# 外卖路线规划和订单分配问题的优化分析

# 外卖路线规划和订单分配问题的优化分析

## 1、摘要

互联网迅速发展，餐饮业与互联网结合，产生了外卖派送员这一行业。想要高效的将外卖送到消费者手中，就需要对外卖派送员的派送路径和派送流程进行优化调整，从而提高派送的效率。

为了更好的解决外卖配送问题，我们首先根据给定的某时段订单分布图，利用 K-means 聚类，将临近的订单聚集在一起。然后利用四叉树模型对区域进行划分，接着利用蚁群算法解决了外卖骑手的路线规划问题。

接着我们选取了上述分区中的其中一个，调查了一个外卖店家在该区域的一个月的日订单数量变化，通过机器学习分析区域内的订单数量变化情况，并选取一个时段的外卖需求量进行分析。

然后通过层级分析法计算出合理的优化指标，再利用 0-1 规划模型和网络流模型求解当前时段订单分配问题，并且利用强化学习解决序列决策问题，找出最优的订单分配方案。最后对不同的算法效率进行对比。

**关键词：**外卖配送；路径规划；订单分配；0-1 规划；网络流；机器学习；强化学习

## Abstract

The rapid development of the Internet and the combination of the catering industry and the Internet have resulted in the industry of takeaway delivery personnel. To efficiently deliver takeaways to consumers, it is necessary to optimize the adjustment of delivery routes and delivery processes of outbound delivery personnel in order to increase the efficiency of delivery.

In order to better solve the problem of take-away delivery, we first use the K-means clustering to aggregate the adjacent orders based on the order distribution chart for a given time period. Then the quadtree model is used to divide the area, and then the ant colony algorithm is used to solve the problem of route planning for the rider.

Then we selected one of the above sub-districts, investigated the change in order quantity per month for a take-away store in the region, analyzed the change in order quantity in the region through machine learning, and selected a period of take-out demand for analysis. .

Then, a reasonable optimization index is calculated by the hierarchical analysis method. Then the 0-1 planning model and the network flow model are used to solve the order allocation problem in the current period, and reinforcement learning is used to solve the sequence decision problem to find the optimal order allocation plan. Finally, compare the efficiency of different algorithms.

**Keywords:** Takeaway Delivery; Path Planning; Order Allocation; 0-1 Planning; Network Flow; Machine Learning; Reinforcement Learning

# 2、问题重述

## 2.1 问题背景

互联网迅速发展，餐饮业与互联网结合，产生了外卖派送员这一行业。最近两年，外卖的市场规模持续以超常速度发展。近期美团外卖订单量峰值达到 1600 万，是全球规模最大的外卖平台。

目前各外卖平台正在优质供给、配送体验、软件体验等各维度展开全方位的竞争，其中，配送时效、准时率作为履约环节的重要指标，是外卖平台的核心竞争力之一。而想要高效的将外卖送到消费者手中，就需要对外卖派送员的派送路径和派送流程进行优化调整，从而提高派送的效率。

## 2.2 问题重述

假设每个外卖员途中平均速度为 25km/h，每个客户的需求必须满足，且只能由一个人送货。每辆车所走的路线不能重复，每条送外卖的路径上各个客户的需求量之和不超过个人最大负重即 6 份。

需解决如下问题：

（1）给出一张某一时刻在某一范围的订单分布图，对该图进行合理的分区

（2）针对问题(1)给出的分区，选择一个区域设计骑手行进路线；

（3）自行调查数据商家中午 11:00-13:00 的数据，并分成 12 个时段，预测中午某个时间段的订单数量；

（4）考虑前述路径优化问题请设计一个合理的 cost 函数，来衡量某位骑手对于某份订单的适配度；

（5）选定(1)中设计的某个分区，按照(3)(4)当中得出的订单数量和 cost 指标，如何把这部分订单分配给不同的骑手。

（6）考虑序列决策问题，针对未来可能出现的订单，对订单分配问题建模，在(5)的条件下，如何对这部分订单进行合理地分配。


# 3、模型建立与求解

## 3.1 路径问题分析与建模

这个部分我们着力于解决 2.2 中的前两个问题（1）给出一张某一时刻在某一范围的订

单分布图，对该图进行合理的分区；（2）针对问题(1)给出的分区，选择一个区域设计骑手行进路线。

### 3.1.1 聚类分析

因为外卖订单具有聚集性，为了简化模型，可以将小范围内例如（宿舍楼，小区等）的外卖都集中归结到一个点上， 这里采用聚类分析的方法。聚类分析是一种定量方法，从数据分析的角度看，它是对多个样本进行定量分析的多元统计分析方法。

划分聚类的代表是 K-Means 算法，它需要在一开始指定类目数，根据距离最近的原则，把待分类的样本点划分到不同的族，然后按照平均法计算各个族的质心，重新分配质心，直到质心的移动距离小于某个值。K-Means 算法也称 K-均值聚类算法，是一种广泛使用的聚类算法，也是其他聚类算法的基础。

假定输入样本为 S = X1，X2，…，Xm，则算法步骤为：

1. 选择初始的 k 个类别中心 μ1，μ2，…，μk；
2. 对于每个样本 Xi，将其标记为距离类别中心最近的类别（距离计算采用欧式距离）；
3. 将每个类别中心更新为隶属该类别的所有样本的均值；
4. 重复最后两步，直到类别中心的变化小于某阈值。

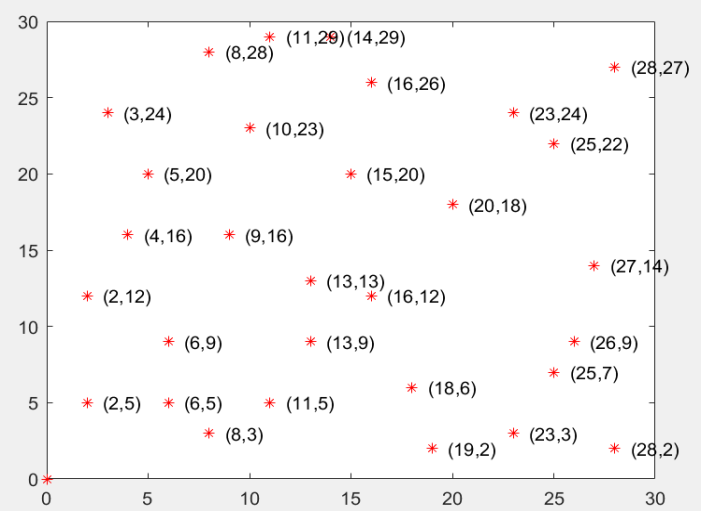
终止条件一般有迭代次数，族中心变化率，最小平方误差 MSE（Minimum Squared Error）


聚类后的订单分布

上图是玉泉中午某一时刻 2 分钟内的外卖订单需求分布，因为点比较密集，我们根据上

述的聚类原则，把邻近的需求点聚合成一个点。

| 送货点 | 外卖量 T (份数) | 坐标 (*0.1km) | | 送货点 | 外卖量 T (份数) | 坐标 (*0.1km) | |
|---|---|---|---|---|---|---|---|
| | | x | y | | | x | y |
| 1 | 2 | 2 | 5 | 16 | 1 | 18 | 6 |
| 2 | 2 | 8 | 28 | 17 | 1 | 15 | 20 |
| 3 | 1 | 3 | 24 | 18 | 2 | 25 | 22 |
| 4 | 2 | 11 | 29 | 19 | 2 | 13 | 9 |
| 5 | 1 | 2 | 12 | 20 | 1 | 23 | 3 |
| 6 | 1 | 3 | 11 | 21 | 2 | 20 | 18 |
| 7 | 2 | 11 | 5 | 22 | 2 | 5 | 20 |
| 8 | 1 | 27 | 14 | 23 | 1 | 27 | 9 |
| 9 | 1 | 9 | 16 | 24 | 2 | 15 | 19 |
| 10 | 1 | 10 | 23 | 25 | 2 | 8 | 3 |
| 11 | 1 | 4 | 16 | 26 | 2 | 28 | 27 |
| 12 | 3 | 6 | 5 | 27 | 3 | 25 | 7 |
| 13 | 1 | 16 | 12 | 28 | 1 | 26 | 9 |
| 14 | 1 | 28 | 2 | 29 | 2 | 19 | 2 |
| 15 | 3 | 6 | 9 | 30 | 1 | 13 | 13 |




