基于聚类分析和多目标规划的众包任务定价策略

摘要

在移动互联网的迅猛发展的时代背景下，一批“拍照赚钱”APP 应运而生，这些 APP 的运营模式实质上是一种自助式的拍照任务众包模式。这种模式具有调查成本低、数据真实性高、调查周期短等众多优势，因此也就有了比传统市场调查方式更为可观的发展前景。

本题主要关注的就是拍照任务众包模式中的定价策略设计问题。

针对问题一，根据经验和相关资料推测影响任务定价的因子，对这些因子进行聚类与量化，利用已有的所有数据点进行 Robust 回归分析，确定原定价策略。经过分析发现原定价策略主要关注会员数量，而忽视了会员信誉值等其它因素。接着，利用已有的任务完成数据点进行 Robust 回归分析，通过与原定价策略进行对比，分析任务未完成的原因。发现任务未完成的主要原因是定价过低；同时结合未完成任务分布区域的实地特殊情况进行分析。

针对问题二，设计新的定价策略，首先包含原定价策略中已考虑的因子——会员数量；其次，提高会员信誉值的影响权重；最后，引入任务完成时间排序等新的影响因子。从会员完成任务的动机出发，不断分解和细化影响因子，利用多目标规划得到更优的定价策略。从任务完成数量和支付酬金金额两方面与原定价策略进行比较与评价。

针对问题三，引入原创算法——链式成组算法，添加任务打包机制和任务包定价机制，对原有的任务完成情况与任务定价的关系进行修正。通过分析打包机制的内在原因，比较任务联合发布机制引入前后的任务完成情况。

针对问题四，运用综合评价法，对两类评价对象——企业和会员，根据已有的三个指标建立企业满意度和会员满意度，统一为帕累托最优的判断依据——总体满意度；并据此得到特定权重下的最优定价策略。

最后，对所建立模型的优缺点给出了客观的评价，并对模型的进一步改进进行展望。

关键字： 拍照赚钱应用　定价策略　多目标规划　模拟退火法　聚类分析　多元回归

一、问题重述

1.1 问题背景

在移动互联网的迅猛发展的时代背景下，一批“拍照赚钱”APP 应运而生，如蚂蚁众包 [1]、高德淘金 [2] 等等。用户仅需注册成为会员，从 APP 上领取并完成指定内容的拍照任务，即可赚取相应任务标定的酬金，而拍摄的照片信息则可以作为企业商业检查和市场调研的依据。实质上，这样的运营模式就是一种自助式的拍照任务众包模式。相比传统的市场调查方式，基于移动互联网的自助式拍照任务众包模式具备调查成本低、数据真实性高、调查周期短等众多优势，因此也就有了更为可观的发展前景。

1.2 问题的提出

众包指的是一个公司或机构把过去由员工执行的工作任务，以自由自愿的形式外包给非特定的（而且通常是大型的）大众网络的做法 [3]。在拍照任务众包的过程中，如何为一个拍照任务定价是核心要素。若定价过低，则任务将无人问津，导致调查无法进行；若定价过高，则企业获取照片信息的成本也相应升高，影响企业利润。只有合理定价，才能取得完成任务数量和支付酬金金额的整体最优，才能体现众包模式的优势。这样的规律与基于经验的判断也是一致的。本题主要关注的就是拍照任务众包模式中的定价策略问题，具体研究的问题如下：

1. 根据已结束项目的各个任务的地理位置、定价和完成情况以及会员的位置、信誉值、开始预定时间和预定限额，探究该项目中任务的定价规律，并分析任务未完成的原因。
2. 在问题 1 的基础上，根据已结束项目的各个任务的地理位置、定价和完成情况以及会员的位置、信誉值、开始预定时间和预定限额，设计新的定价策略，并与问题 1 中原定价策略进行比较。
3. 在问题 2 的基础上，根据已有信息，加入“任务联合打包发布”的机制，修改得到新的定价策略，并与问题 2 中的定价策略进行比较。
4. 在前三问分析和设计定价策略的基础上，根据新项目的各个任务的地理位置，为该项目设计定价策略，并对定价策略的实施效果进行评价。

二、问题分析

本题主要关注的是拍照任务众包模式中的定价策略问题。问题一分析原定价策略，问题二设计新的定价策略，问题三在定价策略中加入任务联合发布机制，问题四将定价策略应用于新项目。四个问题层层递进，不断得到更优的定价策略模型并加以分析和应用。下面对四个问题的具体要求和大致思路进行分析。

2.1 问题一

问题一要求研究已结束项目中任务的定价策略，并对任务未完成的原因进行解释。根据经验和相关资料可以推测，任务点的经纬度、任务点附近的会员数量和信誉值、任务点与会员之间的距离都可能对任务定价产生影响。如何判断任务定价与这些因素的相关性，是问题一的关键。对此，可利用已有的所有数据点进行回归分析，判断相关性，建立任务定价与影响因子的关系式，确定原定价策略。解释任务未完成的原因，可利用已有的任务完成的数据点进行回归分析，得到更为合理的定价策略，与原定价策略进行对比，分析两者的差距；也可结合未完成任务分布区域的实地特殊情况加以解释。

2.2 问题二

问题二要求为已结束项目中的任务设计新的定价策略，并与原定价策略进行比较。设计新的定价策略时，首先必须包含原定价策略中考虑的因子；其次，通过问题一中对任务未完成原因的分析，可调整影响因子的权重关系；最后，可根据经验和资料引入新的影响因子。如何科学地将复杂的影响因子结合起来，形成更优的定价策略是问题二的关键。对此，可从会员完成任务的动机出发，不断分解和细化影响因子，得到新的定价策略。与原定价策略的比较可从任务完成数量和支付酬金金额两方面进行比较，并分析其背后的原因。

2.3 问题三

问题三要求在新的定价策略中引入任务联合发布的机制，并分析对最终任务完成情况的影响。确定为引入新机制而要做出的修改是问题三的关键。确定需要做出的修改可分为两个方面，任务联合发布本身需要添加的方法和规则，以及任务打包后对原有关系或约束的影响。分析最终任务完成的情况，也就是要比较任务联合发布机制引入前后的任务完成情况，并分析该机制对结果产生影响的本质原因。

2.4 问题四

问题四要求对新项目设计定价策略，并评价实施效果。在完成前三问之后，存在三种模型，分别是原定价模型，未打包的定价策略，以及打包的定价策略。对于新项目，哪一种策略有着更佳的效果不得而知，因此，利用综合评价的方法，根据帕累托最优的理念，从企业和会员两个角度，构建满意度的概念作为评价标准，可得到一个和企业实际状态相关的总体满意度。根据总体满意度，可以评价并选出具有最优实施效果的定价策略。

三、模型的假设

1. 用户心理指标不随定价策略的改变而改变；
2. 所有用户的心理指标相近；
3. 商家具有固定财源用来支付任务酬金；
4. 认为在可以完成某任务时，用户决定是否完成任务仅与任务酬金和任务距离有关；
5. 假设问题研究区域（珠三角地区）居民支付能力相同；
6. 假设会员位置的整体分布不发生显著变化；
7. 假设任务只在定价之前产生，即不会动态产生任务；
8. 针对第三问，假设用户群体对捆绑销售无主观态度；
9. 假设每个任务以及任务包由单个用户独立完成；
10. 针对第三问，认为同一包内所有任务同时刻完成；

四、符号系统

表 1　符号系统

符号	符号含义	单位
Lon_i	第 i 个任务的经度	度
Lat_i	第 i 个任务的纬度	度
ρ_i	第 i 个任务的会员密度	个
Den_i	第 i 个任务的相对会员密度	/
μ_i	第 i 个会员的信誉值	/
$\mu_i^{'}$	第 i 个会员的相对信誉值	/
Rep_i	第 i 个任务的相对信誉值	/
Dis_i	第 i 个任务点到最近会员聚类中心的距离	千米
p_i	第 i 个任务的定价	元
X_i	第 i 个任务的完成情况	/
Z	企业收益	元
d_{iout}	任务包中各点到最近的会员聚类中心距离的平均值	千米
d_{iin}	会员在任务包中各点之间移动的距离	千米
S_1	企业满意度	/
$s_2(i)$	第 i 个任务点的会员满意度	/
c_{ji}	第 j 个会员完成第 i 个任务所需付出的代价	/
S_2	会员满意度	/
N_m	会员总数	个
N_t	任务总数	个
S	总体满意度	/

五、问题建模与求解

5.1 问题一——多元回归寻找定价规律，失败任务原因分析

原定价策略的探究

为确定已结束项目的定价策略，需要找出该项目中影响定价的因素及各因素的影响程度。认为影响定价的因素主要有：任务点的经度、任务点的纬度、任务点附近区域会员的数量、任务点附近区域会员的信誉值、任务点到会员的距离。下面建立指标定量衡量五个影响因素。

