基于聚类分析的任务定价与分配模型

摘 要

“拍照赚钱”的任务定价问题会直接影响到任务完成情况，因此需要在定价与完成率中找到平衡，使得项目方能低成本、高效率地完成商品的信息收集。本文根据具体要求，建立相对应的数学模型，给出题目所需的定价方案。

问题一中，我们根据所提供信息进行合理推测，进而验证假设。根据所给数据，我们推测定价与任务位置，会员位置，会员信誉值有关，利用地图软件、origin、SPSS等途径，通过定性分析方法，得出结论：会员分布越密集，任务定价越低，任务完成情况较差。同时结合会员和任务所在地的实际情况，分析出任务未完成的原因：1、 定价与当地经济水平不匹配；2、部分任务难度过高；3、部分区域内会员数量不够。

问题二中，我们所设计的新的任务定价方案需要建立以完成任务量最大且总定价最低的双目标规划模型。通过Haversine公式利用经纬度得出任一任务和任一会员间距离，进而设定会员完成任务的能力和任务对于会员的性价比，根据会员开始任务的时间顺序，利用matlab的搜索算法对某一任务的范围圆内的会员进行遍历，即任务与会员的匹配。通过设定的会员完成任务的可能性与阈值的比较，得到任务的完成情况。与原方案比较，此定价方案的完成率为88.38%，较原定价方案提高了28.62%，同时总定价减少了4.2%。

问题三中，在分布较为集中的任务打包发布的情况下更新以完成任务量最大且总定价最低的双目标规划模型。根据任务的位置信息，利用DESCAN聚类算法的思路进行打包方案，可以得出176个打包组合。将打包后的任务视为完整体，可以得出打包组合的中心点和总定价，继而得到打包组合下的会员完成能力和任务性价比。基于会员开始任务的时间先后顺序，利用matlab的搜索进行搜索匹配。将更新后的会员完成任务的可能性与阈值进行比较，得出任务完成情况。在打包定价模型下，任务的完成率为90.12%，较于未打包情况下提升了1.74%，同时总定价减少了3.14%。

问题四中，新项目的任务定价方案采用打包定价模型方案。对于给定的任务位置，利用DESCAN聚类算法，可以得出邻域半径约为100m的385个打包组合。进而采用求解打包定价模型的方法，得出新项目的任务定价方案，在该方案下，新项目的任务完成率为91.56%

关键词：任务分配 双目标规划模型 DESCANJU聚类算法 层次分析法

# 问题重述

## 问题重述

随着移动互联网的发展，企业通过让用户成为APP会员来完成相应的“拍照赚钱”任务成为趋势，这种众包平台为企业提供了各类商品的检查数据和信息收集服务，不仅能显著降低调查成本，还能有效确保数据的真实性、缩短数据收集调查的时间。所以，APP是这其中的核心所在，对于某一任务的定价更是关键因素。故而我们需要进行合理的定价才能尽可能多的完成商品的检查。

## 问题要求

问题设置层层递进，其本质绕不开对于“拍照赚钱”的定价问题。

问题一：根据题目所给以完成项目的一系列相关数据，分析影响定价的相关因素，继而寻找其规律，并依据实际分析任务未完成的原因。

问题二：给已完成的项目重新安排任务定价，并将其与旧方案进行比较。

问题三：考虑到实际生活中，会员可能会因为数个任务位置集中而争相选择，有一种解决方式是将有上述特征的任务合并发布。结合该情况，修改原定的定价模型，并分析修改后的定价模型的任务完成情况的变化。

问题四：给新项目制定任务定价方案，且评价最后的实施效果。

# 问题分析

## 问题一的分析

第一问要求研究一份已完成的项目中任务定价的规律。首先，这里我们根据题目所给出任务的位置、定价和完成情况，以及会员位置、信誉值等数据，合理的猜测任务的定价与其位置、会员位置及信誉值有关。其次我们借助地图、origin、SPSS等软件直观地将任务和会员的位置，以及任务完成程度表示出来，同时可以发现会员与任务集中分布于四个城市。进而借助实际情况逐一分析四个城市的情况，即可得出任务定价规律及部分任务未完成的原因。

## 问题二的分析

第二问的设计新任务定价问题实际上是一个多目标规划的问题。从项目方角度出发，一个合理的定价方案，应在任务总体完成度尽可能高的同时让任务的总定价尽可能低，因此，可以将该问题定义为以完成任务量最大且总定价最低的双目标规划模型。

假设项目方能直接将任务分配给会员，当以每一任务为中心设置大小相同的范围圆后，定价问题在任务分配阶段属于无权重变量，由于每个会员都有不同的信誉值、开始任务时间及与任务的距离，可以利用matlab、excel等途径找出会员完成任务的能力和完成任务的性价比函数，同时根据时间批次分配任务。另外在关于任务完成的定义，利用matlab算出距离、定价和会员完成任务能力三者对于完成任务可能性的权重，进而构建其函数，同时通过原定价方案设立可能性阈值，通过与阈值的比较即可判断任务完成情况。从而确认双目标规划模型的约束条件，进而求解。

## 问题三的分析

第三问实际上类似于第二问，属于多目标规划模型。基于第二问下的无打包定价模型，需要增加联合打包条件的打包定价模型。因此，需要基于任务的经纬度信息和密集程度，给出一个合理的打包方案，利用DESCAN聚类算法的思想，对任务进行分类，确认一个较为合理的打包方案。依据打包方案将任务完成打包后，可以确认出打包后各任务的中心点，和打包任务总定价，同时结合会员的完成能力、会员与任务中心点间距离和会员完成任务的可能性，从而求解出目标函数，即任务的定价方案和完成情况。

## 问题四的分析

第四问要求给新项目设计任务定价方案。将新项目的任务位置导入地图发现，新项目任务仍集中分布于广佛莞深四个城市，且密集程度明显高于第一个项目，同时根据问题三和问题二的模型比较结果，认为打包定价模型具有更高的效益，因此采用打包定价模型。利用DESCAN聚类算法，求解打包方案。仿照问题三打包定价模型的求解，即可得出最终的定价方案和任务完成情况。

# 模型假设

1. 任意两点间的实际路程用直线距离近似代替；
2. 各会员在分配任务后立即前往完成任务；
3. 信誉值越大，完成任务概率越大；
4. 任务发布者能够直接将任务分配给会员；
5. 所有任务的分配在同一周期内，且同一周期内只能分配一次；
6. 若会员被分配到某一打包任务集，则该打包任务集内各任务完成情况一致。

# 符号说明

|  |  |  |
| --- | --- | --- |
| **符号** | **说明** | **单位** |
|  | 任务i是否被会员j完成 | / |
|  | 任务i与会员j的距离 | Km |
|  | 会员i的定价 | 元 |
|  | 在城市k中的任务i范围圆中会员j的信誉值 | / |
|  | 会员j完成任务i的可能性 | / |
|  | 打包后第k组任务的总定价 | 元 |