订单具体坐标图

根据坐标化的地图，根据四叉树法进行分割，建立骑手的外卖配送模型，用 lingo 筛选出最佳路径，最后制定出基站的最佳策略。

在基本问题的假设中，我们假定所有骑手送一次外卖就能满足区块内的所有需求，因此骑手每次带的外卖不超过个人一次带货的份数。

在问题中，我们要解决的问题是骑手的最少人数，每个骑手的路径规划和总的行驶路程问题。追求这些目标的最大优化。事实上，当骑手人数和每个骑手的路径确定以后，总的行驶路径也能确定。

### 3.1.2 基于四叉树的区域划分

四叉树索引的基本思想是将地理空间递归划分为不同层次的树结构它将已知范围的空间等分成四个相等的子空间，如此递归下去，直至树的层次达到一定深度或者满足某种要求后停止分割。这种思想在空间结构的描述方面有很广泛的应用，而且外卖配送作为空间点线关系的处理，采用四叉树方法进行处理有简洁快速的技术优势。

通常四叉树与相邻 8 个网格连接，即有 8 个方向度，如果邻近网格没有需求点，则该四叉树周围无四叉树网格连接，即该网格的方向度为 0，形成了孤岛。

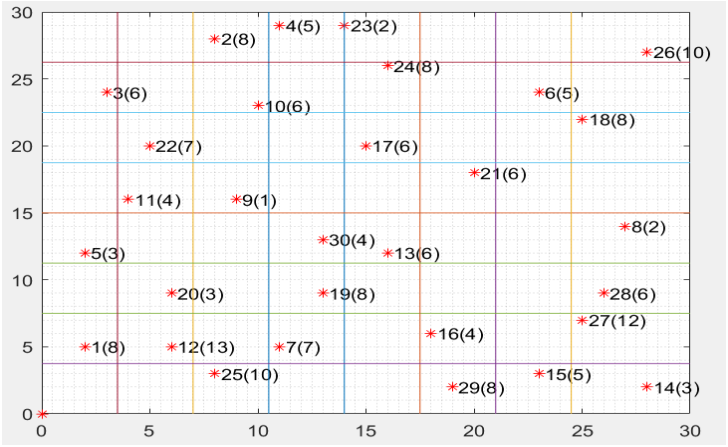
以四叉树中网格包含的需求点外卖需求量小于等于骑手载货量限制为基准进行四叉树的生成，当四叉树网格包含的需求点外卖总需求量小于等于骑手载货量限制时，停止该四叉树网格的划分，则该四叉树网格为一个候选配送分区，剔除不包含需求点的四叉树网格。

确定候选区间。选择距离供给点最远端的四叉树网格，延逆时针方向进行网格中需求点的扩充。如果与之临近网格中有方向度小于 3 的网格，则以临近网格中最小方向度的网格为中心进行补充，原网格作为次选网格。扩充的条件是：如果本网格的外卖总需求量等于骑手载货量限制，则不进行扩充；如果本网格的外卖总需求量小于骑手的载货量限制，则延逆时针方向以最近原则选择相邻一定等级的网格内的需求点（实际上，应当根据配送中心的实际用户分布的聚类特征决定，为了节省计算和存储资源，考虑到外卖配送需求点基本呈均匀分布的特点，可以认为客户呈均匀离散分布。）使得配送区位的总需求量等于或者接近骑手载货量限制。所选择的需求点从原从属网格中剔除；将无需求点的网格剔除，形成非四叉树形式的配送区位。

重复上述操作，使每个需求点仅一次被选择入一个配送区位。

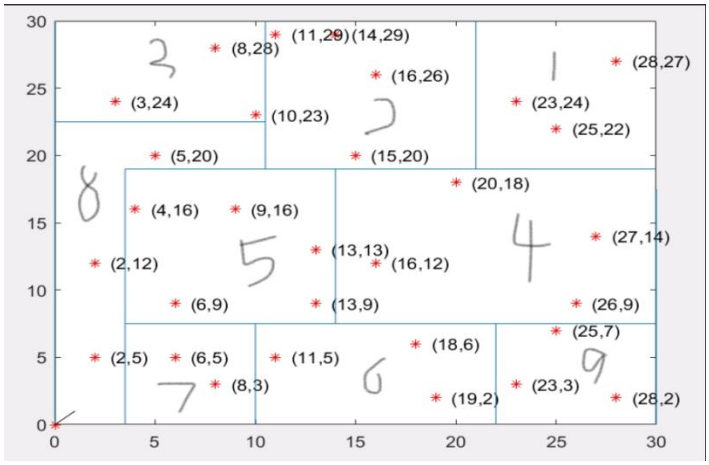
在实际的区域中，送货点被分成大小不等的区域，每个基站负责一个区域。在这个问题

中，我们先对地图进行四等分，然后对每个部分再四等分，使得每个区域总的送货量不断减小，直到每个区域中最多只有一个送货点，等分完的结果如下图。




四叉树三级网格图

然后从离原点最远端开始合并单元格，对没有送货点的区域直接合并，而对于有送货点的区域，按照逆时针方向合并，直到合并的总区域里所有的送货点的外卖单数接近但没有超过 6。最后每个送货点都被单独的分入一个区域，划分的地图一共 9 个区域，如下。这样做是把分散的送货点按照单数合并起来，便于接下来的模型求解。




四叉树配送分配区域图

### 3.1.3 拥挤度

随着汽车保有量的持续增长，交通拥堵现象日益频繁，为了解决交通拥堵问题，城市管理者采用了各种渠道缓解交通拥堵问题。其中有获取市内各地区道路实时交通情况，并在公共平台公布，供驾驶员参考，使得一定程度上缓解拥堵情况的方法。

交通拥挤度由速度，密度，流量等多个标准衡量，反映的是道路的畅通程度。我们假定各个道路的道路等级基本相同，因此简单以道路速度评价道路的拥挤度。道路拥挤可以拓展，

分为周期性道路拥挤和非周期性道路拥挤，周期性的比如上下班高峰时间，非周期性的比如天气，道路突发状况等，实际上可以拓展为宽泛的概念。我们在这里只考虑周期性的道路拥挤，以某一时段的拥挤度为例加入之前的 TSP 模型中重新求解。

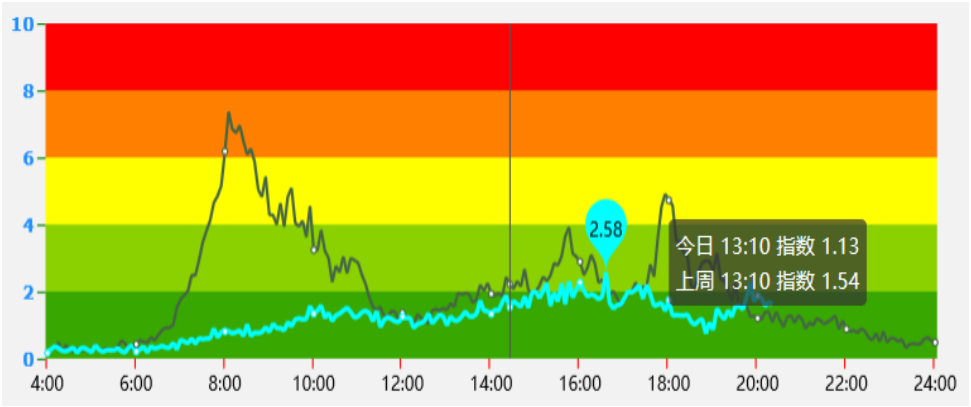
各城市对拥挤度的定义不同，现在采用较多的是北京的《城市道路交通运行评价指标体系（DB11/T 785—2011）》，其中把拥堵指数分为五个区间，数值越大拥挤程度越高。

| 道路交通运行指数 | [0,2] | [2,4] | [4,6] | [6,8] | [8,10] |
|---|---|---|---|---|---|
| 道路网拥堵等级 | 非常畅通 | 畅通 | 轻度拥堵 | 中度拥堵 | 重度拥堵 |