定义第 i 个任务点的经度和纬度分别为 Lon_i 和 Lat_i，则前两个影响因素可直接表示为 Lon_i 和 Lat_i，无需转化。

定义第 i 个任务点的会员密度 ρ_i 为，以该点为中心 $0.05°$ 对应地理长度为半径的范围内的会员数。对 ρ_i 进行如下的尺度变换，得到第 i 个任务的相对会员密度 Den_i：

$$Den_i = \begin{cases} 0 & \text{如果 } \rho_i = 0 \\ \log \rho_i + 3 & \text{其它} \end{cases}$$

用相对会员密度 Den_i 衡量第 i 个任务点附近区域会员的数量。

定义第 i 个会员的信誉值为 μ_i，第 i 个任务点受信誉值影响的范围是，以该点为中心 $0.05°$ 对应地理长度为半径的圆形区域。由于会员的信誉值跨度很大，因此对范围内每个会员的信誉值进行如下的尺度变换，得到会员相对信誉值 $\mu_i^{'}$：

$$\mu_i^{'} = \log \mu_i + 3$$

定义范围内所有会员的编号分别为 $a_1, a_2, \cdots, a_n$，对范围内所有会员的会员相对信誉值取平均，得到任务点的相对信誉值 Rep_i：

$$Rep_i = \frac{\sum_{j=1}^{n} \mu_{a_j}^{'}}{n}$$

用相对信誉值 Rep_i 衡量第 i 个任务点附近区域会员的信誉值。

为衡量任务点到会员的距离，首先使用 K-means 聚类算法对会员的地理位置进行聚类。搜索会员地理位置发现，会员绝大部分位于佛山、广州、东莞、深圳四座城市（香港境内的点位于深港边界，归类于深圳），因此取 $K = 4$，聚类结果如图所示。

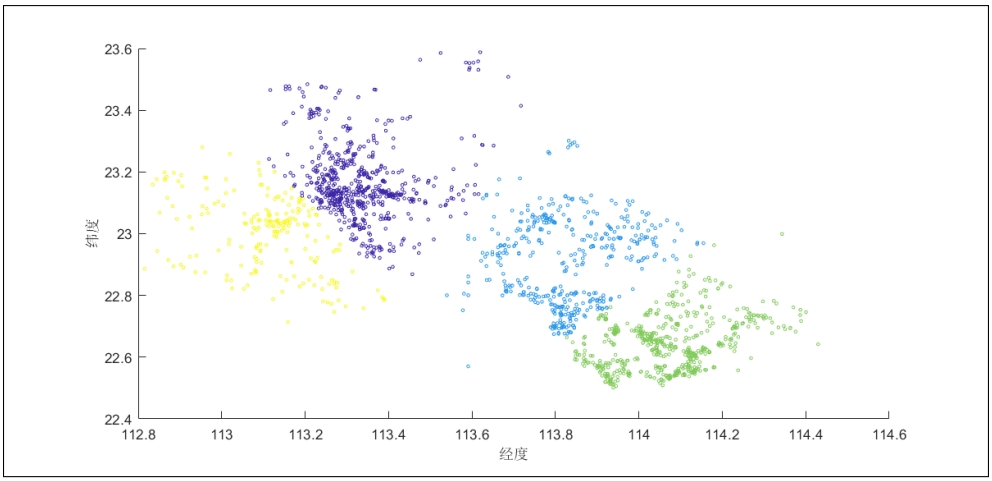

图 1　会员聚类示意图

聚类后确定各个聚类的中心。由于聚类的分界与城市的分界基本一致，而会员通常不会跨城市完成任务，因此用任务点到最近会员聚类中心的距离衡量任务点到会员的距离。定义第 i 个任务点到最近会员聚类中心的距离为 Dis_i。

至此，定价策略中影响任务定价的五个因子分别为：任务点的经度（Lon_i）、任务点的纬度（Lat_i）、任务点的相对会员密度（Den_i）、任务点的相对信誉值（Rep_i）、任务点到最近会员聚类中心的距离（Dis_i）。

定义第 i 个任务的定价为 p_i。基于文献 [4] 中的思想，以上面的五个因子作为解释变量，提出任务定价的 Robust 回归模型，如公式(1)所示。

$$p_i = \beta_1 Lon_i + \beta_2 Lat_i + \beta_3 Den_i + \beta_4 Rep_i + \beta_5 Dis_i + \beta_0 \tag{1}$$

使用附件一中的所有数据点进行回归分析，得

表 2　所有数据点回归分析参数表

β_1	β_2	β_3	β_4	β_5	β_0
1.4878	1.4768	-2.2277	-0.0985	1.4454	-128.0681

已结束项目的定价策略为：

$$p_i = 1.4878 Lon_i + 1.4768 Lat_i - 2.2277 Den_i - 0.0985 Rep_i + 1.4454 Dis_i - 128.0681 \tag{2}$$

由公式(2)可知，定价策略主要考虑了任务的相对会员密度，对任务的相对信誉值则几乎没有考虑。

任务未完成原因的分析

图2和图3分别为任务分布图和会员分布图。图2中，每个圆点对应一个任务，绿色圆点对应完成的任务，红色圆点对应未完成的任务，圆点越大，任务定价越高。图3中，每个圆点为一个会员，圆点的颜色越深，会员信誉值越高。

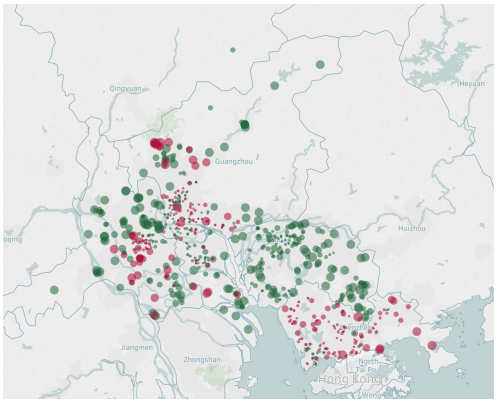

图 2　任务分布图

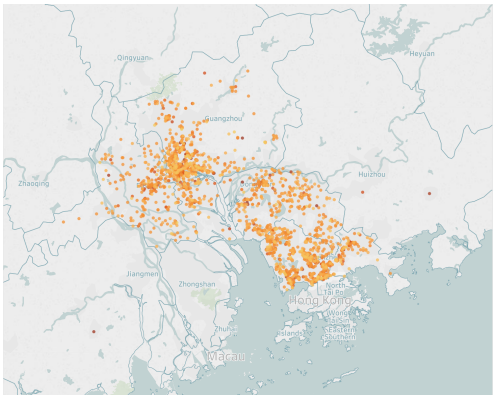

图 3　会员分布图

观察图2，发现未完成的任务（红色圆点）大都定价较低（半径小），则定价低很可能是任务未完成的原因。进一步对比两幅图，以深圳地区与东莞地区为例进行比较。深圳的会员密度（圆点密度）明显高于东莞，会员信誉值（圆点颜色深度）也高于东莞，根据经验判断，深圳的任务完成情况应该优于东莞，但事实恰恰相反，且任务完成情况的分界线与深圳东莞的地理区划分界线相当吻合，如下图所示。

图 4　深圳东莞分界线附近任务完成情况

原因就在于，深圳的任务定价（圆点大小）明显低于东莞。分析佛山禅城区及广州天河区附近的任务未完成情况可以得到同样的结论。由此，该已结束项目中，任务未完成的主要原因是定价过低，定价与任务点的情况严重不符。

进一步分析定价过低的原因。仅利用任务完成的数据点，运用公式(1)表示的Robust 回归模型进行分析，得

表 3　所有数据点回归分析参数表

β_1	β_2	β_3	β_4	β_5	β_0
1.7431	1.1917	-2.1470	-0.2939	1.3610	-149.9739

由此得出的定价策略为：

$$p_i = 1.7431Lon_i + 1.1917Lat_i - 2.1470Den_i - 0.2939Rep_i + 1.3610Dis_i - 149.9739 \quad (3)$$

比较公式(2)和公式(3)中同一因子对应参数值的差异，发现公式(2)中任务点相对信誉值的权重仅为公式(3) 中的约三分之一，这说明原定价策略对任务点相对信誉值的考虑明显不足。观察图2和图3，深圳与佛山未完成任务点附近的会员密度较高，应当降低定价，但这些任务点的相对信誉值较高，应当提高定价。原定价策略正是因为过多地考虑了会员密度的影响，而忽略了任务点相对信誉度的影响，所以对这些任务点的定价过低，导致未能被完成。

另外，发现图2中也存在少数任务定价相对合理但也未完成的情况，可能的原因有二：有些任务点过于偏远，导致会员不愿前往完成任务；有些任务点位于特殊地形（如山地、水域等），完成任务的代价不仅仅取决于其位置，但原定价策略对这类特殊情况并不适用，导致这类任务点的定价偏低。