# 模型的建立与求解

## 问题一求解

### 任务定价的规律

对于附件一中所给的任务的经纬度、定价和完成情况以及附件二中给出的会员位置等的数据和信息，我们合理猜测任务定价与任务自身位置和会员分布有影响。

对已知数据进行分析，将附件一中任务的经纬度位置导入地图进行标注，如图1所示，可以发现任务分布集中分布在广东省的广州市、佛山市、东莞市、深圳市这四个城市。

将附件一中任务的定价根据其经纬度，利用origin可以作出任务定价的分布图，由图2可以看出广州与深圳的定价普遍低于东莞和佛山。

其中绿色代表任务完成，红色表示任务未完成；绿色区域为佛山市，蓝色区域为广州市，黄色区域为东莞市，红色区域为深圳市。

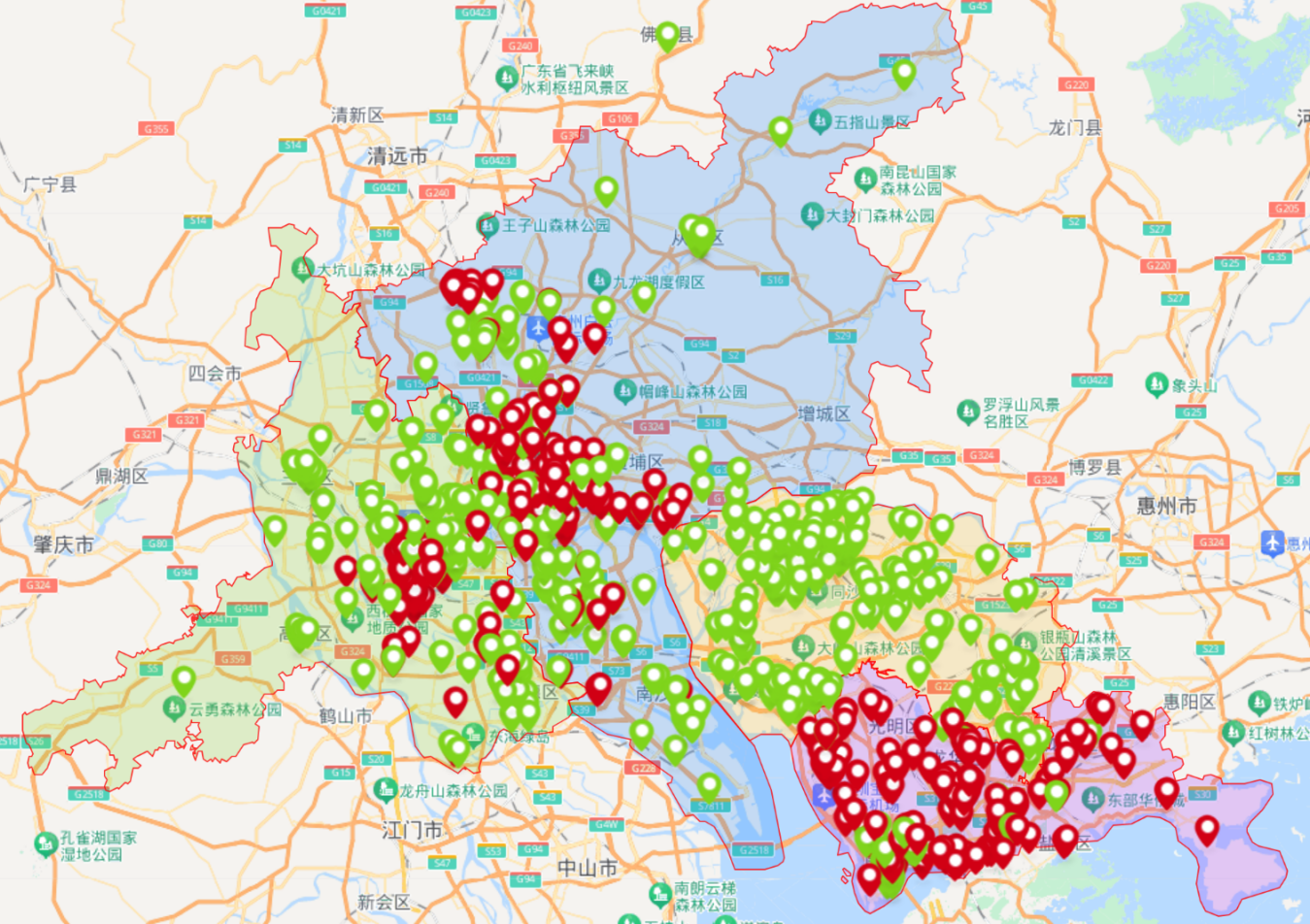


图 1任务分布图

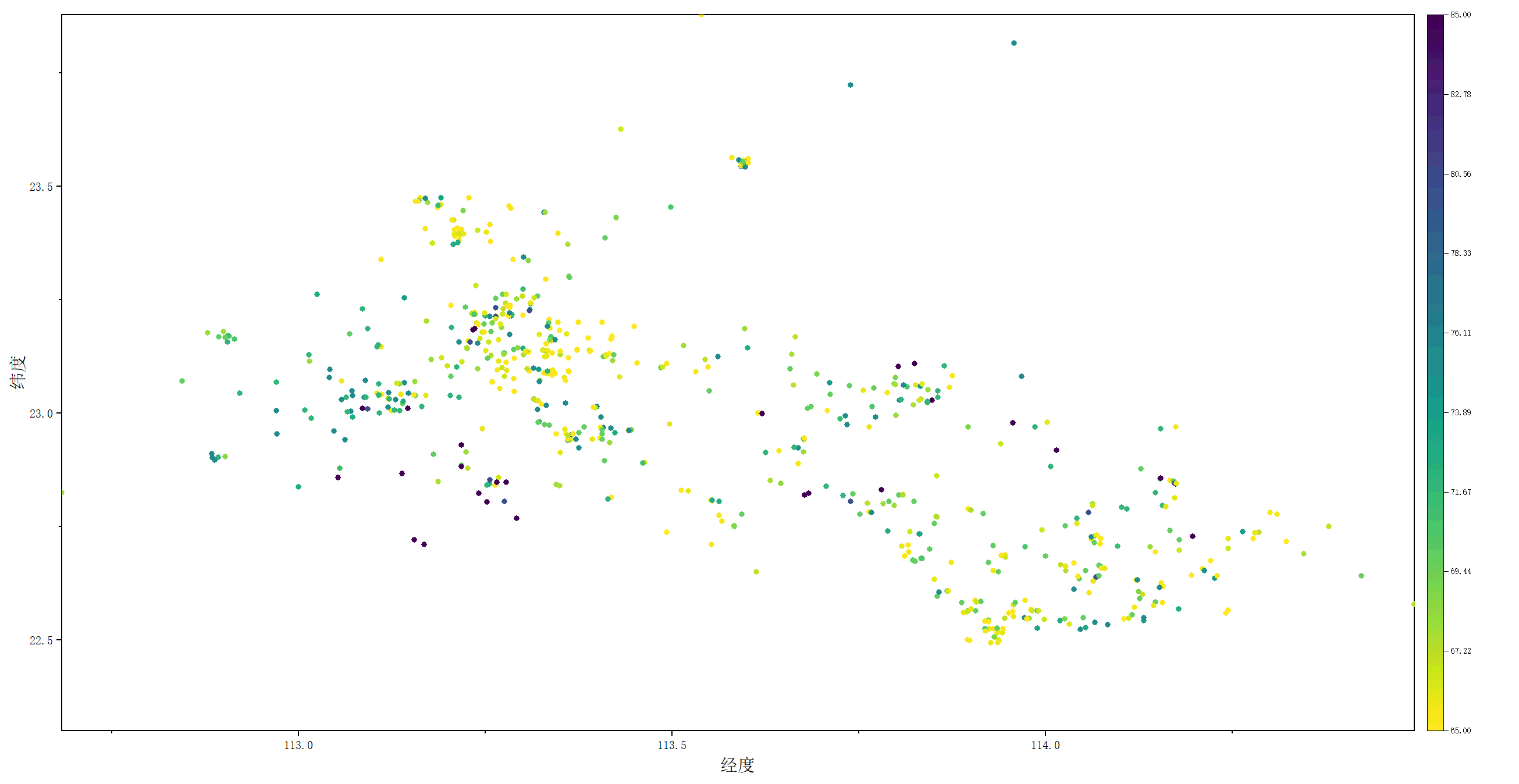


图 2任务定价分布图

将附件二中会员位置也导入地图中，得到图3，可以看出广州市和深圳市会员数量远多于佛山市和东莞市。

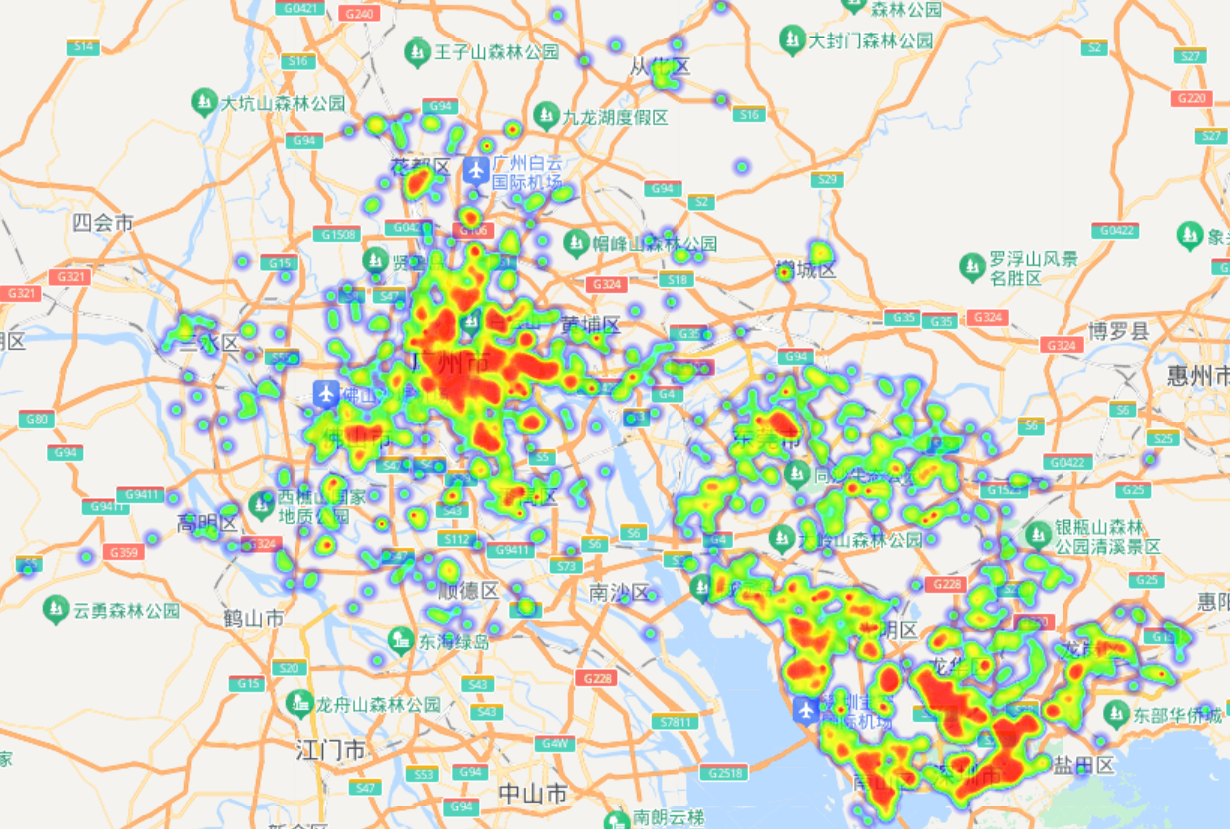


图 3会员分布图

借助SPSS可以得出四个城市任务的总体完成情况，其中东莞完成率最好，所以我们将东莞的完成率设为1，将对应的得出佛山为0.66，广州为0.61，深圳为0.20。

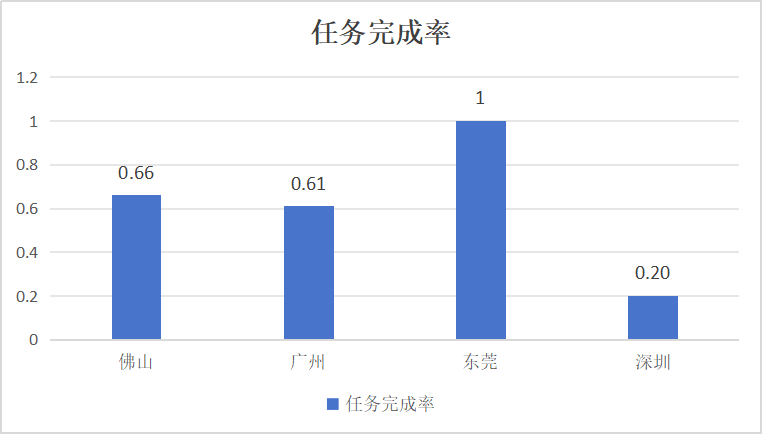


图 4四个城市的任务完成率

综上分析，东莞的完成率最好，而该地的会员分布较少，定价较高；深圳的完成率最差，该地的会员分布较密集且定价较低。因此可以得出结论：

会员分布越密集，任务定价越低，任务完成情况较差。

### 任务未完成原因分析

从图1可以看出，未完成的任务分布较为集中。可以归纳为以下三点原因。

1. 定价与当地经济水平不匹配。GPD表明一所城市的经济水平，对于经济发展水平较高的广州天河区、海珠区和深圳的绝大部分区而言，任务定价却普遍低于东莞和佛山，因此在这些区域就会出现对会员吸引力不够的问题，导致广州、深圳中心地区任务完成度较差。
2. 部分任务难度过高。部分任务位于大型景点附近，例如广州市白云区的白云山、深圳市的阳台山，出现周边环境可能有交通不便、挑战过大等问题，导致任务性价比不高，因此会员不愿选择该任务，或者任务无法及时完成。
3. 部分区域内会员数量不够。在一些区域内任务多且密集，但附近会员人数不足以应对，例如广州市花都区和佛山市禅城区。

## 问题二模型的建立与求解

### 问题二的分析

首先，我们要明确我们所设计的新的定价方案需要尽可能的解决问题一中任务未完成的缺点，即要使任务价格总和尽可能小的同时使任务的总完成率尽可能大。接着，我们延续问题一思考影响任务定价的思路，我们需要考虑的影响因素有：

1. 任务与会员的距离问题，我们将这部分定义为会员接受任务的意愿程度。
2. 会员的信誉值，可以将其视为会员完成任务的能力。
3. 任务的密集程度，即任务被竞争的程度。
4. 会员的密集程度。
5. 会员开始任务的时间，我们将同一时间开始任务的会员视为同一周期。

综上，我们建立新的多目标规划的任务定价模型。

### 问题二模型的建立

我们将设计定价方案分为任务分配阶段与任务完成阶段，为确保每个任务点都能被考虑到，我们在以每个任务点为圆心，半径为5km的寻找任务的最佳定价方案，及最优完成途径。

第一阶段：任务分配

在该阶段，我们以每个任务点为中心，检索周围5km内的会员分配的可能和会员已接受的任务数。所以第一步我们需要计算出会员与任务间距离。

通过Haversine公式（如式1），利用附件一和附件二中任务与会员的经纬度，可以计算出各任务与各会员间的距离。

|  |  |
| --- | --- |
|  | () |

其中，lat表示纬度，lon表示经度。

将任务点周围的会员根据任务开始时间分为不同批次，从最早批次开始进行匹配，直至会员时间批次为最后一批，即视为任务分配结束。

对于该任务点的定价问题，首先需要判断原方案中任务是否被完成，若完成，即在原定价基础上减价；若未完成，即在原定价基础上加价。

借助origin软件利用会员的位置作出会员的关于经度和维度的直方图，如图可以看出会员的经度和纬度都符合正态分布，即满足拉依达准则的条件。我们认为满足与算术平均值之差的绝对值大于三倍的标准偏差的数据是含有粗大误差值的坏值，应予剔除，利用matlab软件，通过拉依达准则所剔除的粗大误差有B005、B007、B0022、B0039、B0048、B0082、B0136、B0472、B1708、B1727、B1822，以及由于经纬度颠倒的B1175需要被剔除。

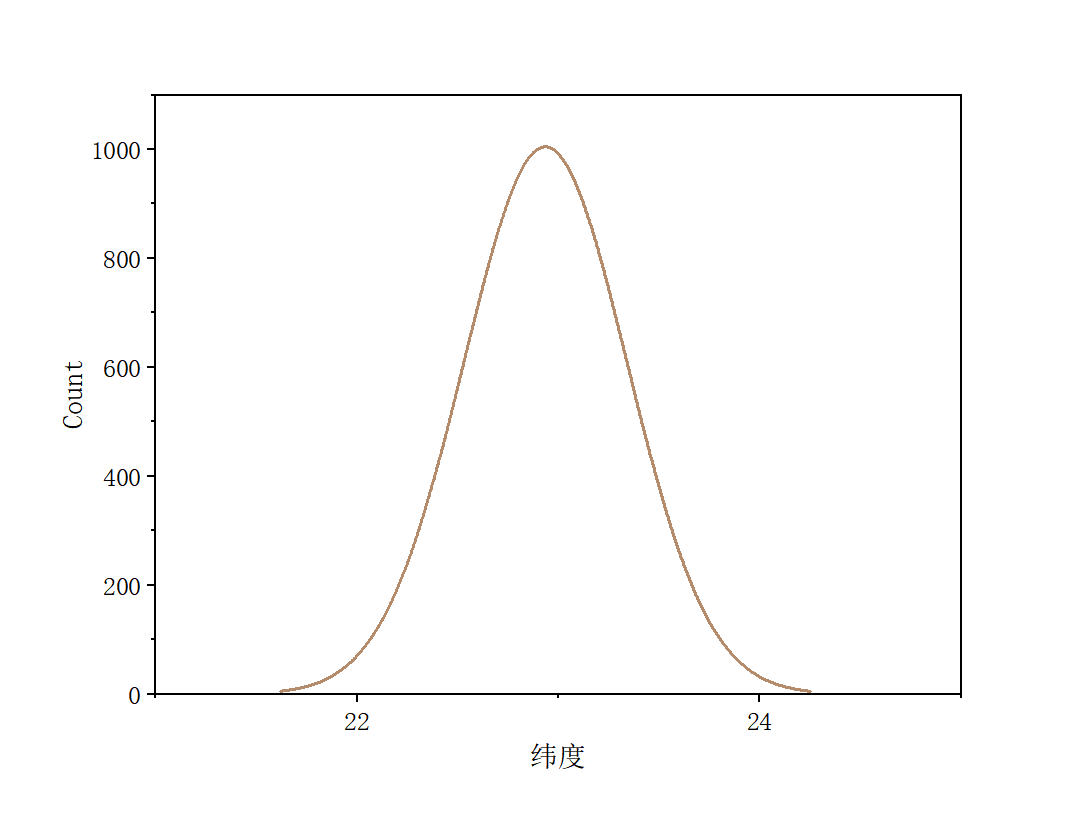


图 5会员位置关于经度和纬度的直方图

对于以任务i，会员的信誉值越大，该会员完成任务i的能力越大，我们将会员j完成任务的能力定义为：

|  |  |
| --- | --- |
|  | () |

其中，表示k市中会员j的信誉值，表示k市中在任务i的范围圆内的最大信誉值。式（1）表示在以任务i的范围圆内，每个会员完成任务i的能力。当该范围内某一会员的信誉值为最高时，常数C的引入使得会员完成任务的概率会趋于100%。