各指数的含义如下：

| 交通指数 | 对应路况 | 出行时间 |
|---|---|---|
| 0-2 | 基本没有道路拥堵 | 可以按标准速度行驶 |
| 2-4 | 有少量道路拥堵 | 比畅通时多耗时 0.4 倍 |
| 4-6 | 部分主干路拥堵 | 比畅通时多耗时 0.6 倍 |
| 6-8 | 大量主干度拥堵 | 被畅通时多耗时 1 倍 |
| 8-10 | 大部分道路拥堵 | 比畅通时多耗时 1.2 倍以上 |

这是杭州某天下午的拥挤度情况




我们对各个送货点设定了拥挤度，代表了通往这个送货点的道路的通畅程度。给图的每个节点都加上一个权值代表拥挤度（具体算法实施过程中，通往某个点的道路都加上了当前点的拥挤度），然后重新计算距离。对前面的四叉树法进行分区求解，每个分区内仍按照 TSP 问题以组合方法进行求解，得出每个区域的送货时间。

30 个点的拥挤度如下：

| 序号 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 拥挤度 | 2.8 | 2.8 | 8 | 8 | 8 | 4.8 | 1.2 | 2.8 | 1.6 | 9.6 |

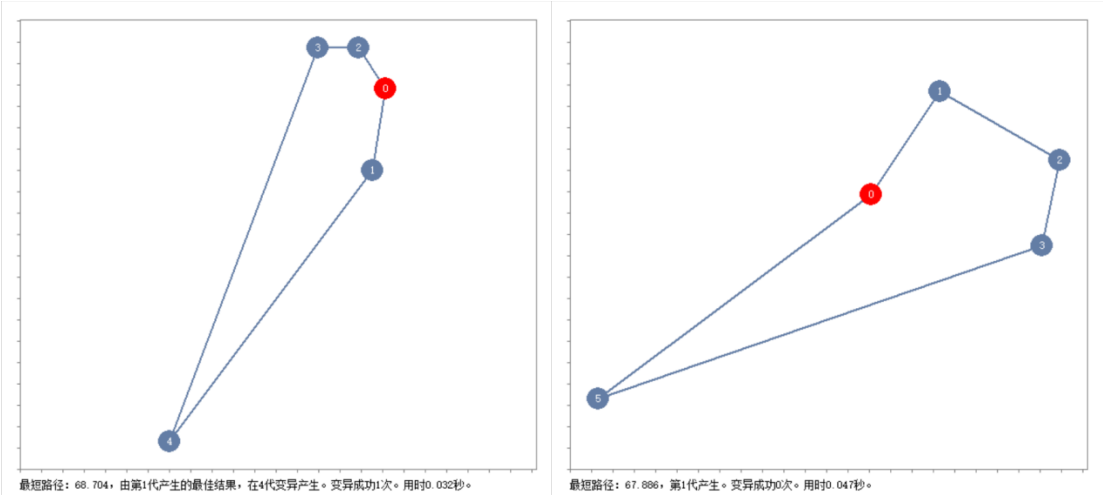
| 序号 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|
| 拥挤度 | 6.4 | 2.1 | 9.6 | 4.8 | 4.8 | 6.4 | 9.0 | 2.8 | 2.8 | 7.5 |
| 序号 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
| 拥挤度 | 9.6 | 1.2 | 2.6 | 2.8 | 6.0 | 6.0 | 1.6 | 9.6 | 1.1 | 6.4 |

### 3.1.4 路径问题的求解

根据每位骑手的载单量对地图进行重组与规划，规划出 8 个区域，然后采用蚁群算法确定每个区域的最佳路径，即总路程，根据速度算出时间，情况如下。

| 区号 | 各区路线 | 此分区送货时间 |
|---|---|---|
| 1 | 0-18-26-6-0 | 27.5 min |
| 2 | 0-17-24-23-4-0 | 22.5 min |
| 3 | 0-2-3-0 | 18 min |
| 4 | 0-28-8-21-13-0 | 22.5 min |
| 5 | 0-11-9-30-19-20-0 | 14.5 min |
| 6 | 0-7-16-29-0 | 18 min |
| 7 | 0-12-25-0 | 6.5 min |
| 8 | 0-5-22-1-0 | 12.5 min |
| 9 | 0-15-27-14 | 14.5 min |
| 总路程(KM) | 56.7 | |

下图是 2 区（左）和 4 区（右）的求解结果：



## 3.2 现有订单分配问题的分析与建模

实际外卖配送问题中，平台收到的订单数量往往是非常巨大的，而且平台在分配时往往会给同一个骑手分配同一个区域的订单，故在此我们也只考虑某一个区域在某一时段的订单分配问题。问题具体描述如下：

在某个时段（记作 t0）有 A 个骑手分布在商家附近位置，平台收到了 B 份来自同一区域的订单，不同骑手对该区域的熟悉程度也不同，骑手 i 可接受的最大订单数为 a[i]，对该区域的熟悉程度记为 c[i]。现平台需要将这部分订单合理地分配给骑手，希望骑手对分配到的区域足够熟悉，且骑手能够尽快到商家取货。

这部分我们着力解决问题(3)(4)(5):自行调查数据商家中午 11:00-13:00 的数据，并分成 12 个时段，预测中午某个时间段的订单数量；请设计一个合理的 cost 函数，来衡量某位骑手对于某份订单的适配度；选定(1)中设计的某个分区，按照(3)(4)当中得出的订单数量和 cost 指标，如何把这部分订单分配给不同的骑手。

### 3.2.1 订单数据调查与数据可视化

首先我们解决问题（3）自行调查数据商家中午 11:00-13:00 的数据，并分成 12 个时段，预测中午某个时间段的订单数量。

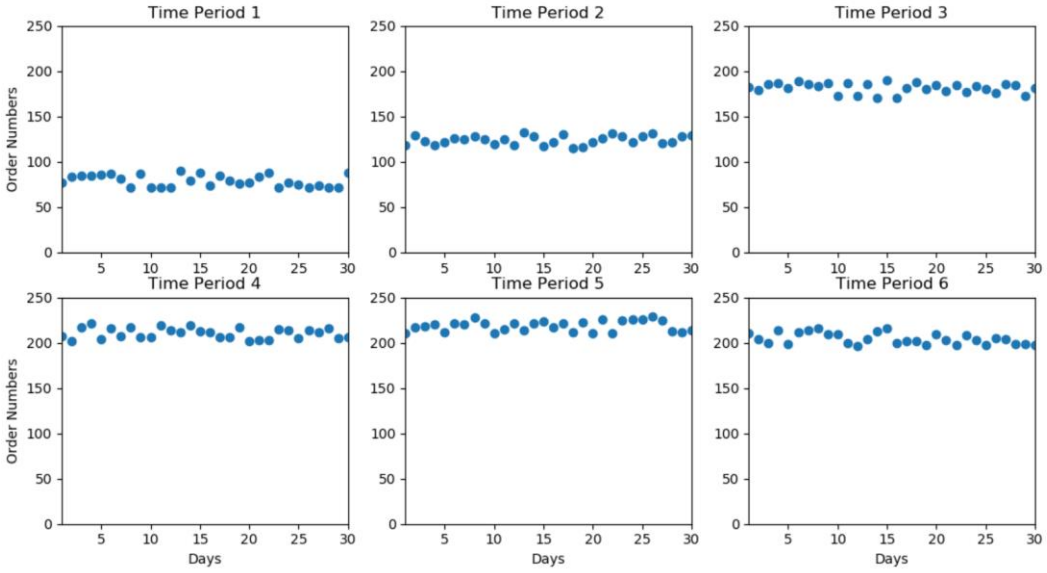
其实这里本可以直接把订单的分布堪称是泊松分布，但为了模型更贴近实际，我们需要自行收集数据。