5.2　问题二——多目标规划设计新定价，多角度比较原定价策略

定价策略模型的构建

定义第 i 个任务的完成情况为 X_i，若任务完成，则 $X_i = 1$，若任务未完成，则 $X_i = 0$。根据众包模式的定义及优势，设计新的定价策略应该有如下两个优化目标，即完成任务数量最大化和支付酬金金额最小化。

$$\begin{aligned} &\text{maximize} \quad \sum_i X_i \\ &\text{minimize} \quad \sum_i X_i p_i \end{aligned}$$

除了这两个目标以外，为了体现信息量，即任务完成情况对于此类拍照赚钱公司的重要价值，增加一惩罚性指标，用以衡量未完成任务给企业带来的损失：

$$\text{minimize} \quad \sum_i (1 - X_i)$$

对于多目标优化问题，本题采用将多目标优化转换为单目标优化的方法进行求解。完成任务数量和支付酬金金额的量纲不同，首先统一量纲：若任务完成，则企业借助获得的照片信息能够有所收益，但同时也要支付酬金；若任务未完成，则企业由于照片信息的缺失有一定的机会成本。定义企业收益指标 Z 为优化目标，则有：

$$Z = \sum_i X_i - \sum_i (1 - X_i) - \frac{k}{85} \sum_i X_i p_i \tag{4}$$

公式(4)中，第一部分表示任务完成后借助获得的照片信息取得的收益，第二部分表示任务未完成而产生的机会成本，第三部分表示任务完成所支付的酬金，k 是可调参数，用于调节各个部分的权重关系，k 越大，支付酬金对企业收益的影响能力就越大。在问题中二中，$k = 1.2$。通过公式(4)，完成任务数量和支付酬金金额被统一到了企业收益中。最终的优化目标如下，即企业收益的最大化。

$$\text{maximize} \quad Z = \sum_i X_i - \sum_i (1 - X_i) - \frac{k}{85} \sum_i X_i p_i$$

要设计定价策略，需要确定 X_i 与 p_i 的关系，即任务定价对任务完成情况的影响。根据经验可以判断，会员在任务定价高于完成任务的代价时才会试图完成任务，而完成任务的代价在很大程度上是会员自身感受的一个估计值。因此，可运用定价策略中的需求导向定价法 [5] 来确定任务定价对任务完成情况的影响。在需求导向定价法中，主要依据购买者感受的价值，而非产品的成本来定价。

这里主要关注任务自身以及任务关于会员的性质的定量情况。在问题一中，发现任务定价与相对会员密度有较强的线性关系，因此首先引入相对会员密度。定义会员完成第 i 个任务的收益为 $p_i - kDen_i - Th$，其中 k 和 Th 均为常数。会员密度越低，则大部分会员距任务点的距离较远，因此 $kDen_i$ 可视作会员完成任务的代价中与距离相关的部分，如路费成本、时间成本等；Th 则可视作与距离无关的部分，如投入的精力等，这个量主要取决于会员自身的感受，本题中认为每个会员的感受相同。则任务完成情况 X_i 满足：

$$X_i = \begin{cases} 0 & \text{如果 } p_i - kDen_i - Th \leq 0 \\ 1 & \text{如果 } p_i - kDen_i - Th > 0 \end{cases} \tag{5}$$

求解公式(5)中的常数 k 和 Th。由于会员的心理指标不随定价策略的改变而改变，故使用已结束项目中所有数据点对任务定价和会员密度进行线性回归，回归结果为：$p_i = -2.2740 Den_i + 75.2988$。因此，认为公式(5)中 $k = -2.2740$，$Th = 75.2988$。任务

完成情况与任务定价的具体关系为：

$$X_i = \begin{cases} 0 & \text{如果 } p_i + 2.2740 Den_i - 75.2988 \leq 0 \\ 1 & \text{如果 } p_i + 2.2740 Den_i - 75.2988 > 0 \end{cases}$$

然而，p_i 不是随机变动的，而是受到第 i 个任务点情况的影响。考虑三个因素：任务被选择的时间排序、任务到最近会员聚类中心的距离和任务的相对信誉值。考虑任务被选择的时间排序是因为，如果一个任务迟迟不被选择，即它不被高信誉值的会员所偏好，则它的定价可能偏低，是需要调整的；考虑任务到最近会员聚类中心的距离是因为，这一因素对任务的受欢迎程度有影响，也就对任务被选择的时间排序有影响；考虑任务的相对信誉值是因为，正如业绩好的员工应当获得更多的奖金，信誉值高（完成度高）的会员完成任务获得的酬金应当更高，以激励会员努力提高信誉值 [6]。

关于以上三个因素，有以下三个基于经验的规律：

- 一个任务被选择的时间排序越靠前，这个任务的受欢迎度越高。
- 一个任务距会员聚类中心越近，定价越高，越可能受欢迎。
- 一个任务被选择的时间排序越靠前，这个任务周围高信誉值的会员可能越多。

总结上面三个规律可得，一个任务的相对信誉值越高，这个任务很可能到最近会员聚类中心的距离越短，定价越高。这种关系可表示为

$$Rep_i \propto \frac{p_i}{Dis_i}$$

将定价 p_i 移至一侧可得

$$p_i = \alpha Dis_i Rep_i + \beta$$

其中，α 和 β 为可调参数。调整 α 和 β 的值，即可改变定价 p_i 的值，进而改变任务完成情况，最终影响企业收益。

综上，问题二的优化模型描述如下所示。

$$\begin{aligned} \text{maximize} \quad & Z = \textstyle\sum_i X_i - \sum_i (1 - X_i) - \frac{k}{85} \sum_i X_i p_i \\ \text{subject to} \quad & X_i = \begin{cases} 0 & \text{如果 } p_i + 2.2740 Den_i - 75.2988 \leq 0 \\ 1 & \text{如果 } p_i + 2.2740 Den_i - 75.2988 > 0 \end{cases} \\ & p_i = \alpha Dis_i Rep_i + \beta \\ & p_i, Dis_i, Rep_i, Den_i > 0 \\ & \alpha, \beta \in \mathbb{R} \end{aligned}$$

优化的实质就是调节 α 和 β 的值，使企业收益 Z 最大化。通过 Z 最大时的 α 和 β，可计算出对应的定价 p_i，得到最佳的定价策略。

定价策略模型的求解

运用模拟退火算法实现优化过程，求解近似最优解。模拟退火算法的等温内循环流程与降温外循环流程如图5和图6所示。

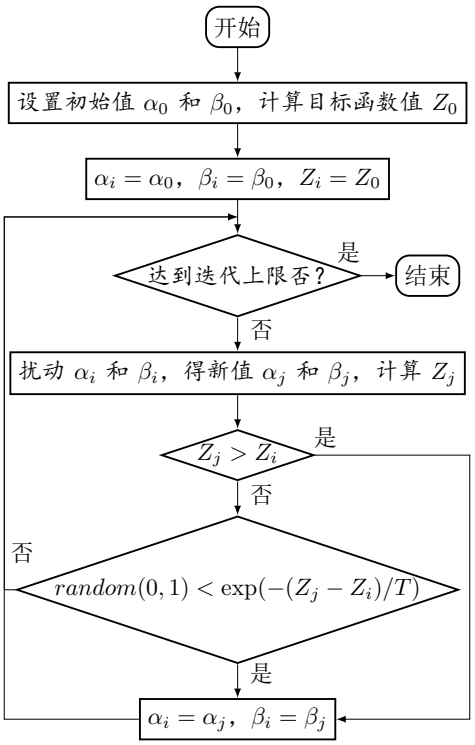

图 5　等温内循环流程图

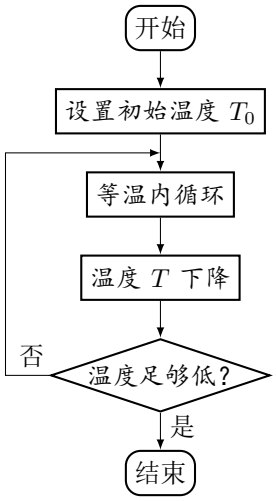

图 6　降温外循环流程图

实际求解过程中，设马尔科夫链长度（等温内循环的迭代上限）为 1000，初始温度为 100，最终温度为 1，衰减参数为 0.95。优化的最终结果为

表 4　问题二优化结果

α	β	Z	$\sum_i X_i$	$\sum_i X_i p_i$
-0.945	75.848	-62.7557	822	61023

需要说明的是，这里仅关注 Z 的相对大小，而不考虑其本身值的大小。

定价策略比较

从商家的角度来看，相比原定价策略，新定价策略考虑了相对信誉值、任务被选择的时间排序、任务点到最近会员聚类中心的距离等因素，构建了更加全面、更加符合实际的定价策略模型。由于考虑因素全面性的提高，区域之间任务完成情况出现较