在任务与会员进行匹配时，需要将任务分配给范围圆内已接受任务书未超过限额且匹配性价比最高的会员，我们将任务i对于会员j的性价比定义为：

|  |  |
| --- | --- |
|  | () |

其中，表示任务i与会员j的距离。当对于同一任务时，任务的分配与距离和会员完成该任务的能力有关。

考虑到最终任务完成率的问题，定义会员j完成任务i的可能性为：

|  |  |
| --- | --- |
|  | () |

其中，表示任务i的定价、任务i的定价和会员j的完成任务的能力与完成任务可能性成正比，两者间距离与其成反比。

考虑到四个城市的经济发展问题，我们定义城市k完成任务的可能性的阈值为：

|  |  |
| --- | --- |
|  | () |

其中，将阈值设置为城市k中原方案中已完成任务的最低定价与未完成任务的最高定价的平均值，当会员j完成任务的可能性大于所在城市的阈值时，该任务即会被完成，反之则未完成。且最终无论完成与否，该任务都会被剔除。

第二阶段：任务完成

1. 目标函数的确认

一个合理的定价方案，最关键的是要保证任务的完成率尽可能大的同时总定价尽可能低。因此得出以完成任务量最大且总定价最低的目标函数：

|  |  |
| --- | --- |
|  | () |

1. 约束条件的确认

约束条件一：在范围圆内，根据定价、距离和会员完成任务能力三者的权重可以得出，想要让信誉值过低的会员完成某一任务，需要使任务与会员的距离极近，或任务定价远高于其他任务定价，可能会导致会员完成该任务的性价比不高。因此：

|  |  |
| --- | --- |
|  | () |

约束条件二：对于一个合理的定价方案，应尽可能的让更多的会员参与到任务完成中，因此需要设置会员接受任务的限额。即：

|  |  |
| --- | --- |
|  | () |

同时，将会员j已接受任务的数量可视化，以便计算。因此：

|  |  |
| --- | --- |
|  | () |

约束条件三：任一任务至多只能被完成一次。因此：

|  |  |
| --- | --- |
|  | () |

综上所述，我们建立的多目标规划模型为：

|  |  |
| --- | --- |
|  | () |
|  | () |

### 问题二模型的求解

Step 1：的求解

由式(4)的会员j完成任务i的可能性函数可以得出会员完成任务可能性与任务定价，任务与会员间距离，会员完成能力具有相关性。建立层次分析模型，计算任务定价，任务与会员间距离，会员完成能力的权重值。

在该步骤中的目的是寻找准则层的权重，所以在评价体系中方案层可省略。

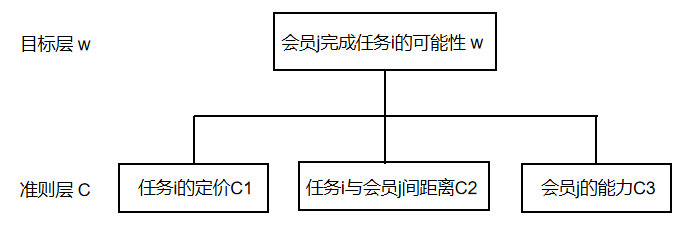


图 6评价体系

建立任务5km以内会员j完成任务可能性的判断矩阵：

|  |  |  |  |
| --- | --- | --- | --- |
|  | 定价 | 距离 | 能力 |
| 定价 | 1 | 1/2 | 1/9 |
| 距离 | 2 | 1 | 1/8 |
| 信誉 | 9 | 8 | 1 |

利用matlab软件，通过算术平均法可以求出三者权重为：

|  |  |  |  |
| --- | --- | --- | --- |
|  | 定价 | 距离 | 能力 |
| 权重 | 0.0753 | 0.1244 | 0.8004 |

利用matlab软件，通过特征值法求三者权重：

|  |  |  |  |
| --- | --- | --- | --- |
|  | 定价 | 距离 | 能力 |
| 权重 | 0.0738 | 0.1218 | 0.8024 |

并且可以计算的判断矩阵的最大特征值为：=3.0369

根据最大特征值计算一致性指标：。其中判断矩阵为三阶矩阵，于是这里的n=3。因此算得：

CI=0.0184

为计算一致性比例，根据一致性指标表，找出相应的平均随机一致性指标RI。可以得到：RI=0.52。

最后，结合上述数据可以计算出一致性比例：

CR=0.0355

当CR≤0.1时，认为判断矩阵的一致性是可以接受的，即可以采用特征值法下得出的三者权重值。

因此，为使最后确认的权重更具有准确性，我们取两种方法下得到的权重平均值作为最终权重，即：



Step 2：的求解

采用与相同的思路，即通过算术平均法和特征值法分别计算相应的权重，最终权重为二者均值。因此可以得到：



Step 3:各城市阈值的求解

根据前言，可以知道会员与任务锁定于广州，深圳，佛山和东莞。即让k=1,2,3,4与四个城市一一对应。

利用任务的经纬度将任一会员和任务具体确认在某一城市中，通过matlab软件，可以找出每一城市中原定价下未完成任务的最高定价和已完成任务的最低定价，根据式（5）可以得出：



Step 4：从各城市中最低定价的任务开始与第一批次开始任务的会员进行匹配，即将该任务分配给对应范围圆内6：30开始任务，且最高的会员。将原方案中未完成的任务和已完成的任务分别增加和减少一定的价钱，同时根据式（4）所得的w与进行对比，若高于阈值，则该任务视为完成，则不再参与后续匹配，同时根据阈值、距离、会员完成能力，即可得到该任务的定价；若低于阈值，则该任务未被完成，同时不再参与后续时间批次的会员匹配。

各城市中最低定价任务与第一批次会员匹配的任务完成情况，其中完成用“1”表示，未完成的任务用“0”表示，如图7：

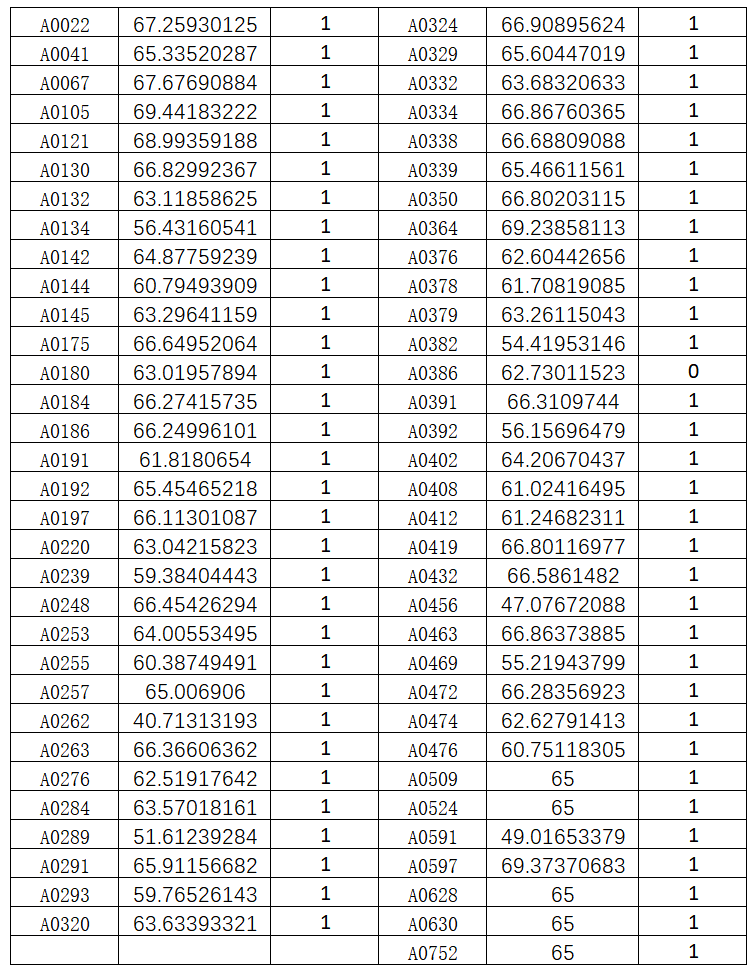


图 7第一批次任务的最终定价和完成情况

Step 5：将城市中的任务从低到高依次与不同批次进行步骤4，最终即可得出城市的任务定价和完成情况。具体定价方案见附件。

### 新定价方案结果分析

通过将新定价方案与原定价方案中的任务总定价与任务完成情况进行对比，可以发现：

新定价方案下的总定价相较于原定价方案减少了2418.042元，降低了4.2%的定价费用，同时任务完成率提高了28.62%，说明该定价模型较好的优化了定价方案，在一定程度上提高了整体效益。

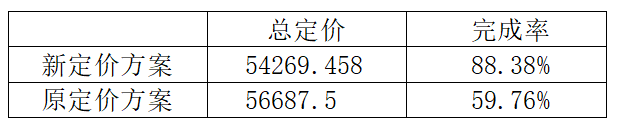


图 8新定价方案与原定价方案具体情况

## 问题三模型的建立与求解

### 问题三的分析

将部分分布密集的任务打包发布不仅能避免会员的争相选择，而且能将一些因定价较低的任务完成率提高。因此，我们可以基于问题二中的以完成任务量最大且总定价最低的目标模型，建立打包定价模型。

在该问题中，我们需要规定打包的数量限额，打包后任务总定价限额和任务密集程度的界定。因此，我们认为：

1. 某一范围内任务打包的数量与该范围内会员的数量有关；
2. 打包的总定价始终限额在某一范围中任务的单个最低金额与总定价之间；
3. 打包定价模型的任务分配阶段与任务完成阶段与无打包定价模型相同。

于是，我们作出以下打包原则：

1. 市中心的任务因该区域内会员较多应尽量不打包或打包数量尽量少，郊区的任务尽量要打包，这样可以最大限度的调动市中心的会员，以及提高郊区任务的完成度；
2. 任一打包的数量不应超过邻域内会员的最高限额；
3. 能进行打包的各任务间距离需要足够近；
4. 任一任务在邻域半径为10km的范围圆内进行打包。

### 问题三的打包定价模型的建立

对于最终的任务完成情况，采用与问题二相同的评价方式，即从总定价与总任务完成情况进行评价，因此，我们可以得出与问题二相同的目标函数：

|  |  |
| --- | --- |
|  | () |

打包定价模型是在无打包定价模型的基础上建立的，因此，打包定价模型的约束条件应有：

|  |  |
| --- | --- |
|  | (13) |

同时，基于打包原则，可以得出另一约束条件：

|  |  |
| --- | --- |
|  | (14) |

其中，表示打包的任务总定价，该总定价应高于该范围圆内任一任务的原定价，且低于范围圆内任务的原定价总和。

综上，我们建立的以完成任务量最大且总定价最低的打包定价模型为：

|  |  |
| --- | --- |
|  | (15) |
|  | (16) |

### 打包定价模型的求解

Step 1：打包组合

鉴于实际情况中跨市的问题，将各打包组合限制于各所属城市中，且将各组合中的数量限制于[2,13]中。打包组合过程如下（流程图见附录4）：

1. 将各城市的任务位置合并为各数据集。
2. 随机选择并访问任一任务点Q，并产生半径约为1km的邻域，将产生该邻域内的任务集N。
3. 判断该任务点Q能否作为核心点，通过聚类算法，若该任务不能视为核心点，则标记为噪音点；若能视为核心点，则将该任务分类至新簇C中。
4. 在核心点的任务集N中随机选择某一任务点P，若该任务点P未被访问，则视为无类别点，即可加入新簇C中；若该任务点P已被访问，即会出现两种情况，一种情况为该任务点P已被视为核心点，则将该任务点下产生的新簇与任务点Q下产生的新簇进行合并，第二种情况为该任务点P已被视为噪音点，则将其加入C中。
5. 直至任务点Q下的任务集N中所有任务点均完成范围，则返回数据集重复上诉步骤2—4。
6. 直至各城市的数据集均完成分类，即得到所需的打包组合，如图9，其中，相同颜色则表示同一打包组合，无颜色则表示噪声点，即不需要或不能进行打包的任务。