我们调查了一位店家规模很大的外卖店的老板，调查到了一个月内的中午 11：00-13：00 订单数量变化。我们调查了一位店家规模很大的外卖店的老板，他告诉我们一天内的高峰主要在中午和晚上，而且分布情况很相似。于是我们小组决定只调查中午的情况，并且向该位老板调查到了一个月内的中午 11：00-13：00 订单数量变化。首先我们来看一下获取到的部分数据（10 组）情况，我们把 11：00-13：00 分成了 12 个时段(P1, P2, P3…P12)，该表格第一列表示第几天，第一行表示第几个时段。

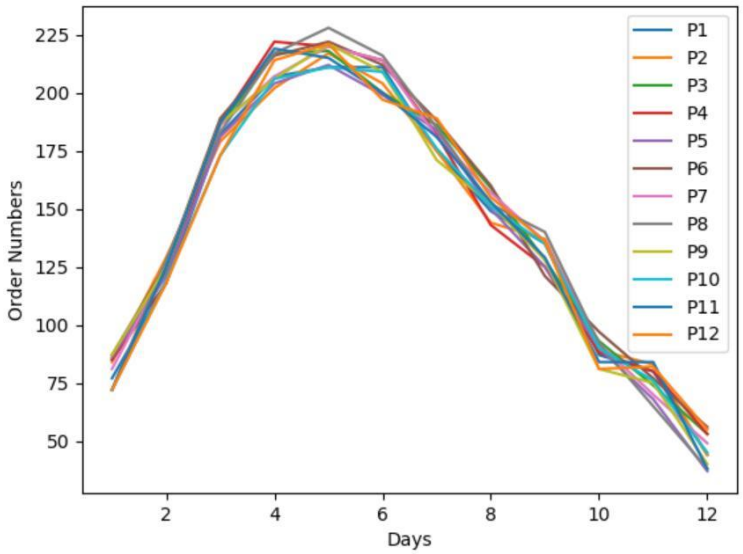
| Day | P1 | P2 | P3 | P4 | P5 | P6 | P7 | P8 | P9 | P10 | P11 | P12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 77 | 118 | 182 | 207 | 211 | 211 | 175 | 149 | 135 | 87 | 80 | 44 |

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 84 | 129 | 179 | 202 | 217 | 204 | 175 | 144 | 137 | 89 | 83 | 44 |
| 3 | 85 | 123 | 186 | 217 | 218 | 200 | 186 | 159 | 128 | 93 | 74 | 53 |
| 4 | 85 | 118 | 187 | 222 | 220 | 214 | 182 | 143 | 125 | 88 | 80 | 53 |
| 5 | 86 | 121 | 181 | 204 | 212 | 199 | 184 | 150 | 125 | 90 | 68 | 37 |
| 6 | 87 | 126 | 189 | 216 | 222 | 212 | 188 | 160 | 121 | 97 | 77 | 56 |
| 7 | 81 | 125 | 186 | 207 | 220 | 214 | 181 | 157 | 136 | 92 | 70 | 49 |
| 8 | 72 | 128 | 183 | 217 | 228 | 216 | 185 | 152 | 140 | 92 | 65 | 38 |
| 9 | 87 | 125 | 187 | 206 | 221 | 209 | 171 | 152 | 128 | 81 | 75 | 40 |
| 10 | 72 | 119 | 173 | 206 | 211 | 209 | 176 | 151 | 135 | 90 | 76 | 45 |

当然表格的形式不容易看出规律来，于是我们对前 6 个时段的数据进行了可视化，




一个月内每天前 6 个时段的订单量散点图（横轴代表第几天，纵轴代表订单数量）



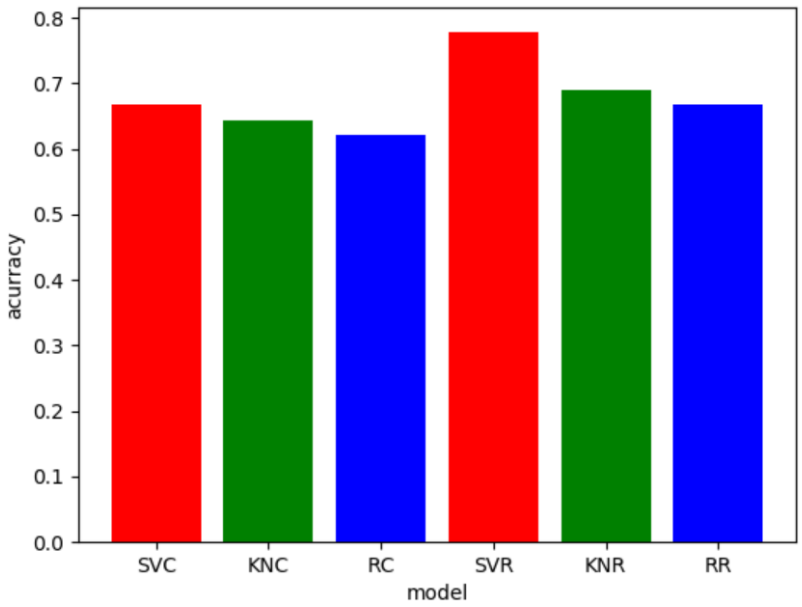

一个月内每天不同时段的订单量（横轴代表第几天，纵轴代表订单数量）

### 3.2.2 预测第 12 个时段的订单数量

不难看出一个月内同一时段的订单量变化是有一定规律的，我们可以用各种方法来对数据的分布进行拟合。由于工业界对于此类问题大多是选择机器学习来进行建模，所以这里我们叶决定利用机器学习方法来预测订单的分布情况。

我们可以利用前 i 个时间段的订单数量来预测下一个时间的订单数量。比如我们利用前 11 个时段的数据来预测第 12 个时段的订单数量。但是若要使用机器学习方法，我们的数据量显得有些太少了。在获取步得到大量的数据的情况下，往往需要自己对数据进行增强。于是我们自己造了一些伪数据(pseudo data)，在前 11 个时段的数据量上加上一个(-2, -2)的随机数，于是把一个月的数据构造出了 6 个月的数据。

通常来说我们可以把这个订单量预测问题看成是分类问题或者回归问题，在此情况下，看成回归问题显然是更好的，于是我们分别利用随机森林、SVM、KNN 的回归器来对数据分布进行预测。我们把数据按 7:3 分成训练集和测试集，下图是各个模型在测试集上的正确率对比（其中 SVC/KNC/RC 是分类器，SVR/KNR/RR 是回归器）。可以看出，在该数据情况下，SVR (SVM 的回归器形式)的正确率是最高的。




故我们选择 SVR 模型来预测未来的订单数量。如果现在我们前 11 个时段的订单数量是[ 73 122 177 209 228 199 174 159 130 95 67]，那么利用 SVR 模型可以预测第 12 个时段的数量是 **60** 份，**这个结论我们在之后的分析会作为一个条件来使用**。

### 3.2.3 骑手对订单 cost 的建立

我们希望设计一个 cost 矩阵，来合理地表示平台对订单分配地要求。显然这个 cost 矩阵需要用到路径优化的模型，所以 cost 是连接本问题中路径优化和订单分配的一个桥梁。

