statset('Display','final');
```

```
[idx,C] = kmeans(X,2,'Distance','cityblock',...
    'Replicates',5,'Options',opts);


figure;
plot(X(idx==1,1),X(idx==1,2),'r.','MarkerSize',12)
hold on
plot(X(idx==2,1),X(idx==2,2),'b.','MarkerSize',12)
plot(C(:,1),C(:,2),'kx',...
     'MarkerSize',15,'LineWidth',3)
legend('Cluster 1','Cluster 2','Centroids',...
      'Location','NW')
title 'Cluster Assignments and Centroids'
hold off

%根据用户优先级排列，选择对应的聚类集合
   yonghuGPS1=daoru1(2:201,1);
   yonghuGPS2=daoru1(2:201,2);
   yonghuGPS=[yonghuGPS1 yonghuGPS2];


 % 根据点 绘制泰森多边形
   taisenx=yonghuGPS1';
  taiseny=yonghuGPS2';
  length=
voronoi(taisenx,taiseny);
设定 x 轴的边界
xlim([22.5 23.5]);
设定 y 轴的边界
ylim([112.5 114.5]);
axis equal



对聚类结果进行标号
 julei11=[];
 julei12=[];
 julei1=[julei11 julei12];
 julei21=[];
 julei22=[];
 julei2=[julei21 julei22];
   for i=1:1:30
       if idx(i)==1
        julei11=[julei11 renwuGPS1(i)];
```

```
       julei12=[julei12 renwuGPS2(i)];
      end

      if idx(i)==2
         julei21=[julei21 renwuGPS1(i)];
         julei22=[julei22 renwuGPS2(i)];
      end
   end

  [m1,n1]=size(find(idx==1));
  [m2,n2]=size(find(idx==2));
  yonghuduiying=[];
  yonghujuli=[];
   for i=1:1:200
  temp1=[];
  temp2=[];

      for j=1:1:n1

         temp1=[temp1
sqrt((yonghuGPS1(i)-julei11(j))^2+(yonghuGPS2(i)-julei12(j))^2)];

      end

       mintemp1=min(temp1);
      对每个给定用户，此时的 temp1 代表其到聚类 1 的距离

      for j=1:1:n2
         temp2=[temp2
sqrt((yonghuGPS1(i)-julei21(j))^2+(yonghuGPS2(i)-julei22(j))^2)];

      end
      mintemp2=min(temp2);
        对每个给定用户，此时的 temp2 代表其到聚类 2 的距离

        yonghujuli(i)=min(mintemp1,mintemp2)

      if mintemp1>=mintemp2
         yonghuduiying(i)=1;
      end

      if mintemp1<mintemp2
         yonghuduiying(i)=2;
      end
```

```
   end

   yonghuduiying
yonghujuli
```
此时做出了用户选择的聚类中心的距离数组

接下来计算节省的距离，以前是选择一个

```
 scatter(yonghuGPS1,yonghuGPS2,'.')

  根据点 绘制泰森多边形
  taisenx=yonghuGPS1';
  taiseny=yonghuGPS2';
  length=
voronoi(taisenx,taiseny);
```
设定 x 轴的边界
```
xlim([22.5 23.5]);
```
设定 y 轴的边界
```
ylim([112.5 114.5]);
axis equal
```