大差距的情况有了明显的缓解。图7为新定价策略下的任务分布图，可以看到区域之间任务完成情况更加平衡，这对商家扩展业务区域是十分有利的。

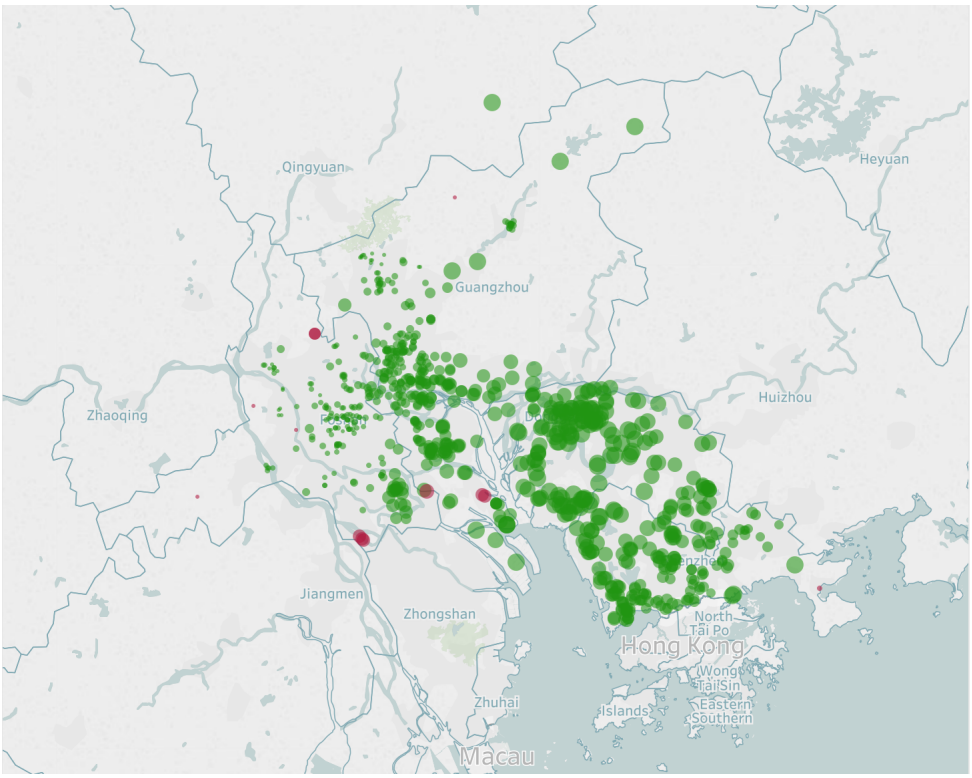

图 7　新定价策略下的任务分布图

已结束项目中，$\sum_i X_i = 522$，$\sum_i X_i p_i = 36446$。将原定价策略中的任务定价和任务完成情况代入新模型，得：$Z = -311.6571$。新定价策略中的企业收益远高于换算后原定价策略中的企业收益，这意味着，企业通过相对更少的钱获得了更多任务被完成所提供的信息。

5.3　问题三——链式成组算法打包，任务包影响力分析

链式成组算法

链式成组算法遵循一定的规则，尽可能地将分布紧密的任务点进行打包。

首先运用贪心算法将所有的任务点连成一条链。具体过程为：随机选取一个任务点为起始点，从其余任务点中选出与起始点距离最近的点，将它与起始点相连成链。依次不断地从未被选择的任务点中选出与最新加入链的任务点距离最近的点，将其连到链上，直至所有点都依次相连。将形成的链称为“任务链”。

在“任务链”的基础上进行打包。设定任务点距离阈值 L 和打包任务点最大数量阈值 N。从起始点出发沿“任务链”扫描，若当前任务点与下一任务点之间的距离大于阈值 L 或以当前任务点为最后一个点的子链上的任务点数量等于 N，则切断当前任务点与下一任务点之间的连接；否则，继续沿“任务链”扫描。直至扫描完整条“任务链”。

此时，将仍然相连的任务点打包。调节两个阈值，可以达到较为理想的打包效果。以上的过程可表示为如下的算法：

算法 1: 链式成组算法

Input: 任务链

Output: 任务打包情况

```
当前任务点为起始点;
while 当前任务点不是任务链上最后一个任务点 do
    if 当前任务点与下一任务点之间连接的长度大于 L 或以当前任务点为最后一个点的子链上的任务点数量等于 N then
        切断当前任务点与下一任务点之间的连接;
    else
        继续扫描下一任务点;
将仍然相连的子链中的任务点打包;
```

打包任务定价机制

记一个包中的所有的任务分别为第 $a_1, a_2, \cdots, a_n$ 个任务，则打包后这个包的总定价为 $n \times \min_{i=1,\cdots,n} p_{a_i}$。其含义是，将包中所有任务的定价都统一到包中任务最低的定价，使得完成包中任务需要支付的酬金低于不打包的情况。

任务完成情况与任务定价的关系

与问题二类似，运用定价策略中的需求导向定价法来确定任务完成情况和任务定价的关系。当第 i 个任务未被打包时，X_i 的表达式不变。这里进一步讨论第 i 个任务被打包的情况。定义会员完成第 i 个任务的收益为 $p_i - k^{'}(d_{iout} + d_{iin}) - Th^{'}$，其中 $k^{'}$ 和 $Th^{'}$ 均为常数。d_{iout} 的含义为任务包中各点到最近的会员聚类中心距离的平均值，d_{iin} 的含义是会员在任务点之间移动的距离，因此 $k^{'}(d_{iout} + d_{iin})$ 可视作会员完成任务的代价中与距离相关的部分；$Th^{'}$ 则可视作与距离无关的部分。则任务完成情况 X_i 满足：

$$X_i = \begin{cases} 0 & \text{如果 } p_i - k^{'}(d_{iout} + d_{iin}) - Th^{'} \leq 0 \\ 1 & \text{如果 } p_i - k^{'}(d_{iout} + d_{iin}) - Th^{'} > 0 \end{cases} \tag{6}$$

求解公式(6)中的常数 $k^{'}$ 和 $Th^{'}$。使用已结束项目中所有数据点对任务定价和任务点到最近会员聚类中心的距离进行线性回归，回归结果为：$p_i = 2.0383 Dis_i + 68.1819$。由于 Dis_i 和 $d_{iout} + d_{iin}$ 都是与距离相关的代价量，因此，认为公式(6)中 $k^{'} = 2.0383$，$Th^{'} = 68.1819$。任务完成情况与任务定价的具体关系为：

$$X_i = \begin{cases} 0 & \text{如果 } p_i - 2.0383(d_{iout} + d_{iin}) - 68.1819 \leq 0 \\ 1 & \text{如果 } p_i - 2.0383(d_{iout} + d_{iin}) - 68.1819 > 0 \end{cases}$$

定价策略模型的构建与求解

综上，问题三的优化模型描述如下所示。

$$\text{maximize} \quad Z = \sum_i X_i - \sum_i (1 - X_i) - \frac{k}{85} \sum_i X_i p_i$$

$$\text{subject to} \quad \begin{aligned} &X_i\text{未被打包时} \quad X_i = \begin{cases} 0 & \text{如果 } p_i + 2.2740 Den_i - 75.2988 \leq 0 \\ 1 & \text{如果 } p_i + 2.2740 Den_i - 75.2988 > 0 \end{cases} \\ &X_i\text{被打包时} \quad X_i = \begin{cases} 0 & \text{如果 } p_i - 2.0383(d_{iout} + d_{iin}) - 68.1819 \leq 0 \\ 1 & \text{如果 } p_i - 2.0383(d_{iout} + d_{iin}) - 68.1819 > 0 \end{cases} \\ &p_i = \alpha Dis_i Rep_i + \beta \\ &p_i, d_{iout}, d_{iin}, Rep_i, Den_i > 0 \\ &\alpha, \beta \in \mathbb{R} \end{aligned}$$

优化的实质就是调节 α 和 β 的值，使企业收益 Z 最大化。

同样, 运用模拟退火算法实现优化过程，求解近似最优解。实际求解过程中，设马尔科夫链长度（等温内循环的迭代上限）为 100，初始温度为 100，最终温度为 1，衰减参数为 0.9。优化的最终结果为

表 5　问题三优化结果

α	β	Z	$\sum_i X_i$	$\sum_i X_i p_i$
2.4103	68.3345	-47.0144	805	57541

任务完成情况的分析

多个任务联合打包发布的做法，与经济学中捆绑销售的理念相契合。捆绑定价，是企业的一种营销策略，即生产者将一种产品与其他产品组合在一起以一个价格出售。它通过降低消费者的搜寻成本来增加他们的购买欲，同时减少了对每个任务的宣传等花销，即实现了范围经济。在此之上，由大数定律和中心极限定理可以知道，消费者对产品的估价的平均值接近每件产品期望值，那么消费者剩余会收敛于零。在本题情景下，即会员对任务定价的期望减小，使得收益趋近于支付意愿（成本），即会员的消费者剩余收敛于零 [7]。

由模型求解的结果，可得 $\sum_i X_i = 805$，$\sum_i X_i p_i = 57541$。比较支付酬金金额和完成任务数量可以发现，在大幅减少支付酬金金额的情况下，引入打包机制前后，完成任务的数量保持相近水平。这表明，引入打包机制可在保持完成任务数量的基础上大幅减少支出。