我们用 cost 来表示某位骑手对某个订单的适配程度，若 cost 越高，则骑手对该订单适配程度低，反之越高。我们希望一个骑手对该区域非常熟悉，对该类订单（不同类的订单对骑手本身的要求不同，如火锅订单的要求非常高）熟悉，且能够尽快到商家取货，用户对骑手的评分尽可能高，那么该骑手的 cost 就是比较小的，我们希望的就是减小 cost。那么总结下来与 cost 有关的指标有：

- 骑手送完当前外卖回到商家取货的时间 P1，P1 这个指标需要利用我们之前的；
- 用户对该骑手的评分 P2；
- 骑手对该区域的陌生程度 P3（0 代表对该区域非常熟悉，1 代表对该区域非常不熟悉）；
- 骑手对该类订单的陌生程度 P4（0 代表对该类订单非常熟悉，1 代表对该类订单非常不熟悉）。

由于我们需要减小这四个指标，那么就需要对这 4 个指标赋权并相加。用$cost_{ij}$表示第 i

个骑手对第 j 份订单的 cost：

$$\mathrm{cost_{ij}} = \mathrm{w_1} * \mathrm{P_1} + \mathrm{w_2} * \mathrm{P_2} + \mathrm{w_3} * \mathrm{P_3} + \mathrm{w_4} * \mathrm{P_4}$$

### 3.2.4 基于层级分析法的模型建立 cost 矩阵

为了得到四个指标的各自权重，这里我们采用层级分析法（AHP）。AHP 实质上不是一个完全的定量分析法，因为在设置对比矩阵的时候，依然依靠评分者的客观评分，但是它依然在实际应用中发挥很好的作用。AHP 在人力资源绩效、薪酬、人员测评、岗位分析的确定指标权重中，同样能够很好的应用。

首先需要构建指标对比矩阵，按照如下的规则进行度量：

aij=1，元素 i 与元素 j 对上一层次因素的重要性相同

aij=3，元素 i 比元素 j 略重要；

aij=7，元素 i 比元素 j 重要得多；

aij=9，元素 i 比元素 j 的极其重要；

aij=2n，n=1,2,3,4，元素 i 与 j 的重要性介于 aij=2n-1 与 aij=2n+1 之间；

反之 aij=1/ aij。

按照上述规则，我们采访了 7 名美团外卖员和 12 名同学，对权重做出如下初步评估

| P | P1 | P2 | P3 | P4 |
|---|---|---|---|---|
| P1 | 1 | 5 | 3 | 3 |
| P2 | 1/5 | 1 | 1/2 | 1/2 |
| P3 | 1/3 | 2 | 1 | 1 |
| P4 | 1/3 | 2 | 1 | 1 |
| SUM | 1.867 | 10.000 | 5.500 | 5.500 |

(1)对每一列进行求和

(2)由于各列量纲不同，我们需要对每一列进行归一化处理

$$B_{ij} = \frac{A_{ij}}{\sum A_{ij}}$$

其中：$\sum A_{ij}$的值为各列的和，如上图的 SUM 行，用各列的元素除以列和得到一个新矩阵 B

(3) 再对新矩阵每一行进行求和，得到特征向量

(4) 计算指标的权重：对特征向量进行归一化处理

$$W_i = \frac{B_j}{\sum B_j}$$

| B | P1 | P2 | P3 | P4 | SUM | Wi |
|---|---|---|---|---|---|---|
| P1 | 0.536 | 0.500 | 0.545 | 0.545 | 2.126 | 53.2% |

| | | | | | | |
|---|---|---|---|---|---|---|
| P2 | 0.107 | 0.100 | 0.091 | 0.091 | 0.389 | 9.70% |
| P3 | 0.179 | 0.200 | 0.182 | 0.182 | 0.743 | 18.6% |
| P4 | 0.179 | 0.200 | 0.182 | 0.182 | 0.743 | 18.6% |
| SUM | 1.000 | 1.000 | 1.000 | 1.000 | 4.000 | 100% |

简单的说，对矩阵进行归一化处理就可以得出指标的权重。但是这个权重不一定就是有效、可取的，我们需要对其检验，检验矩阵的一致性。

### 3.2.5 对比矩阵一致性检验权重的合理性

对矩阵进行归一化处理就可以得出指标的权重。但是这个权重不一定就是有效、可取的，我们需要对其检验，检验矩阵的一致性。检验步骤如下：

1. 计算矩阵的最大特征根：

$$\lambda_{max} = \frac{\sum(PW)_i}{nW_i}$$

这个公式的意思：两个矩阵相乘的结果是一个列向量，然后用列向量中的每一个元素除以阶数和相对应的权重的乘积。因此我们可求得 $\lambda_{max} = 4.101$

2. 计算判断矩阵的一致性指标（Constant index）

$$C.I. = \frac{\lambda_{max} - n}{n - 1}$$

示例中的 C.I.=（4.101-4）/3=0.034

3. 计算随机一致性比率 R.I

检验一个矩阵的一致性指标为矩阵的随机一致性比率，计算公式为：

$$C.R. = \frac{C.I.}{R.I}$$

R.I.表示：平均随机一致性指标，这个是一个常量，根据阶数可以在量表里查询。4 阶 R.I.值为 0.9，所以示例中的 C.R.=0.034/0.9=0.038<0.1，即保持显著水平，对比矩阵是保持一致性的，如果 C.R.>0.1，就表示未保持显著水平，需要对对比矩阵进行调整。通常情况下，自己调整很负责，不如重新评一次。

综上所述，我们对 P1, P2, P3, P4 依此赋权为 53.2%，9.70%，18.6%，18.6%。有了各项指标的权重，我们自然可以很快得出 cost 矩阵。

### 3.2.6 订单分配问题的求解

我们在 3.2.1 中预测了一天内不同时间段的订单分布后，我们得出了 12:50~13:00(即第

12 个时段)预计有 60 份订单，为了简化问题，我们认为 11:50~11:52 平均收到 12 份，并针对这 12 份订单重新描述问题如下：

在 11:50~11:52 有 3 个骑手分布在商家附近位置，平台收到了 12 份来自同一区域的订单，。现平台需要将这部分订单合理地分配给骑手，希望骑手对分配到的区域足够熟悉，且骑手能够尽快到商家取货。骑手的坐标和评分等信息我们不再设置，而是直接假设对于一批 12 份来自同一区域订单，有如下这样一个 cost 矩阵。

| | B[1] | B[2] | B[3] | B[4] | B[5] | B[6] | B[7] | B[8] | B[9] | B[10] | B[11] | B[12] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A[1] | 0.14 | 0.17 | 0.23 | 0.55 | 0.47 | 0.26 | 0.19 | 0.17 | 0.12 | 0.11 | 0.12 | 0.16 |
| A[2] | 0.37 | 0.40 | 0.49 | 0.09 | 0.05 | 0.53 | 0.42 | 0.39 | 0.12 | 0.12 | 0.14 | 0.19 |
| A[3] | 0.59 | 0.62 | 0.67 | 0.22 | 0.17 | 0.06 | 0.03 | 0.02 | 0.08 | 0.10 | 0.12 | 0.15 |