因此，利用matlab软件，我们得到了176个打包组合。



图 9 打包组合的具体位置

Step 2：打包组合的中心位置

为了使打包后的任务组合与未进行打包的单独任务具有相同的特征，我们规定出任一打包组合的中心位置，则将打包组合的定价问题可以转换为中心位置的定价问题。即：

|  |  |
| --- | --- |
|  | (17) |

其中，n表示为打包数量，该位置的定价为打包组合总定价的均定价，如图10，部分打包组合中心位置的经纬度和原均定价。

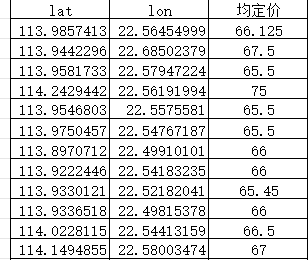
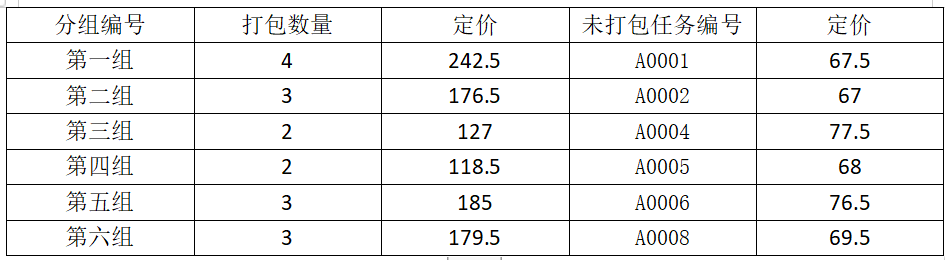


图 10 部分打包组合的中心位置情况

Step 3：从各城市中最低定价的任务开始与第一批次开始任务的会员进行匹配，即将该任务分配给对应范围圆内6：30开始任务，且最高的会员。将原方案中未完成的任务和已完成的任务分别增加和减少一定的价钱，同时根据式（4）所得的w与进行对比，若高于阈值，则该任务视为完成，则不再参与后续匹配，同时根据阈值、距离、会员完成能力，即可得到该任务的定价；若低于阈值，则该任务未被完成，同时不再参与后续时间批次的会员匹配。

Step 4：将城市中的任务从低到高依次与不同批次进行步骤3，最终即可得出城市的任务定价和完成情况。得到的部分新项目任务定价及完成情况如下表（具体定价方案见附件）：



### 打包定价方案结果分析

根据最终的总定价和任务完成率的结果，相较于未打包的定价模型下的总定价和任务完成情况，可以发现：

|  |  |  |
| --- | --- | --- |
|  | 总定价 | 任务完成率 |
| 打包定价模型 | 52566.569 | 90.12% |
| 未打包定价模型 | 54269.458 | 88.38% |

## 问题四模型的建立与求解

### 问题四模型的建立

根据问题三中的打包定价模型与问题二中未打包的定价模型的任务总定价和完成情况对比，可以发现问题三的打包定价模型具有更高的效益，因此，新项目的定价方案依据打包定价模型而定。即：

|  |  |
| --- | --- |
|  | (15) |
|  | (16) |

### 问题四模型的求解

Step 1：打包组合

根据新项目中的任务经纬度，将其导入地图中。可以发现，任务依然分布于广州，深圳，东莞和佛山四个城市，且任务密集程度远大于前述项目。

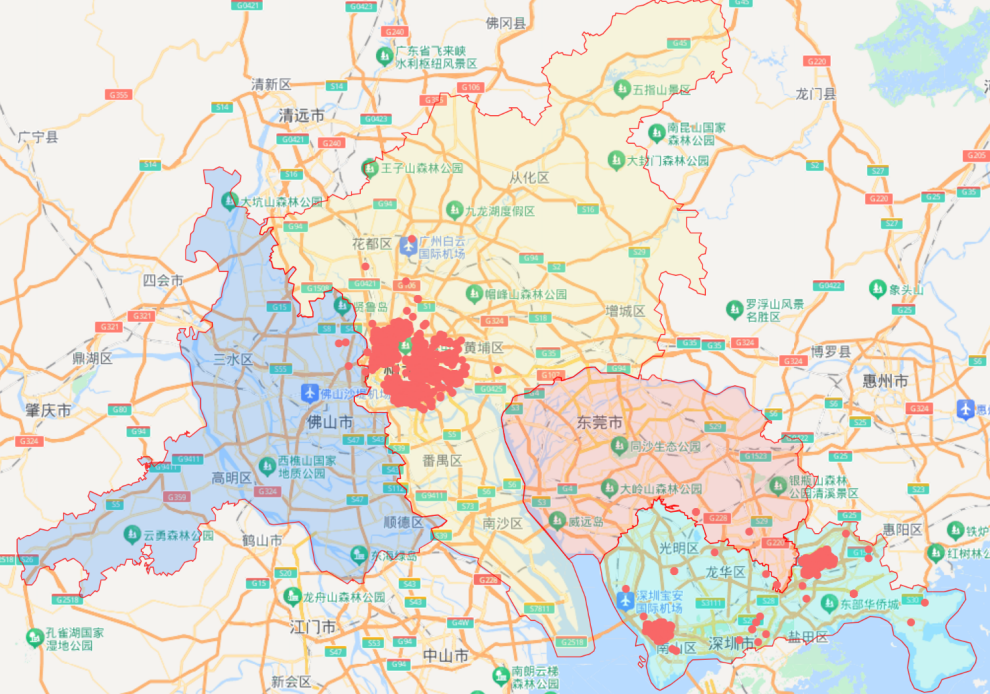


图 11新项目任务分布

执行问题三中打包组合步骤，为使得打包的合理性，将其中的邻域半径缩小为约100m。根据DESCAN聚类算法，利用matlab软件，最终得到385个打包组合。

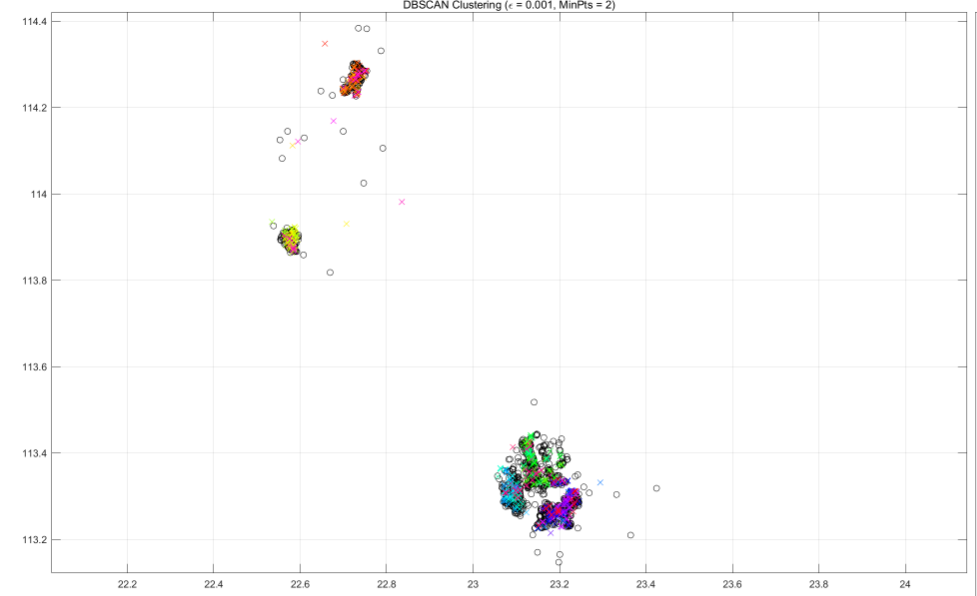


图 12新项目打包情况

其中，相同颜色的任务为同一打包组合，无颜色点表示不需或不能进行打包的任务。

Step 2：采用类似于问题二中任务分配和任务完成的判断，可以得到任务完成率为91.56%的结果。得到部分新项目任务定价和完成情况如下（具体结果见附件）：



# 模型的评价、改进与推广

## 模型的优点

1. 在任务分配阶段利用两种方法求解定价，距离和会员完成能力三者的权重，排除了权重的偶然性结果；
2. 分别设置了任务分配优化模型和双目标规划模型，全面的考虑了任务密集程度，不同城市实际经济水平等因素；
3. 建立的打包定价模型，较好的解决了问题二中部分任务未完成的问题；
4. 在打包问题中，采用DESCAN聚类算法分析，保证了打包方案的合理性和可行性。

## 模型的缺点

1. 忽略了实际生活中任务难易问题和会员的实际情况；
2. 采用搜索式算法，在全面搜索的同时降低了一定的效率。

# 参考文献

[1]姜启源,谢金星,叶俊.数学模型.第4版[M].高等教育出版社,2011.

[2]周水庚,周傲英,曹晶.基于数据分区的DBSCAN算法[J].计算机研究与发展, 2000, 37(10):7.DOI:CNKI:SUN:JFYZ.0.2000-10-000.

附录

|  |
| --- |
| 附录1 |
| 被剔除点位的代码 |
| a=xlsread('新建 XLSX 工作表 (3).xlsx',1,'B2:B1878'); %读取excel文件中各点纬度的数据，经度同理  %%  aa=mean(a); %算出x的平均值  sig=std(a); %算出x的标准偏差  n = zeros(size(a));  i=1;  for t=1:length(a)  m(t)=abs(a(t)-aa);  if m(t)>3\*sig  n(t)=a(t); %记录异常数据  i=i+1;  end  end |

|  |
| --- |
| 附录2 |
| 算术平均法和特征值法求权重 |
| %% 获取判断矩阵  disp('请输入判断矩阵A')  A=input('A=');  [n,n] = size(A);  %% 方法1： 算术平均法求权重  Sum\_A = sum(A); %将A矩阵按列求和  SUM\_A = repmat(Sum\_A,n,1);  Stand\_A = A ./ SUM\_A;  disp('算术平均法求权重的结果为：');  w1 = sum(Stand\_A,2)./n;  disp(w1)  %% 方法2： 特征值法求权重  [V,D] = eig(A);  Max\_eig = max(max(D));% 先按列求最大值，得到行向量，再从这个向量里面求最大值  [r,c]=find(D == Max\_eig , 1);  disp('特征值法求权重的结果为：');  w2 = V(:,c) ./ sum(V(:,c));  disp(w2)  disp('两种方法的平均权值为：');  disp((w1 + w2) ./ 2);  %% 计算一致性比例CR  CI = (Max\_eig - n) / (n-1);  RI=[0 0.0001 0.52 0.89 1.12 1.26 1.36 1.41 1.46 1.49 1.52 1.54 1.56 1.58 1.59];  % 这里n=2时，RI=0，我们为了避免分母为0，将这里的第二个元素改为了很接近0的正数  CR=CI/RI(n);  disp('最大特征值为:');  disp(Max\_eig);  disp('一致性指标CI=');disp(CI);  disp('一致性比例CR=');disp(CR);  if CR<0.10  disp('CR<0.10，该判断矩阵A的一致性可以接受!');  else  disp('注意：CR >= 0.10，该判断矩阵需要进行修改!');  end |