5.4 问题四——综合评价法确定新策略，各模型效果评估

综合评价法确定定价策略

运用综合评价法对定价策略的实施效果进行评价。评价对象有两个：企业和会员。从企业角度进行评价的原因在于，定价策略本身就是为企业盈利而设计的，企业收益的提高是定价策略的根本目的。从会员角度进行评价的原因在于，用户的满意程度对其未来的购买行为有很大的影响 [8]，适当保持会员满意度本质上有利于企业的长远发展。通过任务完成度、支付酬金（会员收益）、会员成本三个指标，构建企业满意度和会员满意度。

定义企业满意度 S_1 的表达式如下：

$$S_1 = \frac{\sum_j X_j}{\sum_j X_j p_j}$$

分子 $\sum_j X_j$ 为完成的任务总数，分母 $\sum_j X_j p_j$ 为对完成任务的会员支付的酬金。因此，S_1 的实际含义是企业支付单位酬金所能获得的任务完成度。

对 S_1 进行标准化，标准化后的表达式如下：

$$S_1 = \frac{2}{\pi} \arctan\left(\frac{r\sum_j X_j}{\sum_j X_j p_j}\right)$$

其中，r 为可调参数。因为 $\dfrac{\sum_j X_j}{\sum_j X_j p_j}$ 的范围在零附近的极小范围内，所以要通过系数 r 进行拉伸，使 S_1 较为均匀地分布在 $(0,1)$ 区间内。经尝试，$r = 300$ 时效果较好。

定义会员满意度之前，首先定义第 i 个任务点的会员满意度 $s_2(i)$，其表达式如下：

$$s_2(i) = \frac{p_i}{\dfrac{\sum_{j=1}^{N_m} c_{ji}}{N_m}}$$

其中，c_{ji} 为第 j 个会员完成第 i 个任务所需付出的代价，N_m 为会员的总数。分子 p_i 为完成第 i 个任务获得的酬金，分母为平均每个会员完成第 i 个任务需要付出的代价。因此，$s_2(i)$ 的实际含义是平均每个会员付出一个单位的代价所获得的酬金。

对 $s_2(i)$ 进行标准化，标准化的表达式如下：

$$s_2(i) = \frac{2}{\pi} \arctan\left(\frac{p_i}{\dfrac{\sum_{j=1}^{N_m} c_{ji}}{N_m}}\right)$$

记任务总数为 N_t，定义会员满意度的表达式如下：

$$S_2 = \sqrt[N_t]{\prod_{i=1}^{N_t} s_2 i}$$

将企业满意度和会员满意度相结合，构建评价方程，得到总体满意度 S 的表达式如下：

$$S = (S_1)^p (S_2)^{1-p}$$

其中，权重 p 为可调参数。

使用总体满意度 S 评价本题中涉及的三个模型。

表 6　模型满意度比较

	S_1	S_2	S
原模型	0.8544	0.4962	0.8675
未打包新模型	0.8449	0.5249	0.6984
打包新模型	0.8460	0.5128	0.6925

可见，在 $p = 0.6$ 的情况下，未打包模型具有最高的总体满意度。因此对于附件三中给定的新项目，选择未打包模型作为定价策略。

模型效果分析

显然，原模型的企业满意度最高，打包新模型次之，未打包新模型最低；会员满意度未打包新模型最低，打包新模型次之，原模型最低；总体满意度未打包新模型最高，打包新模型次之，原模型最低。

经分析可知以上结论合理：

1. 由于原模型的定价企业仅仅从自身的直接收益水平考虑，所以它的企业满意度最高，会员满意度最低。
2. 由于打包新模型在几乎不减少任务完成数量的基础上减少了支付酬金的总量，所以它的企业满意度高于未打包新模型，而且它的会员满意度低于后者。
3. 总体满意度的最大化意味着在这个企业 -会员的经济系统中达到了帕累托最优，即不可能不通过改变某一方的利益以增加另一方的利益。

在权重 $p = 0.6$ 的条件下，总满意度如上。而随着权重的改变，总体满意度的排序也会随之改变。这个结论的隐含意义是，权重等价于企业的决策重心。当 p>0.5 时，企业的决策重心在提高企业的盈利水平上；当 p<0.5 时，企业的决策重心在提高用户满意度，拓展市场占有额。

基于这些结论，我们还可以得到对于此类拍照赚钱企业的指导思想：早期企业可以通过将重心放在提高用户满意度来占领市场；成熟企业可以通过将决策重心放在提高企业盈利水平上，利用打包等策略提高企业收益率。

回到本项目，由于对于 $p = 0.6$ 的条件未打包模型的总体满意度最高，所以对于 $p = 0.6$ 的企业，未打包定价模型具有最好的效果。具体而言，在这个定价模型中，企业花费每单位酬金可获得的任务完成度和会员付出每单位代价可得到的酬金能够达到帕累托最优 [9]，即不可能不通过减少某一方的满意度来增加另一方的满意度。

六、模型评价与展望

本文分析了众包这一新兴商业模式，从分析原有定价策略出发，寻找可能的、有较大影响力的因素，逐步构建起一个较为完善的全新定价策略。其中在将会员信誉值、会员地理信息、任务地理信息，以及可能发生的特殊情况纳入考量之后，把经济学中的需求定价策略、捆绑销售定价、众包定价策略等工具进行综合运用，以求更合理、更有效、更精确地为众包营销模式确立定价策略。本文虽然有不足之处，如利用数据对用户心理地精准定位上并不能达到完美，对会员的任务上限定位较为模糊，但是我们拥有非常多的亮点，如新算法（链式成组算法）的发明、逻辑推理以及论证的强严密性、定量与定性分析相结合、数据与经济学原理的相互支持、直观的数据可视化、数据科学与优化模型在社科问题上的整合等等。如今中国处于全面深化改革的风口，新兴的商业模式如雨后春笋，需要有科学的市场分析和定价策略予以扶持，所以与本文相似的数理方法和社会科学相结合的模型正是今后需要大力研究和发展的方向。

七、参考文献

[1] 蚂蚁众包. 蚂蚁众包 -第三方互联网户外信息采集. `http://www.antzb.com/collection.html`, 2017.

[2] 高德淘金. 高德淘金. `http://gxd.amap.com/`, 2017.

[3] Jeff Howe. *Crowdsourcing: Why the Power of the Crowd is Driving the Future of Business*. Crown Publishing Group, 2008.

[4] 毛可. 软件众包任务的定价模型与人员匹配方法研究及工具实现, 2014.

[5] Shuba Srinivasan, Koen Pauwels, and Vincent Nijs. Demand-based pricing versuspast-price dependence: A cost–benefit analysis. In *Journal of Marketing*, volume 72, pages 15–27, 2008.

[6] Jing Wang, Panagiotis G. Ipeirotisy, and Foster Provost. A framework for quality assurance in crowdsourcing. In *Social Science and Electronic Publishing*, 2013.

[7] Stefan Stremersch and Gerard J. Tellis. Strategic bundling of products and prices: A new synthesis for marketing. In *Journal of Marketing*, volume 66, pages 837–860, 2006.

[8] Andreas Eisingerich, Omar Merlo, Jan Heide, and Paul Tracey. Customer satisfaction and purchase behavior: The role of customer input. In *Looking Forward, Looking Back: Drawing on the Past to Shape the Future of Marketing*, pages 220–220, 2015.

[9] YEW-KWANGN NG. The economic theory of clubs: Pareto optimality conditions. In *Economica*, volume 40, pages 291–298, 1973.

附录 A　Matlab 源代码

1.1　ObjectFunction.m

```
%目标函数
%调用方法:
%load matlab.mat
%[x, p, count, object] = ObjectFunction(density,task_pos,task_credit, ...
    task2vip_min, 1/70, [-0.945,75.848]);
function [x, p, count, object, payment] = ...
    ObjectFunction(density,task_pos,task_credit, remote, coeff, solution)
task_num = size(task_pos, 1);
p = solution(1)*task_credit.*remote + solution(2); %定价矩阵
x = zeros(task_num, 1);
count = 0;
for i = 1:task_num
    if p(i) > -2.2740*density(i) + 75.2988  %酬金大于路程代价
        x(i) = 1;
        count = count+ 1;
    end
end
payment = sum(x.*p);        %计算商家需要支付的总酬金
object = sum(x) - sum(1.- x) - coeff*payment;  %计算目标函数值
```