有了 cost 矩阵，我们的目标就是将该时段所有的订单都分配给骑手，并且使总的 cost 达到最小。

#### 3.2.6.1 0-1 规划求解

该问题可看作是一个指派问题，建立 0-1 规划组合优化模型。

优化目标：骑手对订单的总适配度最小

约束条件：一份订单只能由一名骑手派送；一名骑手一次可派送的最大订单数为 6

模型符号说明：

$c_{ij}$：表示骑手 i 对于订单 j 的适配度。

$x_{ij}$：表示决策矩阵，为一个 0-1 矩阵，即 $x_{ij}=1$表示将区域 j 内的订单分配给骑手 i；

$x_{ij}=0$ 表示不将区域 j 内的订单分配给骑手 i

建立如下的 0-1 规划数学模型：

$$\min\ P=\sum_{i=1}^{3}\sum_{j=1}^{12}x_{ij}c_{ij}$$

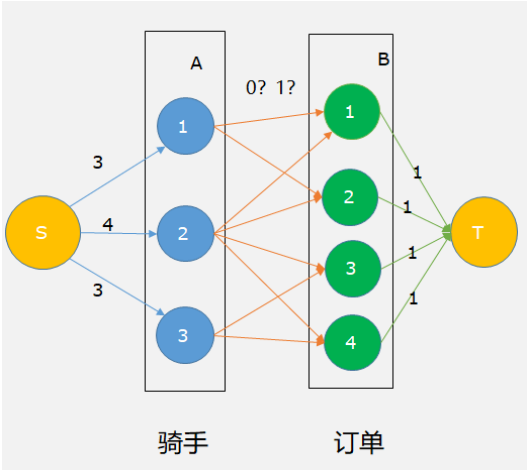
$$s.t.\begin{cases}\sum_{i=1}^{3}x_{ij}=1\\ \sum_{j=1}^{12}x_{ij}\le 6\\ x_{ij}=0,1\end{cases}$$

编写 LINGO 程序求解得：

x(1,1)=x(1,2)=x(1,3)=x(1,11)=x(2,4)=x(2,5)=x(3,6)=x(3,7)=x(3,8)=x(3,9)=x(3,10)=x(3,12)=1,其余 x(i,j)=0。即骑手 1 派送订单 1、2、3、11，骑手 2 派送订单 4、5，骑手 3 派送订单 6、7、8、9、10、12。最优值 P=1.24。

#### 3.2.6.2 网络流模型求解

我们把订单分配的问题考虑成二分图最大多重匹配问题。而二分图最大多重匹配问题实质上是一个网络流模型。我们让所有的骑手组成一个集合 A，所有的订单组成一个集合 B，集合 A 和集合 B 构成二分图中的两类顶点。在此基础上我们再引入一个源点 S，在 S 和 A[i]之间连接一条容量为 a[i]的边（即骑手可接受的最大订单数）；引入汇点 T，在 B[j]和 T 之间连接容量为 1 的边（每份订单只能分配给一个骑手）。而 A[i]和 B[i]之间的边的流量表示骑手 A[i]是否分配到了订单 B[j]，若分配到了则流量为 1，否则流量为 0。




至此，我们的网络流构建完成，而我们的目标是最大化流量并且订单合理地分配给骑手，希望骑手对分配到的区域足够熟悉，且骑手能够尽快到商家取货。把订单分配合理看成一个 cost，那么可以这个问题可以看成是一个求解网络流的最小费用最大流的问题。

通过 EK，Dinic，ISAP 算法可以得到网络流图中的最大流，一个网络流图中最大流的流量最大流量(max flow)是唯一的，但是达到最大流量 max flow 时每条边上的流量分配 f 并不是唯一的。 如果给网络流图中的每条边都设置一个费用 cost，表示单位流量流经该边时会导致花费 cost。那么在这些流量均为 max flow 的流量分配 f 中，存在一个流量总花费最小的最大流方案。

$$\min z = \sum \mathrm{cost}(u,v) * f(u,v)$$

$f(u,v)$为 边$(u,v)$上的流量， f 为某一个最大流方案

接下来我们就可以利用最小费用最大流问题的常规解法。采用贪心的思想，每次找到一条从源点到达汇点的路径(由于网络中存在负权边，不能使用 Dijskstra 因而使用 SPFA 来实现)，增加流量，且该条路径满足使得增加的流量的花费最小，直到无法找到一条从源点到达汇点的路径，算法结束。

由于最大流量有限，每执行一次循环流量都会增加，因此该算法肯定会结束，且同时流量也必定会达到网络的最大流量；同时由于每次都是增加的最小的花费，即当前的最小花费是所有到达当前流量 flow 时的花费最小值，因此最后的总花费最小。

**具体求解步骤：**

（1）利用 SPFA 算法找到一条从源点到达汇点的“距离最短”的路径，“距离”使用该路径上的边的单位费用之和来衡量。

（2）然后找出这条路径上的边的容量的最小值 f，则当前最大流 max flow 扩充 f，同时当前最小费用 min cost 扩充 f*min cost(s,t)。

（3）将这条路径上的每条正向边的容量都减少 f，每条反向边的容量都增加 f。

（4）重复（1）-（3）直到无法找到从源点到达汇点的路径。

解出的答案同样是骑手 1 派送订单 1、2、3、11，骑手 2 派送订单 4、5，骑手 3 派送订单 6、7、8、9、10、12。

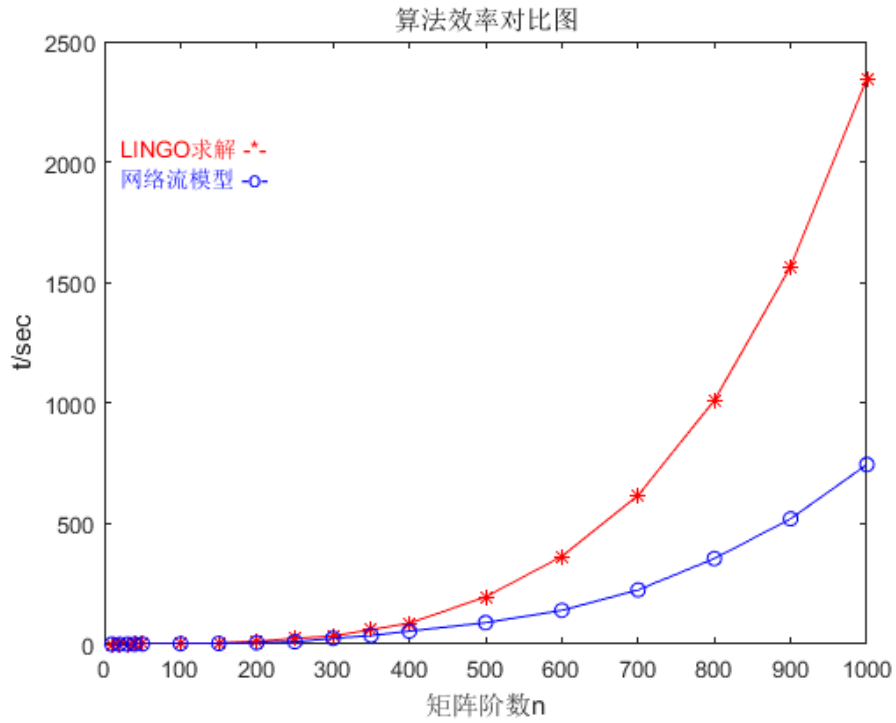
**3、算法效率对比**

由于外卖配送具有时效性，所以算法求解时间在订单配送这个模型中是非常重要的。随机生成不同阶数的 cost 矩阵，测试两种算法各自的求解时间如下：

| LINGO直接求解 | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 矩阵阶数n | 10 | 20 | 30 | 40 | 50 | 100 | 150 | 200 | 250 |
| 求解时间t/s | 0 | 0 | 0 | 1 | 2 | 3 | 6 | 12 | 24 |
| 矩阵阶数n | 300 | 350 | 400 | 500 | 600 | 700 | 800 | 900 | 1000 |
| 求解时间t/s | 34 | 60 | 87 | 196 | 364 | 617 | 1011 | 1566 | 2345 |

| 网络流模型求解 | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 矩阵阶数n | 10 | 20 | 30 | 40 | 50 | 100 | 150 | 200 | 250 |
| 求解时间t/s | 0 | 0 | 0 | 0 | 1 | 2 | 3 | 6 | 10 |
| 矩阵阶数n | 300 | 350 | 400 | 500 | 600 | 700 | 800 | 900 | 1000 |
| 求解时间t/s | 24 | 36 | 54 | 89 | 140 | 225 | 355 | 520 | 745 |