|  |
| --- |
| 附录3 |
| 第二问定价模型代码求解 |
| opts = spreadsheetImportOptions("NumVariables", 6);  % 指定工作表和范围  opts.Sheet = "Sheet1";  opts.DataRange = "A1:F1878";  % 指定列名称和类型  opts.VariableNames = ["Var1", "VarName2", "VarName3", "VarName4", "VarName5", "VarName6"];  opts.SelectedVariableNames = ["VarName2", "VarName3", "VarName4", "VarName5", "VarName6"];  opts.VariableTypes = ["char", "double", "double", "double", "double", "double"];  % 指定变量属性  opts = setvaropts(opts, "Var1", "WhitespaceRule", "preserve");  opts = setvaropts(opts, "Var1", "EmptyFieldRule", "auto");  % 导入数据  Totaldata\_4 = readtable("I:\大二\数模\暑假训练\附件2处理版1.xlsx", opts, "UseExcel", false)  清除临时变量  clear opts  opts = spreadsheetImportOptions("NumVariables", 4);  % 指定工作表和范围  opts.Sheet = "Sheet1";  opts.DataRange = "A2:D1866";  % 指定列名称和类型  opts.VariableNames = ["VarName1", "VarName2", "VarName3", "VarName4"];  opts.VariableTypes = ["string", "double", "double", "double"];  % 指定变量属性  opts = setvaropts(opts, "VarName1", "WhitespaceRule", "preserve");  opts = setvaropts(opts, "VarName1", "EmptyFieldRule", "auto");  % 导入数据  Totaldata = readtable("I:\大二\数模\暑假训练\附件2处理版.xlsx", opts, "UseExcel", false)  opts = spreadsheetImportOptions("NumVariables", 5);  % 指定工作表和范围  opts.Sheet = "Sheet1";  opts.DataRange = "A2:E1878";  % 指定列名称和类型  opts.VariableNames = ["VarName1", "GPS", "VarName3", "VarName4", "VarName5"];  opts.VariableTypes = ["string", "string", "double", "double", "double"];  % 指定变量属性  opts = setvaropts(opts, ["VarName1", "GPS"], "WhitespaceRule", "preserve");  opts = setvaropts(opts, ["VarName1", "GPS"], "EmptyFieldRule", "auto");  % 导入数据  Totaldata\_2 = readtable("I:\大二\数模\暑假训练\附件2.xlsx", opts, "UseExcel", false)%信誉值  opts = spreadsheetImportOptions("NumVariables", 5);  % 指定工作表和范围  opts.Sheet = "t\_tasklaunch";  opts.DataRange = "A2:E836";  % 指定列名称和类型  opts.VariableNames = ["VarName1", "gps", "gps1", "VarName4", "VarName5"];  opts.VariableTypes = ["string", "double", "double", "double", "double"];  % 指定变量属性  opts = setvaropts(opts, "VarName1", "WhitespaceRule", "preserve");  opts = setvaropts(opts, "VarName1", "EmptyFieldRule", "auto");  % 导入数据  Totaldata\_1 = readtable("I:\大二\数模\暑假训练\附件1.xls", opts, "UseExcel", false)%renwu  clear opts  opts = spreadsheetImportOptions("NumVariables", 6);  % 指定工作表和范围  opts.Sheet = "标注图层1";  opts.DataRange = "A1:F836";  % 指定列名称和类型  opts.VariableNames = ["Var1", "gps", "gps1", "VarName4", "VarName5", "VarName6"];  opts.SelectedVariableNames = ["gps", "gps1", "VarName4", "VarName5", "VarName6"];  opts.VariableTypes = ["char", "double", "double", "double", "double", "double"];  % 指定变量属性  opts = setvaropts(opts, "Var1", "WhitespaceRule", "preserve");  opts = setvaropts(opts, "Var1", "EmptyFieldRule", "auto");  % 导入数据  Totaldata\_renwu = readtable("I:\大二\数模\暑假训练\任务归属地.xls", opts, "UseExcel", false)  clear opts  %算wij  misson\_location\_1 = table2array(Totaldata\_renwu(2:end,:));  misson\_person\_1 = table2array(Totaldata\_4(2:end,:));  sequence\_3 = size(misson\_location\_1,1);  sequence\_4 = size(misson\_person\_1,1);  number\_1 = 1:sequence\_3;  number\_2 = 1:sequence\_4;  misson\_location = [number\_1',misson\_location\_1];  misson\_person = [number\_2',misson\_person\_1];%final  citycells = cell(4,1);  wij =[];  for i= 1:4  citycells{i} = misson\_location(misson\_location(:,6) == i,:);  end  dingjia = cell(4,1);  %将每个市的任务进行分类  for i\_1 = 1:4  Location\_1 = citycells{i\_1,1};%每个市的任务数据    latLon1 = Location\_1(:, 2:3); % 数组1的纬度和经度  latLon2 = misson\_person(:, 2:3); % 数组2的纬度和经度    % 计算每个点之间的距离（单位：公里）  distances = pdist2(latLon1, latLon2, 'euclidean') \* 111.32; % 1度大约等于111.32公里    % 设置5公里的阈值  threshold = 5;    % 找出小于5公里的点  [row\_idx, col\_idx] = find(distances <= threshold);    % 输出结果数组  % 第一列是数组1的序号，第二列是数组2中小于5公里的序号  result\_1 = [Location\_1(row\_idx, 1), misson\_person(col\_idx, 1)];  result\_misson = sortrows(result\_1,1);%人员任务对应值    %求对应值  potential\_city = misson\_person(result\_misson(:,2),5);%xinyuzhi  result\_misson = cat(2,result\_misson,potential\_city)  %    %算信誉值  credit\_3 = [];  for m\_1 = 1:size(result\_misson, 1)  current\_value\_1 = result\_misson(m\_1, 1); % 当前行的第一列的值    % 检查当前行是否是重复值的一部分  is\_repeated = false;    if m\_1 > 1 && result\_misson(m\_1-1, 1) == current\_value\_1  is\_repeated = true;  end    if m\_1 < size(result\_misson, 1) && result\_misson(m\_1+1, 1) == current\_value\_1  is\_repeated = true;  end    if is\_repeated  % 如果是重复值，找出相同值对应的第二列的最大xinyuzhi  same\_value\_indices\_1 = find(result\_misson(:, 1) == current\_value\_1);  max\_second\_col\_value\_1 = max(result\_misson(same\_value\_indices\_1, 2));  max\_credit\_1 = result\_misson(m\_1,2) ./ max\_second\_col\_value\_1;  credit\_3 = [credit\_3; max\_credit\_1];  else  % 如果不是重复值，直接输出第二列的值  only\_credit\_1 = result\_misson(m\_1, 2) ./ (result\_misson(m\_1,2) + 1);  credit\_3 = [credit\_3; only\_credit\_1];  end  end  % 输出结果，信誉值  disp(credit\_3);    %% 算距离,初始化举证为序号对应的坐标  numRows\_1 = size(result\_misson, 1);    % 初始化结果数组  result\_3 = zeros(numRows\_1, 4); % 假设最终结果包含四列    % 遍历第一个数组并提取数据  for z = 1: size(result\_misson,1)  idx = Location\_1(:,1) == result\_misson(z,1);  result\_3(z,1:2) = Location\_1(idx,2:3);  idx\_1 = misson\_person(:,1) == result\_misson(z,2);  result\_3(z,3:4) = misson\_person(idx\_1,2:3);    end    clear z  clear idx    %算距离  distance\_2 = [];  for i\_9 =1:size(result\_3,1)  lat\_5 = result\_3(i\_9,1);  lon\_5 = result\_3(i\_9,2);  lat\_6 = result\_3(i\_9,3);  lon\_6 = result\_3(i\_9,4);  lat\_5 = deg2rad(lat\_5);  lon\_5 = deg2rad(lon\_5);  lat\_6 = deg2rad(lat\_6);  lon\_6 = deg2rad(lon\_6);  % Haversine 公式  dlat = lat\_6 - lat\_5;  dlon = lon\_6 - lon\_5;    a = sin(dlat/2)^2 + cos(lat\_5) \* cos(lat\_6) \* sin(dlon/2)^2;  c = 2 \* atan2(sqrt(a), sqrt(1-a));    % 地球半径 (公里)  R = 6371;    % 计算距离  d\_3 = R \* c;  d\_4 = round(d\_3,2);  distance\_2(i\_9) = d\_4;  distance\_3 = distance\_2';    end      %查找定价  B = result\_misson(:,1);  % 初始化结果数组  result\_4 = zeros(length(B), size(Location\_1,4));    % 查找数组2中的序号在数组1中的对应数据  for i\_10 = 1:length(B)  % 查找数组1中序号等于数组2中当前序号的行  idx = Location\_1(:, 1) == B(i\_10);    % 存储对应的数据  result\_4(i\_10, :) = Location\_1(idx, 4);  end  clear idx    %算每个对应的任务和人员的wij  potential\_wij = zeros(size(result\_misson,1),1);  for i\_7 = 1:size(result\_misson,1)  potential\_wij\_1 = 0.0745 \* result\_4(i\_7,1) + 0.1231 ./ distance\_3(i\_7) + 0.8024 \* credit\_3(i\_7);  potential\_wij(i\_7,1) = potential\_wij\_1;  end    result\_5 = zeros(size(result\_misson,1),1);    for z = 1: size(result\_misson,1)  idx = Location\_1(:,1) == result\_misson(z,1);  result\_5(z,:) = Location\_1(idx,5);  end    result\_misson\_1 = [result\_misson,result\_5,potential\_wij];  disp (result\_misson\_1);    %按照0和1计算wij的最小和最大值。  max\_wij = [];  min\_wij = [];  for z\_4 = 1:size(result\_misson\_1,1)  if result\_misson\_1(z\_4,4) == 0 && result\_misson\_1(z\_4,5)<10  max\_wij(z\_4,1) = result\_misson\_1(z\_4,5);  end  if result\_misson\_1(z\_4,4) == 1  min\_wij(z\_4,1) = potential\_wij(z\_4,1);  end  end  if ~isempty(max\_wij)  city\_max\_wij = max(max\_wij);    end    if ~isempty(min\_wij)  min\_wij = min\_wij(min\_wij ~= 0 );  city\_min\_wij = min(min\_wij);    end    disp("wij的最大值是")  disp(city\_max\_wij);  disp("wij的最小值是")  disp(city\_min\_wij);  wij(i\_1,1) = (city\_min\_wij + city\_max\_wij) ./ 2.15;    %，返回wij，算出定价表  misson\_prize = zeros(size(result\_misson\_1,1),1);  for z\_10 = 1:size(result\_misson\_1,1)  misson\_prize(z\_10,1) = (wij(i\_1,1) - 0.1231 ./ distance\_3(z\_10) - 0.8024 \* credit\_3(z\_10)) ./ 0.0745;  end  misson\_prize\_1 = [result\_misson\_1(:,1),misson\_prize];    final\_misson = [];  unique\_misson = unique(misson\_prize\_1(:,1)) ;  for z\_9 = 1:size(unique\_misson,1)  current\_value\_2 = unique\_misson(z\_9,1);  same\_unique\_misson = find(misson\_prize\_1(:,1) == current\_value\_2);  if numel(same\_unique\_misson,1) > 1  Max\_prize = max(misson\_prize\_1(same\_unique\_misson,2));  final\_misson = [final\_misson;Max\_prize];    else  oringnal\_prize = misson\_prize\_1(same\_unique\_misson,2);  final\_misson = [final\_misson;oringnal\_prize];  end        end      disp("定价表");  disp(final\_misson);  final\_misson\_1 = [result\_misson\_1(:,1),final\_misson];        %找到最小定价  [uniqueTasks, ~, idx] = unique(final\_misson\_1(:,1)); % 找到唯一的任务  minPrices = accumarray(idx, final\_misson\_1(:,2), [], @min); % 找到每个任务的最小定价  % 将结果组合在一起  result\_7 = sortrows([uniqueTasks, minPrices],1);  dingjia{i\_1} = result\_7;      end  disp("wij的值")  disp(wij)  writecell(dingjia, '定价.xlsx')  Finishtime = table2array(Totaldata(:,4))  [Finishtime\_1,ind\_1] = sortrows(Finishtime)%ind\_1是该时间在原来表格的序号,排序表  Finishitime\_2 = cat(2,ind\_1,Finishtime\_1)%Finishitime\_2是组合后的表格  %按照时间分类，一个map  unique\_value = unique(Finishitime\_2(:,2))  map = cell(1,length(unique\_value))  for l = 1:length(unique\_value)  value = unique\_value(l);  [row, col] = find(Finishitime\_2(:,2) == value); % 找到该值在矩阵中的位置  %fprintf('Value %d is found at:\n', value);%value是时间段  %disp((row)); % 显示该值的行索引  map{l} = [Finishitime\_2(row)];  end  misson = table2array(Totaldata(:,2:3))%人员  rowNumbers = (1:size(misson, 1))';  misson\_withIndex = [rowNumbers, misson];  misson\_1 = table2array(Totaldata\_1(:,2:3)) %任务  result\_map = cell(1,31);  %循环，遍历每个时间  for k = 1:length(map)  k\_1 = map{1,k}%k\_1是按时间顺序分类的序号  misson\_withIndex\_1 = misson\_withIndex(:,2:3);  misson\_2 = misson\_withIndex\_1(k\_1,:)%人员  % for m = 1:length(misson\_2)    % 1. 