1.2　ObjectFunction2.m

```
%目标函数（打包模式下）
%调用方法:
%load matlab.mat
%[x, p, count, object, payment] = ...
    ObjectFunction2(task_distance,package,density,task_credit, ...
    task2vip_min, 1/70,  [2.4103, 68.3345])
function [x, p, count, object, payment]  = ...
    ObjectFunction2(distance,package,density,task_credit, remote, ...
    coeff, solution)
task_num = size(remote, 1);
p = zeros(size(package,2), 1);
%计算打包结构体的定价
for i = 1:size(package,2)    %package 为打包的结构体序列
```

```
    if package(i).node_num == 1     %如果任务包为单点包，按照原方法定价
        num = package(i).rec(1);
        p(i) = solution(1)*task_credit(num)*remote(i) + solution(2);
    else                            %如果任务包为多点包，包内每个任务点的定价用密度计算
        p_group = zeros(package(i).node_num, 1);
        for j = 1:package(i).node_num
            num = package(i).rec(j);
            p_group(j) = solution(1)*task_credit(num)/(density(num) + 1) + solution(2);
        end
        p(i) = package(i).node_num * min(p_group); %任务包的定价为包内最低价格任务点的价格 n 倍
    end
end
%预测所有任务包的完成度
x = zeros(size(package,2), 1);
count = 0;
com_sum = 0;
for j = 1:size(package,2)
    if package(j).node_num == 1
        if p(j) >  -2.2740*density(j) + 75.2988
            x(j) = 1;
            count = count+ 1;
            com_sum = com_sum + 1;
        end
    else
        internDis = 0;  %记录任务包的内部距离和
        orgDis = 0;    %记录任务包的各点外部距离和
        for jj = 1:package(j).node_num - 1
            internDis = internDis + distance(package(j).rec(jj), package(j).rec(jj + 1));
            orgDis = orgDis + remote(package(j).rec(jj));
        end
        orgDis = orgDis + remote(package(j).rec(package(j).node_num));
        if p(j) > 2.0383*(orgDis/package(j).node_num + internDis) + 68.1819  %由第一问拟合而得的距离定价线
            x(j) = 1;
            count = count+ package(j).node_num;
        end
    end
end
payment = sum(x.*p);
```

```
object = count - (task_num - count) - coeff*payment;          %输出目标函数的最终值
```

1.3 test.m

```
%数据测试以及满意度求解
%调用方法:
%load matlab.mat
%[x, p, count, object, Sat1, x2, p2, count2, object2, Sat2, p3, ...
    object3, Sat3]= test(test_data, vip_pos_mod, vip_credit, isdone, ...
    task_pri, 0.6);
function [x, p, count, object, Sat1, x2, p2, count2, object2, Sat2, ...
    p3, object3, Sat3] = test(test_data, vip_pos, vip_credit, isdone, ...
    org_task_pri, m)
total = size(test_data,1);      %记录测试数据中任务点总数
task_vip_distance = getAllDist(vip_pos, test_data);    %计算任务点与会员点之间的距离
density = getDensity(task_vip_distance, 0.05);      %计算每个任务点处的会员密度
task_credit = task_credit2(task_vip_distance,vip_credit, 0.05);      %计算每个任务点所在位置的相对地理信誉值
[cluster_dis, minDis, test_idx, center] = vip_cluster(vip_pos, 4, ...
    test_data);       %对会员点分布聚类
[x, p, count, object, ¬] = ...
    ObjectFunction(density,test_data,task_credit, minDis, 1/70, [ ...
    -0.9566,75.8729]);       %计算不打包模式下的目标函数值、完成情况和定价。可选[-0.945,75.848]

%不打包模式的商家满意度
Sat11_org = sum(x)/sum(x.*p);  %商家初始满意度表示
disp('无打包模型下软件运营者满意度:')
Sat11 = atan(300*Sat11_org)*2/pi  %商家满意度标准化

%不打包模式的用户满意度
length = zeros(4, 1);      %记录每个聚类总任务点到中心的距离总和
count_clus = zeros(4, 1);      %记录每个聚类中任务数量
Sat12 = zeros(total, 1);    %记录每个任务能提供的满意度
for j = 1:size(vip_pos)
    count_clus(test_idx(j)) = count_clus(test_idx(j)) + 1;
    length(test_idx(j)) = length(test_idx(j)) + cluster_dis(j, ...
        test_idx(j));
end
mean_len = length./count_clus;       %每个聚类中用户前往任务的平均距离
```

```
len_cost = 2.0383*mean_len + 68.1819;       %路程代价
res = 0;
for i = 1:total
    Sat12(i) = atan(p(j)/len_cost(test_idx(j)))*2/pi;    %对每一个满意度归一化处理
    res = res + Sat12(i);
end
disp('无打包模型下软件用户满意度：')
Sat12 = res/total     %所有任务为用户提供的综合满意度

n = 1 - m;
disp('无打包模型综合满意度：')
Sat1 = Sat11^m * Sat12^n

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%打包模式的满意度计算
distance = task_distance(test_data);  %计算任务点之间的距离
sequence = link(test_data);           %将任务点链接成链
package = packnode(test_data, sequence, 0.02, 4);    %将任务链切断打包
pack_num = size(package,2);                %记录总任务包数
[x2, p2, count2, object2, ¬] = ...
    ObjectFunction2(distance,package,density,task_credit, minDis, 1/85, ...
    [2.41, 68.3345]);    %计算打包模式下的目标函数值

%打包模式的商家满意度
Sat21_org = sum(x2)/sum(x2.*p2);  %商家初始满意度表示
disp('打包模型下软件运营者满意度：')
Sat21 = atan(300 * Sat21_org)*2/pi  %商家满意度标准化

%打包模式下的用户满意度
pack_pos = zeros(pack_num, 2);    %记录任务包的位置坐标点
all_center_distance = zeros(pack_num, 4);    %记录每个任务包到每个会员聚类中心的距离
pack_minDis = zeros(pack_num, 2);          %记录每个任务包到最近会员中心的距离
intern_dis = zeros(pack_num, 1);  %记录每个包的内部距离
length2 = zeros(4, 1);          %记录每个聚类中所有任务到聚类中心的总路程
count_clus2 = zeros(4, 1);  %记录每个聚类中的任务包总数
Sat22 = zeros(pack_num, 1);    %记录每个任务对应的满意度
for i = 1:pack_num     %计算上述各记录矩阵的值
    if package(i).node_num == 1
        pack_pos(i,:) = test_data(package(i).rec(1), :);
    else
        xx = 0;
```

```
66          yy = 0;
67          for j = 1:package(i).node_num
68              xx = xx + test_data(package(i).rec(j), 1);
69              yy = yy + test_data(package(i).rec(j), 2);
70              if j ~= package(i).node_num
71                  intern_dis(i) = intern_dis(i) + ...
                        distance(package(i).rec(j), package(i).rec(j + 1));
72              end
73          end
74          %任务包位置按照包中任务的平均坐标值计算
75          pack_pos(i:1) = xx/package(i).node_num;
76          pack_pos(i:2) = yy/package(i).node_num;
77      end
78      for k = 1:4
79          all_center_distance(i,k) = ((center(k,1) - pack_pos(i,1))^2 + ...
              (center(k,2) - pack_pos(i,2))^2)^(1/2);    %计算每一个任务包到每个
              聚类中心的距离
80      end
81      [pack_minDis(i,1), pack_minDis(i,2)] = min(all_center_distance(i, ...
          :));    %记录每个任务包到最近聚类中心的距离和所属分
          类
82      length2(pack_minDis(i,2)) = length2(pack_minDis(i,2)) + ...
          pack_minDis(i,1);    %记录每个聚类中用户前往任务包的距离
          和
83      count_clus2(pack_minDis(i,2)) = count_clus2(pack_minDis(i,2)) + 1; ...
              %计算每个聚类中任务包的个
          数
84  end
85  mean_len2 = length2./count_clus2;                %计算用户前往任务包的平均距离
86  len_cost2 = 2.0383*mean_len2 + 68.1819;          %计算路程代价
87  res2 = 0;
88  for i = 1:pack_num
89      Sat22(i) = atan(p2(i)/len_cost2(pack_minDis(i,2)))*2/pi;       %满意度归
          一化
90      res2 = res2 + Sat22(i);
91  end
92  disp('打包模型下软件用户满意度：')
93  Sat22 = res2/pack_num            %所有任务包为用户提供的综合满意度
94  %Sat22 = res2/pack_num
95  disp('打包模型综合满意度：')
96  Sat2 = Sat21^m * Sat22^n
97
98  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
99  %题目中原本的定价策略
100 %默认模式商家满意度
```

```
Sat31_org = sum(isdone)/sum(isdone.*org_task_pri);
disp('题目原定价方案下的运营者满意度：')
Sat31 = atan(300 * Sat31_org)*2/pi    %商家满意度标准化

%默认模式用户满意度
Sat32 = zeros(total, 1);    %记录每个任务能提供的满意度
info = zeros(total, 5);
info(:,1) = test_data(:,1);
info(:,2) = test_data(:,2);
info(:,3) = density;
info(:,4) = minDis;
info(:,5) = task_credit;
weight =  [1.4768,1.4878,-2.2277,1.4454,-0.0985];
p3 = -128.0681 +info* weight';
object3 = Sat31_org * total - (total - Sat31_org) + ...
    1/70*Sat31_org*sum(p3);
res3 = 0;
for i = 1:total
    Sat32(i) = atan(p3(j)/len_cost(test_idx(j)))*2/pi;    %对每一个满意度归一化处理
    res3 = res3 + Sat32(i);
end
disp('题目原定价方案下的用户满意度：')
Sat32 = res3/total     %所有任务为用户提供的综合满意度

n = 1 - m;
disp('题目原定价方案综合满意度：')
Sat3 = Sat31^m * Sat32^n
```