算法效率对比图：




可以看到，当模型矩阵规模较小时，两者的计算速度差别不大，但是当矩阵规模较大时，明显可以看出采用网络流模型要比用 LINGO 直接求解快得多，矩阵规模越大体现得越明显。主要是由于 LINGO 求解使用的是暴力搜索，算法效率低，而网络流模型则是采取启发式算法，能够很好地提升算法效率，节省时间。

但实际上两种算法求解时间还是太长了，而且这只是考虑给定时间内的一批订单，没有考虑到未来订单的情况。下面利用强化学习来考虑未来订单分配情况。

## 3.3 未来订单分配问题的分析与建模

### 3.3.1 问题分析

这部分我们着力解决问题（6），即考虑序列决策问题，针对未来可能出现的订单，对订单分配问题建模，在(5)的条件下，如何对这部分订单进行合理地分配。

以上只是针对给定的一批订单进行匹配决策的优化问题在建模时所需考虑的部分因素。事实上，在外卖配送场景中，我们希望的不是单次决策的最优，而是策略在一段时间应用后的累积收益最大。换句话说，我们不追求某一个订单的指派是最优的，而是希望一天下来，所有的订单指派结果整体上是全局最优的。

### 3.3.2 基于强化学习的模型求解

订单分配是一个动态决策过程，当前订单分配结果直接影响到下一订单的分配，符合 Markov 决策过程(Markov Decision Process, MDP) 即系统下个状态不仅和当前的状态有关, 也和当前采取的动作有关。

强化学习的原理可以简述为如下：Agent 通过执行某种操作，引起某种改变，当它所作出的改变给其带来了正向的结果，则 Agent 应该得到奖励，并且应该记住这种奖励；当它所执行的某种操作给环境带来了负向的结果，那么它应该受到惩罚，而且这种惩罚应该对其造成一定的影响。因而，通过不断的行动，不断的试错，Agent 会渐渐地明白，在这样的环境下它该执行怎样的操作，才会获得最大的收益。因而，在训练后，Agent 可以在面对不同的环境时，选择能使其收益最大的操作，从而实现智能的行为。从某种程度上说，强化学习类似于生物个体在环境中不断尝试而获得知识的过程。

常见的强化学习算法包括 Q-Learning，Sarsa, TD-Learning 等等。这里，我们采用的方法是 Q-learning 算法。下面着重介绍一下在这次工程中我们实现的 Q-Learning 结合神经网络的方法，也即 Deep Q Network。

#### 3.3.2.1 Q-learning 实现原理

如前所述，在强化学习中，我们关心的是在给定状态下，应该采取怎样的行动。在 Q-learning 中，最重要的是对当前状态以及所能采取动作的评估。例如，我们在状态 S 下可以采取四种行动，而经过了前一阶段的学习，在当前的环境中，采取这四种行动所能获得的奖励分别记为 Q（S,a1），Q（S，a2），Q（S，a3）和 Q（S，a4）。我们所要找到的即是最大的 Q（S，a*），即寻找到奖励最大的步骤。这里，奖励更新公式如下所示：

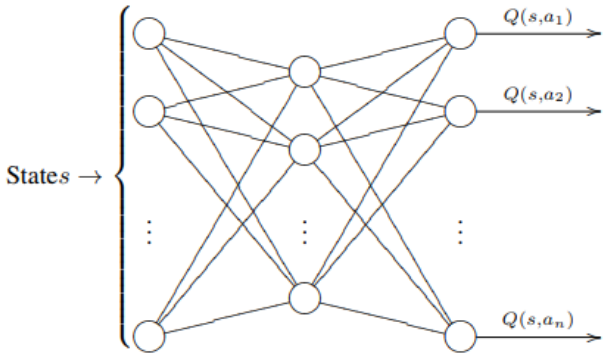
$$\widehat{\mathrm{Q}}(s_t, a_t) \leftarrow \widehat{\mathrm{Q}}(s_t, a_t) + \alpha(r_t + \gamma \max_a \hat{Q}\,(s_{t+1}, \mathrm{a}) - \widehat{\mathrm{Q}}(s_t, a_t))$$

这里，α 为学习速率（learning rate）。而 γ 为增益递减速率，这个值存在的意义在于：距离当前状态越近的状态，对目前状态和行为所造成的影响最大，而离当前状态越远的，所造成的影响就应该越小。

#### 3.3.2.2 Deep Q-Network

但是我们会注意到这样的状态空间实在太大，每一维又有很多个取值可能。当地图变得更加大的时候，这样多的状态远远超过了计算机所能承受的范围，而且我们还需要计算并存储 Q 值。所以，我们是不可能通过表格来存储状态的。解决这个难题的方法在于神经网络：

将当前状态作为输入，将神经网络的输出设定为对在此状态下采取某种动作的 Q 值。这个神经网络的形状如下所示：




损失函数：

$$L(\varpi) = E\left[\left(r + \gamma \max_{a'} Q(s', a', \varpi) - Q(s, a, \varpi)\right)^2\right]$$

那么我们是怎么训练这个神经网络的呢，通过 Q-Learning 获取无限量的训练样本，然后对神经网络进行训练。核心就是反复试验，然后存储数据。接下来数据存到一定程度，就每次随机采样数据，然后利用梯度下降和反向传播更新神经网络的参数。

**DQN 算法步骤如下：**

通过上面叙述的方法，我们在建立起算法框架前面所面临的问题基本都可以得到解决。因此，我们得到的算法步骤如下所示：

1. 根据当前地图，骑手和订单的情况，得到状态 $S_t$；
2. 对此时所有可能的动作 $a_t$，利用神经网络计算 Q(St,at)；
3. 利用 ε-greedy 策略，选择一个动作$a_t$；
4. 根据 8，计算$Q_{new}(S_t - 1, a_t - 1)$；
5. 使用神经网络计算$Q(S_t - 1, a_t - 1)$；
6. 得到神经网络的误差为$Q_{new}(S_t - 1, a_t - 1) - Q(S_t, a_{t-1})$，更新神经网络；
7. 进入到下一个状态，即$S_{t-1} \leq S_t, a_{t-1} \leq a_t$；
8. 执行动作$a_t$。

强化学习的目标是获得最大的长期收益，在系统处理订单的过程中，及时收益相当于该时间间隙所有订单的收益。

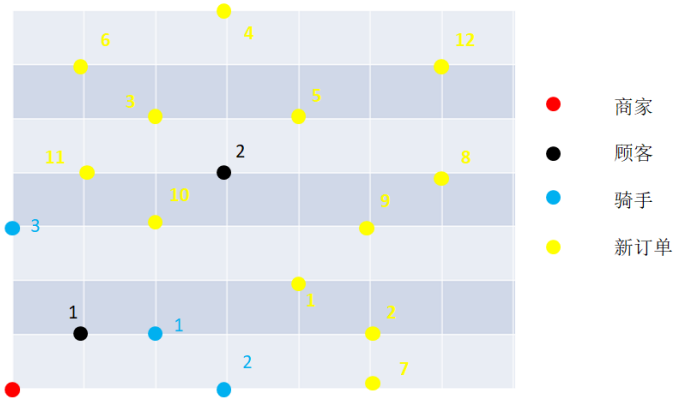
### 3.3.2.3 实验布置及结果

沿袭前面指派问题当中的条件，我们模拟 8*8 的方格地图，1 个位于(0, 0)的商家，3 个

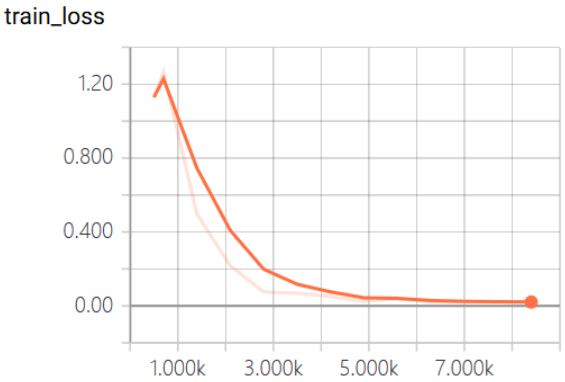
正在路上的骑手，每位骑手最多携带 6 份订单，每时间间隙平台收到 12 份外卖，一共模拟 10 个时间间隙。当前在第一个时间间隙，骑手顾客分布图如下，平台收到了 12 份订单。