将经纬度转换为弧度制  lat2\_rad = deg2rad(misson\_2(:, 1));%人员  lon2\_rad = deg2rad(misson\_2(:, 2));  lat1\_rad = deg2rad(misson\_1(:, 1));%任务  lon1\_rad = deg2rad(misson\_1(:, 2));  % F1 = cat(2,lat1\_rad,lon1\_rad)  % F2 = cat(2,lat2\_rad,lon2\_rad)  % 地球半径（单位：公里）  R = 6371;  n1 = size(misson\_1,1)    n2 = size(misson\_2,1)    % 计算每对点之间的距离  distances = zeros(n1, n2); % 初始化距离矩阵，有misson1（任务）行，misson2列，那么相当与1对应2  for l = 1:n1  for j = 1:n2  deltaLat = lat2\_rad(j) - lat1\_rad(l);  deltaLon = lon2\_rad(j) - lon1\_rad(l);    a = sin(deltaLat/2)^2 + cos(lat1\_rad(l)) \* cos(lat2\_rad(j)) \* sin(deltaLon/2)^2;  c = 2 \* atan2(sqrt(a), sqrt(1-a));    distances(l,j) = R \* c; % 计算距离并存储    end  end      [row, col] = find(distances <= 5); % 找到距离小于等于5公里的点对  Coord = cat(2,row,col);  fprintf('第 %d 次', k)  disp(Coord)  % 获取坐标矩阵的行数  numCods = size(Coord, 1);  % 初始化数组以存储结果  result = zeros(numCods, 1);  % 遍历坐标矩阵并提取数据  for b = 1:numCods  row = Coord(b, 1);  col = Coord(b, 2);  result(b) = distances(row, col);%result代表的是distance中的对应位置的数据  end  disp('距离位置显示:');  disp(result);    unique\_location = unique(Coord(:,2))  misson\_3 = misson\_2(unique\_location,:)%misson\_3 is调用的值在misison2中的位置,是唯一值,人员  Coord\_1 = sortrows(Coord,1)  unique\_location\_1 = unique(Coord\_1(:,1));  misson\_4 = misson\_1(unique\_location\_1,:)%排序后的misson1，是唯一值,任务    % for n\_1 = 1:length(unique\_location)  % if(n\_1 == unique\_location(n\_1))  % [row\_2] = find(Coord(:,2) == unique\_location(n\_1),1);  % disp(row\_2)  % end  %  % end  %创建序号  numRows\_1 = size(misson\_3, 1);  numRows\_2 = size(misson\_4, 1);  sequence\_1 = (1:numRows\_1)';  sequence\_2 = (1:numRows\_2)';  % 将序号列添加到矩阵 A 的前面  Misson\_3 = [sequence\_1, misson\_3];%注意大写  Misson\_4 = [sequence\_2, misson\_4];  n\_3 = size(Misson\_3(:,1),2)  n\_4 = size(Misson\_4(:,1),2)      % 获取第一个数组的行数  numRows = size(Coord, 1);    % 初始化结果数组  result\_2 = zeros(numRows, 4); % 假设最终结果包含四列    % 遍历第一个数组并提取数据，创建任务人员对应表  for i = 1:numRows  % 从第一个数组中提取索引  index2 = Coord(i, 1);  index3 = Coord(i, 2);    % 使用索引提取第二和第三个数组中的数据  coord2 = misson\_1(index2, :);%任务  coord3 = misson\_2(index3, :);%coord3人员    % 将结果存储在结果数组中  result\_2(i, :) = [coord2, coord3];  end    % 显示结果  disp('任务匹配表')  disp(result\_2);  task\_1 = table2array(Totaldata\_1(:,2:5));%任务  task\_2 = table2array(Totaldata(:,2:4));%人员  [commonValues, idxArray1] = ismember(result\_2(:,1:2),task\_1(:,1:2), 'rows');  disp('在原任务数组中的位置:');  disp(idxArray1)  [commonValues\_1, idxArray2] = ismember(result\_2(:,3:4 ),task\_2(:,1:2), 'rows');  disp('在原人员数组中的位置:');  disp(idxArray2)  task\_final = sortrows(cat(2,idxArray1,idxArray2));  disp('任务人员排序表')  disp(task\_final);  j\_1 = size(task\_final,1);  cregitfinal = zeros(j\_1,1);  %算每个任务对应人的距离  distances\_1 = [];  for j\_2 = 1:j\_1  j\_3 = task\_final(j\_2,1);  lat\_3 = task\_1(j\_3,1);  lon\_3 = task\_1(j\_3,2);  j\_4 = task\_final(j\_2,2);%遍历第二列  lat\_4 = task\_2(j\_4,1);  lon\_4 = task\_2(j\_4,2);  lat\_3 = deg2rad(lat\_3);  lon\_3 = deg2rad(lon\_3);  lat\_4 = deg2rad(lat\_4);  lon\_4 = deg2rad(lon\_4);  % Haversine 公式  dlat = lat\_4 - lat\_3;  dlon = lon\_4 - lon\_3;    a = sin(dlat/2)^2 + cos(lat\_3) \* cos(lat\_4) \* sin(dlon/2)^2;  c = 2 \* atan2(sqrt(a), sqrt(1-a));    % 地球半径 (公里)  R = 6371;    % 计算距离  d\_1 = R \* c;  d\_2 = round(d\_1,2);  distances\_1(j\_2) = d\_2;    %xinyu  credit\_2 = table2array(Totaldata\_2(:,5));  j\_5 = task\_final(j\_2,2);%renyuan  credit\_1 = credit\_2(j\_5,1) ;    cregitfinal(j\_2) = credit\_1;    end  distances\_2 = distances\_1'%按照人员顺序的距离顺序    l\_1 = task\_final(:,1);  credit\_final = cat(2,l\_1,cregitfinal);%任务对应的信誉    %算信誉值  credit = [];% 初始化一个信用数组  % 遍历矩阵的每一行，算信用值  for m\_1 = 1:size(credit\_final, 1)  current\_value = credit\_final(m\_1, 1); % 当前行的第一列的值    % 检查当前行是否是重复值的一部分  is\_repeated = false;    if m\_1 > 1 && credit\_final(m\_1-1, 1) == current\_value  is\_repeated = true;  end    if m\_1 < size(credit\_final, 1) && credit\_final(m\_1+1, 1) == current\_value  is\_repeated = true;  end    if is\_repeated  % 如果是重复值，找出相同值对应的第二列的最大值  same\_value\_indices = find(credit\_final(:, 1) == current\_value);  max\_second\_col\_value = max(credit\_final(same\_value\_indices, 2));  max\_credit = credit\_final(m\_1,2) ./ max\_second\_col\_value;  credit = [credit; max\_credit];  else  % 如果不是重复值，直接输出第二列的值  only\_credit = credit\_final(m\_1, 2) ./ (credit\_final(m\_1,2) + 1);  credit = [credit; only\_credit];  end  end  % 输出结果  disp(double(credit));    money\_1 = table2array(Totaldata\_1(:,4));  money\_index = money\_1(task\_final(:,1),:);  Attraction\_1 = [];  potential\_1 = [];  for j\_6 = 1:j\_1  Attraction\_2 = 0.3333 \* distances\_2(j\_6) + 0.6667 \* credit(j\_6);  Attraction\_1(j\_6) = Attraction\_2;  potential\_2 = 0.0745 \* money\_index(j\_6,:) + 0.1231 \* distances\_2(j\_6) + 0.8024 \* credit(j\_6);  potential\_1(j\_6) = potential\_2;  end  Attraction = Attraction\_1';  potential = potential\_1';  disp('吸引力数为')  disp(Attraction)  disp('可能性数为')  disp(potential)      Final = [];% 初始化一个信用数组  max\_index = [];  Final\_1 = [task\_final,Attraction,potential];    final\_1 = zeros(size(Final\_1,1),1);      %获取城市坐标  for z\_9 = 1: size(Final\_1,1)  idx\_7 = misson\_location(:,1) == Final\_1(z\_9,1);  final\_1(z\_9,:) = misson\_location(idx\_7,6);  end  final\_2 = [Final\_1,final\_1];          % 找出所有的任务序号  unique\_tasks = unique(final\_2(:, 1));  results = zeros(length(unique\_tasks), 2); % 第一列为任务序号，第二列为任务是否完成（0 或 1）    for i = 1:length(unique\_tasks)    task\_id = unique\_tasks(i);  task\_rows = final\_2(final\_2(:, 1) == task\_id, :);    % 获取当前任务对应的第五列的值  task\_category = task\_rows(1, 5); % 假设每个任务的第5列值都是相同的  wij\_1 = wij(task\_category,1);    % 找到分配指标最大的人员  [~, max\_idx] = max(task\_rows(:, 4));  max\_person\_row = task\_rows(max\_idx, :);    % 判断是否完成任务  if max\_person\_row(4) > wij\_1  is\_completed = 1;  else  is\_completed = 0;  end  results(i, :) = [task\_id, is\_completed];  end    % 输出结果,  disp(results);    result\_map{k} = results;  end  final\_table = []; % 初始化最终完成表  all\_tasks = []; % 用于存储所有可能的任务编号  % 先收集所有可能的任务编号  for i = 1:31  data\_1 = result\_map{i};  all\_tasks = unique([all\_tasks; data\_1(:, 1)]);  end  % 初始化最终完成表，每个任务标记为未完成（2）  final\_table = [all\_tasks, 0\*ones(length(all\_tasks), 1)];  % 遍历所有30组数据  for i = 1:31  data\_1 = result\_map{i};  task\_numbers = data\_1(:, 1);  completion\_status = data\_1(:, 2);    for j = 1:length(task\_numbers)  task\_number = task\_numbers(j);  status = completion\_status(j);    % 查找final\_table中该任务的位置  idx = find(final\_table(:, 1) == task\_number);    if status == 1  % 如果任务完成，标记为已完成（1）  final\_table(idx, 2) = 1;  end  end  end  % 检查是否有任务在30组数据中都未完成  for i = 1:length(all\_tasks)  task\_number = all\_tasks(i);    % 查找所有组中该任务的完成状态  completed = false;  for j = 1:30  data\_1 = result\_map{j};  idx = find(data\_1(:, 1) == task\_number);  if ~isempty(idx) && data\_1(idx, 2) == 1  completed = true;  break;  end  end    % 如果所有组中都未完成，将final\_table中的对应任务标记为0  if ~completed  final\_table(i, 2) = 0;  end  end  % 最后，排序final\_table使任务序号从小到大排列  final\_table = sortrows(final\_table, 1);  final\_table\_T = array2table(final\_table, 'VariableNames', {'TaskNumber', 'CompletionStatus'});  % 输出为 Excel 文件  writetable(final\_table\_T, 'final\_completion\_table.xlsx'); |