1.4 SA.m

```
%不打包定价的目标优化模型 (基于模拟退火法)
%调用方法：
%load matlab.mat;
%[p, alpha, beta] = SA(density, task_pos,task_credit, task2vip_min, ...
    1/70, 0.05, -1,80)
function [p, alpha, beta] = SA(density, task_pos,task_credit, remote, ...
    coeff, StepFactor, start1, start2);
MarkovLength=1000; % 马可夫链长度
DecayScale=0.95; % 衰减参数
Temperature0=100; % 初始温度
```

```
Temperatureend=1; % 最终温度
Boltzmann_con=1; % Boltzmann 常数
AcceptPoints=0.0; % Metropolis 过程中总接受点
% 随机初始化参数
Par_cur=[start1, start2]; % 用 Par_cur 表示当前解
%Par_best_cur = Par_cur; % 用 Par_best_cur 表示当前最优解
Par_best=Par_cur; % 用 Par_best 表示冷却中的最好解
Par_new = Par_cur;
% 每迭代一次退火 (降温) 一次，直到满足迭代条件为止
t=Temperature0;
alpha = start1;
beta = start2;
itr_num=0; % 记录迭代次数
while t>Temperatureend
    itr_num=itr_num+1;
    t = DecayScale*t;  % 温度更新（降温）
    for i=1:MarkovLength
        % 在此当前参数点附近随机选下一点
        Par_new(1) = Par_cur(1) + StepFactor*(rand(1) - 0.5);
        Par_new(2) = Par_cur(2) + StepFactor*(rand(1) - 0.5);
        % 检验当前解是否为全局最优解
        if (ObjectFunction(density,task_pos,task_credit, remote, ...
            coeff,Par_best)<...
                ObjectFunction(density,task_pos,task_credit, remote, ...
                    coeff,Par_new))
            % 保留上一个最优解
            %Par_best_cur = Par_best;
            % 此为新的最优解
            Par_best=Par_new;
        end
        % Metropolis 过程
        if (ObjectFunction(density,task_pos,task_credit, remote, ...
            coeff,Par_cur)-...
                ObjectFunction(density,task_pos,task_credit, remote, ...
                    coeff,Par_new)<0)
            % 接受新解
            Par_cur=Par_new;
            AcceptPoints=AcceptPoints+1;
        else
            changer=(ObjectFunction(density,task_pos,task_credit, ...
                remote, coeff, Par_new)-...
            ObjectFunction(density,task_pos,task_credit, remote, ...
                coeff, Par_cur))/(Boltzmann_con*t);
```

```
                p1 = exp(changer);
                if p1>rand(1)
                    Par_cur=Par_new;
                    AcceptPoints=AcceptPoints+1;
                end
            end
            alpha = Par_best(1);
            beta = Par_best(2);
        end
    end
    [~, p, count, object, ~] = ...
        ObjectFunction(density,task_pos,task_credit, remote, 1/70, Par_best);
    count     %输出完成任务数
    object    %输出目标函数值
    alpha
    beta
```

1.5 SA_for_package.m

```
%打包模式下的目标优化模型（使用退火算法）
%调用方法:
%load matlab.mat;
% [p, alpha, beta] = SA_for_package(task_distance, package, ...
    density,task_credit, task2vip_min, 1/70, 0.5, 1,80)
function [p, alpha, beta] = SA_for_package(distance, package, ...
    density,task_credit, remote, coeff, StepFactor, start1, start2)
MarkovLength=100; % 马可夫链长度
DecayScale=0.95; % 衰减参数
Temperature0=100; % 初始温度
Temperatureend=1; % 最终温度
Boltzmann_con=1; % Boltzmann 常数
AcceptPoints=0.0; % Metropolis 过程中总接受点
% 随机初始化参数
Par_cur=[start1, start2]; % 用 Par_cur 表示当前解
%Par_best_cur = Par_cur; % 用 Par_best_cur 表示当前最优解
Par_best=Par_cur; % 用 Par_best 表示冷却中的最好解
Par_new = Par_cur;
% 每迭代一次退火 (降温) 一次，直到满足迭代条件为止
t=Temperature0;
alpha = start1;
beta = start2;
```

```
itr_num=0; % 记录迭代次数
while t>Temperatureend
    itr_num=itr_num+1;
    t = DecayScale*t;  % 温度更新（降温）
    for i=1:MarkovLength
        % 在此当前参数点附近随机选下一点
        Par_new(1) = Par_cur(1) + StepFactor*(rand(1) - 0.5);
        Par_new(2) = Par_cur(2) + StepFactor*(rand(1) - 0.5);
        % 检验当前解是否为全局最优解
        if (ObjectFunction(distance,package, density,task_credit, ...
            remote, coeff,Par_best)<...
                ObjectFunction(distance,package,density,task_credit, ...
                    remote, coeff,Par_new))
            % 此为新的最优解
            Par_best=Par_new;
        end
        % Metropolis 过程
        if (ObjectFunction(distance,package,density,task_credit, ...
            remote, coeff,Par_cur)<...
                ObjectFunction(distance,package,density,task_credit, ...
                    remote, coeff,Par_new))
            % 接受新解
            Par_cur=Par_new;
            AcceptPoints=AcceptPoints+1;
        else
            changer=(ObjectFunction(distance,package,density,task_credit, ...
                remote, coeff, Par_new)-...
            ObjectFunction(distance,package,density,task_credit, ...
                remote, coeff, Par_cur))/(Boltzmann_con*t);
            p1 = exp(changer);        %接受概率
            if p1>rand(1)
                Par_cur=Par_new;
                AcceptPoints=AcceptPoints+1;
            end
        end
        alpha = Par_best(1); %更新当前最优值
        beta = Par_best(2);
    end
end
%计算最优解下的目标函数，以及定价向量
[¬, p, ¬, ¬, ¬]  = ...
    ObjectFunction2(distance,package,density,task_credit, remote, ...
    coeff, Par_best);
```

1.6 vip_cluster.m

```
%会员聚类，并计算每个任务点到各个聚类中心的距离，以及最短距离
%调用方法：
%load ; matlab.mat
%[task2vip, task2vip_min, vip_idx, vip_C] = vip_cluster(vip_pos, k, ...
    task_pos);
function [distance, minDis, idx, C] = vip_cluster(vip_pos, k, task_pos)
[idx,C, ~, ~] = kmeans(vip_pos,k);
task_num = size(task_pos, 1);
vip_num = size(vip_pos, 1);
figure
for ii = 1:vip_num
    scatter(vip_pos(ii,2), vip_pos(ii,1), 5, idx(ii)*3);
    hold on
end
distance = zeros(task_num, k);
for i = 1:task_num
    for j = 1:k
        distance(i, j) = ((task_pos(i,1) - C(j,1))^2 + (task_pos(i,2) ...
            - C(j,2))^2)^(1/2);
    end
end
minDis = zeros(task_num, 1);
for i = 1:task_num
    minDis(i) = min(distance(i));
end
```

1.7 link.m

```
%任务成链
%调用方法：
%load matlab.mat
% sequence = link(task_pos)
function sequence = link(task_pos)
task_num = size(task_pos,1);
distance = zeros(task_num, task_num);
%计算任务点两两距离
for i = 1:task_num
    for j = 1:task_num
```

```
        distance(i, j) = ((task_pos(i,1) - task_pos(j,1))^2 + ...
            (task_pos(i,2) - task_pos(j,2))^2)^(1/2);
    end
end
count_all = 1;
sequence = zeros(task_num, 1);
flag = zeros(task_num, 1);
i = 1;
sequence(count_all) = i;
flag(1) = 1;
while count_all < task_num
    %贪心法成链
    [¬, minpos] = min(distance(i,:));
    if flag(minpos) == 0
        count_all = count_all + 1;
        sequence(count_all) = minpos;
        i = minpos;
        flag(minpos) = 1;
        distance(i,minpos) = inf;
    else
        distance(i,minpos) = inf;
    end
end
```