|  |
| --- |
| 附录4 |
| 打包组合流程图 |
| d898cd22f8f64eb62141926901da5ac |

|  |
| --- |
| 附录5 |
| 打包组合的DESCAN算法实现 |
| %% Load Data  load mydata;  %% Run DBSCAN Clustering Algorithm  epsilon=0.01;  MinPts=2;  IDX=DBSCAN(X,epsilon,MinPts);  %% Plot Results  PlotClusterinResult(X, IDX);  title(['DBSCAN Clustering (\epsilon = ' num2str(epsilon) ', MinPts = ' num2str(MinPts) ')']);  function [IDX, isnoise]=DBSCAN(X,epsilon,MinPts)  C=0;  n=size(X,1);  IDX=zeros(n,1);  D=pdist2(X,X);  visited=false(n,1);  isnoise=false(n,1);  for i=1:n  if ~visited(i)  visited(i)=true;  Neighbors=RegionQuery(i);  if numel(Neighbors)<MinPts  % X(i,:) is NOISE  isnoise(i)=true;  else  C=C+1;  ExpandCluster(i,Neighbors,C);  end  end  End  function ExpandCluster(i,Neighbors,C)  IDX(i)=C;  k = 1;  while true  j = Neighbors(k);  if ~visited(j)  visited(j)=true;  Neighbors2=RegionQuery(j);  if numel(Neighbors2)>=MinPts  Neighbors=[Neighbors Neighbors2]; %#ok  end  end  if IDX(j)==0  IDX(j)=C;  end  k = k + 1;  if k > numel(Neighbors)  break;  end  end  end  function Neighbors=RegionQuery(i)  Neighbors=find(D(i,:)<=epsilon);  end  end  function PlotClusterinResult(X, IDX)  k=max(IDX);  Colors=hsv(k);  Legends = {};  for i=0:k  Xi=X(IDX==i,:);  if i~=0  Style = 'x';  MarkerSize = 8;  Color = Colors(i,:);  Legends{end+1} = ['Cluster #' num2str(i)];  else  Style = 'o';  MarkerSize = 6;  Color = [0 0 0];  if ~isempty(Xi)  Legends{end+1} = 'Noise';  end  end  if ~isempty(Xi)  plot(Xi(:,1),Xi(:,2),Style,'MarkerSize',MarkerSize,'Color',Color);  end  hold on;  end  hold off;  axis equal;  grid on;  legend(Legends);  legend('Location', 'NorthEastOutside');  end |
| 附录4 |
| 第三问定价模型代码求解  clear  opts = spreadsheetImportOptions("NumVariables", 6);  % 指定工作表和范围  opts.Sheet = "标记位置";  opts.DataRange = "A2:F836";  % 指定列名称和类型  opts.VariableNames = ["VarName1", "VarName2", "VarName3", "VarName4", "VarName5", "VarName6"];  opts.VariableTypes = ["string", "double", "double", "double", "double", "double"];  % 指定变量属性  opts = setvaropts(opts, "VarName1", "WhitespaceRule", "preserve");  opts = setvaropts(opts, "VarName1", "EmptyFieldRule", "auto");  % 导入数据  Totaldate\_selection = readtable("I:\大二\数模\暑假训练\第三问分类结果(1) 的副本.xlsx", opts, "UseExcel", false)  clear opts  opts = spreadsheetImportOptions("NumVariables", 6);  % 指定工作表和范围  opts.Sheet = "Sheet1";  opts.DataRange = "A1:F1878";  % 指定列名称和类型  opts.VariableNames = ["Var1", "VarName2", "VarName3", "VarName4", "VarName5", "VarName6"];  opts.SelectedVariableNames = ["VarName2", "VarName3", "VarName4", "VarName5", "VarName6"];  opts.VariableTypes = ["char", "double", "double", "double", "double", "double"];  % 指定变量属性  opts = setvaropts(opts, "Var1", "WhitespaceRule", "preserve");  opts = setvaropts(opts, "Var1", "EmptyFieldRule", "auto");  % 导入数据  Totaldata\_4 = readtable("I:\大二\数模\暑假训练\附件2处理版1.xlsx", opts, "UseExcel", false)  清除临时变量  clear opts  opts = spreadsheetImportOptions("NumVariables", 4);  % 指定工作表和范围  opts.Sheet = "Sheet1";  opts.DataRange = "A2:D1866";  % 指定列名称和类型  opts.VariableNames = ["VarName1", "VarName2", "VarName3", "VarName4"];  opts.VariableTypes = ["string", "double", "double", "double"];  % 指定变量属性  opts = setvaropts(opts, "VarName1", "WhitespaceRule", "preserve");  opts = setvaropts(opts, "VarName1", "EmptyFieldRule", "auto");  % 导入数据  Totaldata = readtable("I:\大二\数模\暑假训练\附件2处理版.xlsx", opts, "UseExcel", false)  opts = spreadsheetImportOptions("NumVariables", 5);  % 指定工作表和范围  opts.Sheet = "Sheet1";  opts.DataRange = "A2:E1878";  % 指定列名称和类型  opts.VariableNames = ["VarName1", "GPS", "VarName3", "VarName4", "VarName5"];  opts.VariableTypes = ["string", "string", "double", "double", "double"];  % 指定变量属性  opts = setvaropts(opts, ["VarName1", "GPS"], "WhitespaceRule", "preserve");  opts = setvaropts(opts, ["VarName1", "GPS"], "EmptyFieldRule", "auto");  % 导入数据  Totaldata\_2 = readtable("I:\大二\数模\暑假训练\附件2.xlsx", opts, "UseExcel", false)%信誉值  opts = spreadsheetImportOptions("NumVariables", 5);  % 指定工作表和范围  opts.Sheet = "t\_tasklaunch";  opts.DataRange = "A2:E836";  % 指定列名称和类型  opts.VariableNames = ["VarName1", "gps", "gps1", "VarName4", "VarName5"];  opts.VariableTypes = ["string", "double", "double", "double", "double"];  % 指定变量属性  opts = setvaropts(opts, "VarName1", "WhitespaceRule", "preserve");  opts = setvaropts(opts, "VarName1", "EmptyFieldRule", "auto");  % 导入数据  Totaldata\_1 = readtable("I:\大二\数模\暑假训练\附件1.xls", opts, "UseExcel", false)%renwu  clear opts  opts = spreadsheetImportOptions("NumVariables", 6);  % 指定工作表和范围  opts.Sheet = "标注图层1";  opts.DataRange = "A1:F836";  % 指定列名称和类型  opts.VariableNames = ["Var1", "gps", "gps1", "VarName4", "VarName5", "VarName6"];  opts.SelectedVariableNames = ["gps", "gps1", "VarName4", "VarName5", "VarName6"];  opts.VariableTypes = ["char", "double", "double", "double", "double", "double"];  % 指定变量属性  opts = setvaropts(opts, "Var1", "WhitespaceRule", "preserve");  opts = setvaropts(opts, "Var1", "EmptyFieldRule", "auto");  % 导入数据  Totaldata\_renwu = readtable("I:\大二\数模\暑假训练\任务归属地.xls", opts, "UseExcel", false)  clear opts  selection\_misson = table2array(Totaldate\_selection(:,2:end));  % 去除分组序号为0的行  selection\_misson = selection\_misson(selection\_misson(:, 4) ~= 0, :);  % 获取唯一的分组序号  groups = unique(selection\_misson(:, 4));  % 初始化结果cell数组  numGroups = length(groups);  result = cell(numGroups, 5);  % 结果数组，用于存储最终输出  result\_array = zeros(numGroups, 6);  for i = 1:numGroups  % 获取当前组的所有任务  group\_data = selection\_misson(selection\_misson(:, 4) == groups(i), :);    % 计算中心经纬度  avg\_latitude = mean(group\_data(:, 1));  avg\_longitude = mean(group\_data(:, 2));    % 计算组定价的总和  total\_price = sum(group\_data(:, 5));    % 获取该组内任务的序号  task\_indices = find(selection\_misson(:, 4) == groups(i));    % 判断组内完成情况  if any(group\_data(:, 3) == 1)  completion\_status = 1; % 如果有至少一个任务完成  else  completion\_status = 0; % 否则未完成  end    % 将结果保存到cell数组  result{i, 1} = groups(i); % 组序号  result{i, 2} = total\_price; % 总定价  result{i, 3} = [avg\_latitude, avg\_longitude]; % 中心经纬度  result{i, 4} = task\_indices; % 任务序号  result{i, 5} = completion\_status; % 完成情况    % 将结果保存到结果数组中  result\_array(i, 1) = groups(i); % 组序号  result\_array(i, 2) = total\_price; % 总定价  result\_array(i, 3) = avg\_latitude; % 中心纬度  result\_array(i, 4) = avg\_longitude; % 中心经度  result\_array(i, 5) = length(task\_indices); % 任务序号数量  result\_array(i, 6) = completion\_status; % 完成情况  end  % 输出结果数组  disp('组序号 总定价 中心纬度 中心经度 任务序号数量 完成情况');  disp(result\_array);  result\_array\_1 =[result\_array(:,4),result\_array(:,3),result\_array(:,2),result\_array(:,6)]  wij = 6.5216  writematrix(result\_array,'分组情况.xlsx')  Finishtime = table2array(Totaldata(:,4))  [Finishtime\_1,ind\_1] = sortrows(Finishtime)%ind\_1是该时间在原来表格的序号,排序表  Finishitime\_2 = cat(2,ind\_1,Finishtime\_1)%Finishitime\_2是组合后的表格  %按照时间分类，一个map  unique\_value = unique(Finishitime\_2(:,2))  map = cell(1,length(unique\_value))  for l = 1:length(unique\_value)  value = unique\_value(l);  [row, col] = find(Finishitime\_2(:,2) == value); % 找到该值在矩阵中的位置  %fprintf('Value %d is found at:\n', value);%value是时间段  %disp((row)); % 显示该值的行索引  map{l} = [Finishitime\_2(row)];  end  misson = table2array(Totaldata(:,2:3))%人员  rowNumbers = (1:size(misson, 1))';  misson\_withIndex = [rowNumbers, misson];  misson\_1 =result\_array\_1(:,1:2) %任务  result\_map = cell(1,31);  %循环，遍历每个时间  for k = 1:length(map)  k\_1 = map{1,k}%k\_1是按时间顺序分类的序号  misson\_withIndex\_1 = misson\_withIndex(:,2:3);  misson\_2 = misson\_withIndex\_1(k\_1,:)%人员  % for m = 1:length(misson\_2)    % 1. 将经纬度转换为弧度制  lat2\_rad = deg2rad(misson\_2(:, 1));%人员  lon2\_rad = deg2rad(misson\_2(:, 2));  lat1\_rad = deg2rad(misson\_1(:, 1));%任务  lon1\_rad = deg2rad(misson\_1(:, 2));  % F1 = cat(2,lat1\_rad,lon1\_rad)  % F2 = cat(2,lat2\_rad,lon2\_rad)  % 地球半径（单位：公里）  R = 6371;  n1 = size(misson\_1,1)    n2 = size(misson\_2,1)    % 计算每对点之间的距离  distances = zeros(n1, n2); % 初始化距离矩阵，有misson1（任务）行，misson2列，那么相当与1对应2  for l = 1:n1  for j = 1:n2  deltaLat = lat2\_rad(j) - lat1\_rad(l);  deltaLon = lon2\_rad(j) - lon1\_rad(l);    a = sin(deltaLat/2)^2 + cos(lat1\_rad(l)) \* cos(lat2\_rad(j)) \* sin(deltaLon/2)^2;  c = 2 \* atan2(sqrt(a), sqrt(1-a));    distances(l,j) = R \* c; % 计算距离并存储    end  end      [row, col] = find(distances <= 10); % 找到距离小于等于10公里的点对  Coord = cat(2,row,col);  fprintf('第 %d 次', k)  disp(Coord)  % 获取坐标矩阵的行数  numCods = size(Coord, 1);  % 初始化数组以存储结果  result = zeros(numCods, 1);  % 遍历坐标矩阵并提取数据  for b = 1:numCods  row = Coord(b, 1);  col = Coord(b, 2);  result(b) = distances(row, col);%result代表的是distance中的对应位置的数据  end  disp('距离位置显示:');  disp(result);    unique\_location = unique(Coord(:,2))  misson\_3 = misson\_2(unique\_location,:)%misson\_3 is调用的值在misison2中的位置,是唯一值,人员  Coord\_1 = sortrows(Coord,1)  unique\_location\_1 = unique(Coord\_1(:,1));  misson\_4 = misson\_1(unique\_location\_1,:)%排序后的misson1，是唯一值,任务    % for n\_1 = 1:length(unique\_location)  % if(n\_1 == unique\_location(n\_1))  % [row\_2] = find(Coord(:,2) == unique\_location(n\_1),1);  % disp(row\_2)  % end  %  % end  %创建序号  numRows\_1 = size(misson\_3, 1);  numRows\_2 = size(misson\_4, 1);  sequence\_1 = (1:numRows\_1)';  sequence\_2 = (1:numRows\_2)';  % 将序号列添加到矩阵 A 的前面  Misson\_3 = [sequence\_1, misson\_3];%注意大写  Misson\_4 = [sequence\_2, misson\_4];  n\_3 = size(Misson\_3(:,1),2)  n\_4 = size(Misson\_4(:,1),2)      % 获取第一个数组的行数  numRows = size(Coord, 1);    % 初始化结果数组  result\_2 = zeros(numRows, 4); % 假设最终结果包含四列    % 遍历第一个数组并提取数据，创建任务人员对应表  for i = 1:numRows  % 从第一个数组中提取索引  index2 = Coord(i, 1);  index3 = Coord(i, 2);    % 使用索引提取第二和第三个数组中的数据  coord2 = misson\_1(index2, :);%任务  coord3 = misson\_2(index3, :);%coord3人员    % 将结果存储在结果数组中  result\_2(i, :) = [coord2, coord3];  end    % 显示结果  disp('任务匹配表')  disp(result\_2);  task\_1 = result\_array\_1;%任务  task\_2 = table2array(Totaldata(:,2:4));%人员  [commonValues, idxArray1] = ismember(result\_2(:,1:2),task\_1(:,1:2), 'rows');  disp('在原任务数组中的位置:');  disp(idxArray1)  [commonValues\_1, idxArray2] = ismember(result\_2(:,3:4 ),task\_2(:,1:2), 'rows');  disp('在原人员数组中的位置:');  disp(idxArray2)  task\_final = sortrows(cat(2,idxArray1,idxArray2));  disp('任务人员排序表')  disp(task\_final);  j\_1 = size(task\_final,1);  cregitfinal = zeros(j\_1,1);  %算每个任务对应人的距离  distances\_1 = [];  for j\_2 = 1:j\_1  j\_3 = task\_final(j\_2,1);  lat\_3 = task\_1(j\_3,1);  lon\_3 = task\_1(j\_3,2);  j\_4 = task\_final(j\_2,2);%遍历第二列  lat\_4 = task\_2(j\_4,1);  lon\_4 = task\_2(j\_4,2);  lat\_3 = deg2rad(lat\_3);  lon\_3 = deg2rad(lon\_3);  lat\_4 = deg2rad(lat\_4);  lon\_4 = deg2rad(lon\_4);  % Haversine 公式  dlat = lat\_4 - lat\_3;  dlon = lon\_4 - lon\_3;    a = sin(dlat/2)^2 + cos(lat\_3) \* cos(lat\_4) \* sin(dlon/2)^2;  c = 2 \* atan2(sqrt(a), sqrt(1-a));    % 地球半径 (公里)  R = 6371;    % 计算距离  d\_1 = R \* c;  d\_2 = round(d\_1,2);  distances\_1(j\_2) = d\_2;    %xinyu  credit\_2 = table2array(Totaldata\_2(:,5));  j\_5 = task\_final(j\_2,2);%renyuan  credit\_1 = credit\_2(j\_5,1) ;    cregitfinal(j\_2) = credit\_1;    end  distances\_2 = distances\_1'%按照人员顺序的距离顺序    l\_1 = task\_final(:,1);  credit\_final = cat(2,l\_1,cregitfinal);%任务对应的信誉    %算信誉值  credit = [];% 初始化一个信用数组  % 遍历矩阵的每一行，算信用值  for m\_1 = 1:size(credit\_final, 1)  current\_value = credit\_final(m\_1, 1); % 当前行的第一列的值    % 检查当前行是否是重复值的一部分  is\_repeated = false;    if m\_1 > 1 && credit\_final(m\_1-1, 1) == current\_value  is\_repeated = true;  end    if m\_1 < size(credit\_final, 1) && credit\_final(m\_1+1, 1) == current\_value  is\_repeated = true;  end    if is\_repeated  % 如果是重复值，找出相同值对应的第二列的最大值  same\_value\_indices = find(credit\_final(:, 1) == current\_value);  max\_second\_col\_value = max(credit\_final(same\_value\_indices, 2));  max\_credit = credit\_final(m\_1,2) ./ max\_second\_col\_value;  credit = [credit; max\_credit];  else  % 如果不是重复值，直接输出第二列的值  only\_credit = credit\_final(m\_1, 2) ./ (credit\_final(m\_1,2) + 1);  credit = [credit; only\_credit];  end  end  % 输出结果  disp(double(credit));    money\_1 = result\_array\_1(:,3);  money\_index = money\_1(task\_final(:,1),:);  Attraction\_1 = [];  potential\_1 = [];  prize\_close = [];  for j\_6 = 1:j\_1  Attraction\_2 = 0.3333 \* distances\_2(j\_6) + 0.6667 \* credit(j\_6);  Attraction\_1(j\_6) = Attraction\_2;  potential\_2 = 0.0745 \* money\_index(j\_6,:) + 0.1231 \* distances\_2(j\_6) + 0.8024 \* credit(j\_6);  potential\_1(j\_6) = potential\_2;  prize\_close\_1 = (wij - 0.1231 ./ distances\_2(j\_6) - 0.8024 \* credit(j\_6)) ./ 0.0745;  prize\_close(j\_6) = prize\_close\_1;  end  Attraction = Attraction\_1';  potential = potential\_1';  close\_prize = prize\_close';  disp('吸引力数为')  disp(Attraction)  disp('可能性数为')  disp(potential)      Final = [];% 初始化一个信用数组  max\_index = [];  Final\_1 = [task\_final,Attraction,potential];        % 找出所有的任务序号  unique\_tasks = unique(Final\_1(:, 1));  results = zeros(length(unique\_tasks), 2); % 第一列为任务序号，第二列为任务是否完成（0 或 1）    for i = 1:length(unique\_tasks)    task\_id = unique\_tasks(i);  task\_rows = Final\_1(Final\_1(:, 1) == task\_id, :);        % 找到分配指标最大的人员  [~, max\_idx] = max(task\_rows(:, 4));  max\_person\_row = task\_rows(max\_idx, :);    % 判断是否完成任务  if max\_person\_row(4) > wij  is\_completed = 1;  else  is\_completed = 0;  end  results(i, :) = [task\_id, is\_completed];  end    % 输出结果,  disp(results);    result\_map{k} = results;  end  final\_table = []; % 初始化最终完成表  all\_tasks = []; % 用于存储所有可能的任务编号  % 先收集所有可能的任务编号  for i = 1:31  data\_1 = result\_map{i};  all\_tasks = unique([all\_tasks; data\_1(:, 1)]);  end  % 初始化最终完成表，每个任务标记为未完成（2）  final\_table = [all\_tasks, 0\*ones(length(all\_tasks), 1)];  % 遍历所有30组数据  for i = 1:31  data\_1 = result\_map{i};  task\_numbers = data\_1(:, 1);  completion\_status = data\_1(:, 2);    for j = 1:length(task\_numbers)  task\_number = task\_numbers(j);  status = completion\_status(j);    % 查找final\_table中该任务的位置  idx = find(final\_table(:, 1) == task\_number);    if status == 1  % 如果任务完成，标记为已完成（1）  final\_table(idx, 2) = 1;  end  end  end  % 检查是否有任务在30组数据中都未完成  for i = 1:length(all\_tasks)  task\_number = all\_tasks(i);    % 查找所有组中该任务的完成状态  completed = false;  for j = 1:30  data\_1 = result\_map{j};  idx = find(data\_1(:, 1) == task\_number);  if ~isempty(idx) && data\_1(idx, 2) == 1  completed = true;  break;  end  end    % 如果所有组中都未完成，将final\_table中的对应任务标记为0  if ~completed  final\_table(i, 2) = 0;  end  end  % 最后，排序final\_table使任务序号从小到大排列  final\_table = sortrows(final\_table, 1);  final\_table\_T = array2table(final\_table, 'VariableNames', {'TaskNumber', 'CompletionStatus'});  % 输出为 Excel 文件  %writetable(final\_table\_T, 'final\_completion.xlsx');  sequence\_3 = size(result\_array\_1,1);  sequence\_4 = size(misson,1);  number\_1 = 1:sequence\_3;  number\_2 = 1:sequence\_4;  result\_array\_1 = [number\_1',result\_array\_1];  misson = [number\_2',misson];    latLon1 = result\_array\_1(:, 2:3); % 数组1的纬度和经度  latLon2 = misson(:, 2:3); % 数组2的纬度和经度    % 计算每个点之间的距离（单位：公里）  distances = pdist2(latLon1, latLon2, 'euclidean') \* 111.32; % 1度大约等于111.32公里    % 设置5公里的阈值  threshold = 10;    % 找出小于5公里的点  [row\_idx, col\_idx] = find(distances <= threshold);      % 第一列是数组1的序号，第二列是数组2中小于5公里的序号  result\_1 = [result\_array\_1(row\_idx, 1), misson(col\_idx, 1)];  result\_misson = sortrows(result\_1,1);%人员任务对应值    %算信誉值  credit\_3 = [];  for m\_1 = 1:size(result\_misson, 1)  current\_value\_1 = result\_misson(m\_1, 1); % 当前行的第一列的值    % 检查当前行是否是重复值的一部分  is\_repeated = false;    if m\_1 > 1 && result\_misson(m\_1-1, 1) == current\_value\_1  is\_repeated = true;  end    if m\_1 < size(result\_misson, 1) && result\_misson(m\_1+1, 1) == current\_value\_1  is\_repeated = true;  end    if is\_repeated  % 如果是重复值，找出相同值对应的第二列的最大xinyuzhi  same\_value\_indices\_1 = find(result\_misson(:, 1) == current\_value\_1);  max\_second\_col\_value\_1 = max(result\_misson(same\_value\_indices\_1, 2));  max\_credit\_1 = result\_misson(m\_1,2) ./ max\_second\_col\_value\_1;  credit\_3 = [credit\_3; max\_credit\_1];  else  % 如果不是重复值，直接输出第二列的值  only\_credit\_1 = result\_misson(m\_1, 2) ./ (result\_misson(m\_1,2) + 1);  credit\_3 = [credit\_3; only\_credit\_1];  end  end        result\_3 = zeros(numRows\_1, 4); % 假设最终结果包含四列    % 遍历第一个数组并提取数据  for z = 1: size(result\_misson,1)  idx = result\_array\_1(:,1) == result\_misson(z,1);  result\_3(z,1:2) = result\_array\_1(idx,2:3);  idx\_1 = misson(:,1) == result\_misson(z,2);  result\_3(z,3:4) = misson(idx\_1,2:3);    end    clear z  clear idx  distance\_2 = [];  for i\_9 =1:size(result\_misson,1)  lat\_5 = result\_3(i\_9,1);  lon\_5 = result\_3(i\_9,2);  lat\_6 = result\_3(i\_9,3);  lon\_6 = result\_3(i\_9,4);  lat\_5 = deg2rad(lat\_5);  lon\_5 = deg2rad(lon\_5);  lat\_6 = deg2rad(lat\_6);  lon\_6 = deg2rad(lon\_6);  % Haversine 公式  dlat = lat\_6 - lat\_5;  dlon = lon\_6 - lon\_5;    a = sin(dlat/2)^2 + cos(lat\_5) \* cos(lat\_6) \* sin(dlon/2)^2;  c = 2 \* atan2(sqrt(a), sqrt(1-a));    % 地球半径 (公里)  R = 6371;    % 计算距离  d\_3 = R \* c;  d\_4 = round(d\_3,2);  distance\_2(i\_9) = d\_4;  distance\_3 = distance\_2';    end  %查找定价  B = result\_misson(:,1);  % 初始化结果数组  result\_4 = zeros(length(B), size(result\_array\_1,4));    % 查找数组2中的序号在数组1中的对应数据  for i\_10 = 1:length(B)  % 查找数组1中序号等于数组2中当前序号的行  idx = result\_array\_1(:, 1) == B(i\_10);    % 存储对应的数据  result\_4(i\_10, :) = result\_array\_1(idx, 4);  end  clear idx      potential\_wij = zeros(size(result\_misson,1),1);  for i\_7 = 1:size(result\_misson,1)  potential\_wij\_1 = 0.0745 \* result\_4(i\_7,1) + 0.1231 ./ distance\_3(i\_7) + 0.8024 \* credit\_3(i\_7);  potential\_wij(i\_7,1) = potential\_wij\_1;  end    result\_5 = zeros(size(result\_misson,1),1);        result\_misson\_1 = [result\_misson,potential\_wij];  disp (result\_misson\_1);  for z\_10 = 1:size(result\_misson\_1,1)  misson\_prize(z\_10,1) = (wij - 0.1231 ./ distance\_3(z\_10) - 0.8024 \* credit\_3(z\_10)) ./ 0.0745;  end  misson\_prize\_1 = [result\_misson\_1(:,1),misson\_prize];  final\_misson = [];  unique\_misson = unique(misson\_prize\_1(:,1)) ;  for z\_9 = 1:size(unique\_misson,1)  current\_value\_2 = unique\_misson(z\_9,1);  same\_unique\_misson = find(misson\_prize\_1(:,1) == current\_value\_2);  if numel(same\_unique\_misson,1) > 1  Max\_prize = max(misson\_prize\_1(same\_unique\_misson,2));  final\_misson = [final\_misson;Max\_prize];    else  oringnal\_prize = misson\_prize\_1(same\_unique\_misson,2);  final\_misson = [final\_misson;oringnal\_prize];  end        end      disp("定价表");  disp(final\_misson);  final\_misson\_1 = [result\_misson\_1(:,1),final\_misson];        %找到最小定价  [uniqueTasks, ~, idx] = unique(final\_misson\_1(:,1)); % 找到唯一的任务  minPrices = accumarray(idx, final\_misson\_1(:,2), [], @min); % 找到每个任务的最小定价  % 将结果组合在一起  result\_7 = sortrows([uniqueTasks, minPrices],1);    writematrix(result\_7, '定价3.xlsx') |