1.8 packnode.m

```
%断链打包
%调用方法:
%load matlab.mat;
%package = packnode(task_pos, sequence, intervalMost, linkmost)
function package = packnode(task_pos, sequence, intervalMost, linkmost)
task_num = size(task_pos, 1);
distance = zeros(task_num, task_num);
%计算任务点两两距离
for i = 1:task_num
    for j = 1:task_num
        distance(i, j) = ((task_pos(i,1) - task_pos(j,1))^2 + ...
            (task_pos(i,2) - task_pos(j,2))^2)^(1/2);
    end
end
%打包
```

```
pack_number = 0;
count = 1;
pack_node = 1;
record = zeros(task_num, 1);
for i = 1:task_num-1
    %点距超过阈值就切断
    if distance(sequence(i), sequence(i+1)) > intervalMost
        record(i) = count;
        count = count + 1;
        pack_node = 1;
        pack_number = pack_number + 1;
        if i == task_num - 1
            record(i + 1) = count;
            pack_number = pack_number + 1;
        end
    %点距在阈值之内考虑是否划入相同包
    else
        %当前链节点数小于上限，划入同一子链
        if pack_node < linkmost
            record(i) = count;
            pack_node = pack_node + 1;
            if i == task_num - 1
                record(i + 1) = count;
            end
        %否则切断
        else
            record(i) = count;
            count = count + 1;
            %pack_node = 1;
            pack_number = pack_number + 1;
            if i == task_num - 1
                record(i + 1) = count;
                pack_number = pack_number + 1;
            end
        end
    end
end
%建立任务包的结构体序列
for i =  1:pack_number
    package(i).node_num = 0;
    package(i).first = 0;
end
%记录每个任务包的首个任务序号
```

```
for i = 1:task_num
    package(record(i)).node_num = package(record(i)).node_num + 1;
    if package(record(i)).first == 0
        package(record(i)).first = i;
    end
end
%记录每个任务包所有任务序号
for i = 1: pack_number
    num = package(i).node_num;
    package(i).rec = zeros(num,1);
    for j = 1:num
        package(i).rec(j) = sequence(package(record(i)).first + j - 1);
    end
end
```

1.9 analysis.m

```
%分析商家定价为什么造成许多任务未能完成
%经度维度会员密度相对地区信誉度任务点到聚类中心最短距离
%调用方式:
%load matlab.mat
%analysis(task_pos, task_pri,task_credit, density, task2vip_min, isdone)
function analysis(task_pos, task_pri,task_credit, vip_density, minDis, ...
    isdone)
%整理自变量和因变量
analysis1 = zeros(size(task_pos,1), 5);
analysis1(:,1) = task_pos(:,1);
analysis1(:,2) = task_pos(:,2);
analysis1(:,3) = vip_density;
analysis1(:,4) = task_credit;
analysis1(:,5) = minDis;
[b,stats] = robustfit(analysis1,task_pri)

valid_num = sum(isdone,1);
analysis2 = zeros(valid_num, 5);
valid_pri = zeros(valid_num, 1);
count = 1;
for i = 1:size(task_pos,1)
    if isdone(i) == 1
        analysis2(count,1) = task_pos(i,1);
        analysis2(count,2) = task_pos(i,2);
```

```
24          analysis2(count,3) = vip_density(i);
25          analysis2(count,4) = task_credit(i);
26          analysis2(count,5) = minDis(i);
27          valid_pri(count) = task_pri(i);
28          count = count + 1;
29      end
30  end
31  [b2,stats2] = robustfit(analysis2,valid_pri)
```

1.10 getAllDist.m

```
1  %获得任务点和会员两两之间的距离矩阵
2  function distance = getAllDist(vip_pos, task_pos)
3  vip_num = size(vip_pos,1);
4  task_num = size(task_pos,1);
5  distance = zeros(task_num, vip_num);
6  for i = 1:task_num
7      for j = 1:vip_num
8          distance(i, j) = ((task_pos(i,1) - vip_pos(j,1))^2 + ...
               (task_pos(i,2) - vip_pos(j,2))^2)^(1/2);
9      end
10 end
```

1.11 getDensity.m

```
1  %求指定任务点的会员密度
2  function gd = getDensity(distance, disThresh)
3  tmp = distance;
4  task_num = size(distance,1);
5  vip_num = size(distance,2);
6  gd = zeros(task_num, 1);
7  for i = 1:task_num
8      count = 0;
9      for j = 1:vip_num
10         if tmp(i, j) ≤ disThresh  %如果距离小于阈值，纳入计数范围
11             count = count + 1;
12         end
13     end
```

```
14      if count == 0
15          gd(i,1) = 0;
16      else
17          gd(i,1) = log(count);  %取对数处理
18      end
19  end
```

1.12 plot_done.m

```
1  %绘制已完成任务的距离-定价曲线
2  %调用方法:
3  %load matlab.mat
4  %plot_done(isdone, task2vip_min, task_pri)
5  function plot_done(isdone, mindis, task_pri)
6  figure
7  task_num = size(isdone, 1);
8  for i = 1:task_num
9      if isdone(i)== 1
10         scatter(mindis(i), task_pri(i), 5, 3)
11         hold on
12     end
13 end
```

1.13 plot_done2.m

```
1  %绘制全部任务任务的会员密度-定价曲线
2  function plot_done2(isdone, density, task_pri)
3  figure
4  task_num = size(isdone, 1);
5  for i = 1:task_num
6      scatter(density(i), task_pri(i), 5, 3)
7      hold on
8  end
9  b = robustfit(density, task_pri)
10 x = linspace(min(density), max(density), 1000);
11 y = b(2)*x + b(1);
12 plot(x, y);
```

1.14 plot_done3.m

```
%绘制已完成任务的会员密度-定价曲线
%调用方法:
%load matlab.mat
%plot_done3(isdone, density, task_pri)
function plot_done3(isdone, density, task_pri)
figure
task_num = size(isdone, 1);
valid_num = sum(isdone,1);
new_density = zeros(valid_num,1);
new_taskpri = zeros(valid_num,1);
count = 1;
for i = 1:task_num
    if isdone(i) == 1
        scatter(density(i), task_pri(i), 5, 3)
        hold on
        new_density(count) = density(i);
        new_taskpri(count) = task_pri(i);
        count = count + 1;
    end
end
b = robustfit(new_density, new_taskpri)
x = linspace(min(new_density), max(new_density), 1000);
y = b(2)*x + b(1);
plot(x, y);
```

1.15 plot_done4.m

```
%绘制任务与聚类中心距离-任务定价曲线
%调用方法:
%load matlab.mat
%plot_done4(isdone, task2vip_min, task_pri)
function plot_done4(isdone, minDis, task_pri)
figure
task_num = size(isdone, 1);
valid_num = sum(isdone,1);
new_minDis = zeros(valid_num,1);
new_taskpri = zeros(valid_num,1);
count = 1;
```

```
for i = 1:task_num
    if isdone(i) == 1
        scatter(minDis(i), task_pri(i), 5, 3)
        hold on
        new_minDis(count) = minDis(i);
        new_taskpri(count) = task_pri(i);
        count = count + 1;
    end
end
b = robustfit(new_minDis, new_taskpri)
x = linspace(min(new_minDis), max(new_minDis), 1000);
y = b(2)*x + b(1);
plot(x, y);
```

1.16 record_pack.m

```
%用结构体 record 记录每个任务包中的任务序号
%调用方法:
%load matlab.mat
%record = record_pack(package, 4)
function record = record_pack(package, num)
for i = 1:num
    ss = '';
    for j = 1:package(i).node_num
        if package(i).rec(j) ~= 0
            ss = strcat(ss ,num2str(package(i).rec(j)));
            if j < package(i).node_num
                ss = strcat(ss ,', ');
            end
        end
    end
    record(i).content = ss;
end
```

1.17 task_credit1.m

```
%根据会员分布得到指定地点的相对信誉值 (模型最终采用方法为 task_credit2)
function tc = task_credit1(distance,vip_credit, localVip)
```

```
tmp = distance;
task_num = size(distance,1);
tc = zeros(task_num, 1);
for i = 1:task_num
    sum = 0;
    for j = 1:localVip
        [¬,mpos] = min(tmp(i,:));
        sum = sum + log(vip_credit(mpos));
        tmp(i, mpos) = max(tmp(i,:));
    end
    tc(i) = sum/localVip;
end
```

1.18 task_credit2.m

```
%根据会员分布得到指定地点的相对信誉值
function tc = task_credit2(distance,vip_credit, disThresh)
tmp = distance;
task_num = size(distance,1);
vip_num = size(distance,2);
tc = zeros(task_num, 1);
for i = 1:task_num
    count = 0;
    sum = 0;
    for j = 1:vip_num
        if tmp(i, j) ≤ disThresh  %任务与会员距离在阈值内，则加入计数
            count = count + 1;
            sum = sum + log(vip_credit(j));
        end
    end
    if count > 0
        tc(i) = sum/count;
    else
        tc(i) = 0;
    end
end
tc_min = min(tc);
for i = 1:task_num
    if tc(i) ≠ 0
        tc(i) = tc(i) - tc_min;
    end
```

```
27 end
```

1.19 task_distance.m

```
1  %计算所有任务点两两之间的距离
2  %调用方法:
3  %task_distance = task_distance(task_pos);
4  function td = task_distance(task_pos)
5  task_num = size(task_pos,1);
6  td = zeros(task_num, task_num);
7  for i = 1:task_num
8      for j = 1:task_num
9          td(i, j) = ((task_pos(i,1) - task_pos(j,1))^2 + (task_pos(i,2) ...
               - task_pos(j,2))^2)^(1/2);
10     end
11 end
```