强化学习的关键在于回报函数 R 的设计，按照我们之前的层次分析法得出的 cost 设计 R

$$R=\begin{cases}\dfrac{1}{\text{cost}} & (\text{若 } R>0.5)\\ -0.1, & (\text{若 } R\le 0.5)\end{cases}$$




DQN 训练阶段 loss 下降曲线，因为我们的样本规模小，所以很快就收敛了。




对于上述情况，当前的状态对应的各种 Action 的 Q 值(选取其中一部分)如下表。

| **Action** | **0** | **1** | **2** | **3** | **4** | **5** |
|---|---|---|---|---|---|---|
| **Q-score** | 0.161 | -0.0593 | 0.0451 | 0.0313 | -0.0633 | -0.0368 |
| **Action** | **6** | **7** | **8** | **9** | **10** | **11** |
| **Q-score** | -0.1068 | -0.1711 | **0.6384** | 0.3337 | 0.2548 | -0.162 |
| **Action** | **12** | **13** | **14** | **15** | **16** | **17** |
| **Q-score** | -0.0128 | -0.1407 | -0.4343 | 0.042 | 0.0146 | -0.5599 |
| **Action** | **18** | **19** | **20** | **21** | **22** | **23** |

| | | | | | | |
|---|---|---|---|---|---|---|
| **Q-score** | 0.2577 | 0.0577 | 0.012 | 0.1981 | 0.2764 | 0.0909 |
| **Action** | **24** | **25** | **26** | **27** | **28** | **29** |
| **Q-score** | 0.0758 | 0.386 | 0.5309 | -0.2972 | 0.2663 | 0.3685 |

从上表可知，此时应当采取 Action8 即(1,1,2,2,2,3,3,3,3,3,1,3)，即骑手 1 派送订单 1、2、11，骑手 2 派送订单 3、4、5，骑手 3 派送订单 6、7、8、9、10、12。但是此时的解就与 0-1 规划指派问题和网络流的给出的答案有所差别。但是强化学习考虑的是使预计的这 10 个时间间隙得收益最大，而且线上运行速度比前两者快很多，基本上是线性时间。

## 4、模型与结果分析

分区+路径规划模型：本模型从外卖人数安排，路线安排，时间安排和外卖员负重数等几个方面综合考虑，得出了最优的方案。分区时运用四叉树的思想，没有可以寻求最优解，而是首先寻找可行解，然后进行适度优化，具有较强的普适性；利用蚁群算法解决 TSP 问题，虽然不一定能找到全局最优解但减小了算法复杂度。

订单分配模型：0-1 规划模型相当于是暴力搜索，算法复杂度高，网络流模型(最小费用最大流)降低了算法复杂度，但还是不够理想，强化学习算法的引入，能解决序列决策问题。缺点是编程难度较大，训练时间长，对训练技巧要求高。

机器学习预测订单数量：由于我们的数据量非常小，在我们获得的这个数据集上效果较好，但若直接拿到真实环境去预测，误差肯定还是不容忽视的。不过值得一提的是，工业界的数据量是非常巨大的，足以训练好各个机器学习模型，而且当前工业界对预测类问题的主要模型还是使用深度神经网络，参数较多，但是庞大的数据量也足以让深度神经网络训练得足够好。

## 5、结论与展望

本文针对提出的 6 个问题，依次进行合理地建模，将这六个问题的解决方案串联起来，我们可以得到一个完整的订单智能分配系统。由于问题本身主要可分成路径优化和订单配送，我们用 cost 矩阵来建立两者的之间的联系。最后模型能够以较高的效率和准确度解决外卖平台的订单分配问题。

不足的是，该模型易受天气、订单结构、骑手水平等外在随机因素影响，算法效果容易被随机因素湮没从而无法准确评估。未来可以加入这些随机因素来提高模型鲁棒性。另外，

为了提高分配准确度，可以将强化学习方法与启发式搜索相结合，这样可以提高线上决策的精度，而且运行效率也不低。

另一方面，本模型仅用机器学习模型预测了某个时段的订单数量，还有一些因素如商家的出餐时间、到用户所在楼宇上下楼的时间、未来的订单、骑行速度、红绿灯耗时、骑行导航路径还未进行准确预估，可以进一步在此扩展。

## 6、参考文献

[1] 郭建宏，钱莲文，欧阳钟辉，彭道黎.基于 GIS 的城市水果物流分区配送辅助系统.2007,8

[2] 蔡锁章.《数学建模》.北京，中国林业出版社, 2003

[3] 胡列格，何其超，盛玉奎 .《物流运筹学》.北京,电子工业出版社, 2005

[4] 邓志龙，张琦玮，曹皓，谷志阳.一种基于深度强化学习的调度优化方法.2017

[5] 王一松，王直杰. 基于实时交通信息的最优路径规划算法研究. 计算机与现代化. 2013

[6] 赵百轶，张利军，贾鹤鸣. 基于四叉树和改进蚁群算法的全局路径规划. 应用科技, 2011

[7] 陈淮莉，吴梦姣. 基于强化学习的在线订单配送时隙运能分配.上海海事大学学报. 2017

[8] JA Hartigan, MA Wong. A k-means clustering algorithm. JSTOR, 1979

[9] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D.Warde-Farley, S. Ozair, A. Courville, andY. Bengio. Reinforcement learning in the game of Othello: Learning against a fixed opponent and learning from self-play. IEEE International Symposium on Adaptive Dynamic Programming & Reinforcement Learning. 2013.

[10] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. In NIPS Deep Learning Workshop. 2013.

## 7、大作业心得

我们小组在选题的时候并没有选择参照以往的数学建模国赛/美赛的题目，而是选择自己针对一个现象，并提出问题，查找相应的数据。这对我们来说是一个非常大的挑战，没有具体数学建模的题目的指引，使得我们一开始对问题的定义并不是那么地清晰，光是讨论问

题与找资料就花费了一个多星期。在建模的过程我们还不断对自己之前的假设提出质疑，总觉得有些地方不甚合理。我们最初只是把订单智能配送大致地分成了路径规划和订单分配，在撰写论文初稿的时候发现条理不是特别清楚，于是细分了问题，自己设计了题目，使我们的目标更加明确。

另一方面，我们的这个选题其实难度是比较大的，商家有坐标点，骑手有坐标点，订单有坐标点，而且各大外卖平台对数据进行了封锁，我们没法通过爬虫来找到我们想要的数据。虽然我们小组只有四个人，但小组成员齐心协力，还是完成了这个课题，我们都感到收获很大，不仅仅是一些优化方法，更重要的是提出问题的能力以及解决问题的能力。

## 8、成员风采展示

成员合影

成员调查饿了么店家：

成员展示：

## 9、组员分工

| | ** | *** | *** | *** |
|---|---|---|---|---|
| 论文整合 | √ | √ | √ | |
| PPT 制作 | √ | | √ | |
| 展示 | √ | | | |
| 调查外卖商家 | √ | √ | √ | √ |
| 数据分析与处理 | √ | √ | | |
| K-means 聚类 | | | | √ |
| 四叉树模型 | | | √ | √ |
| TSP 求解 | | | √ | √ |
| 机器学习预测订单数量 | √ | √ | | |

| | | | | |
|---|---|---|---|---|
| 0-1 规划 | | √ | | |
| 网络流模型 | √ | | | |
| 强化学习 | √ | | | |
| 拍照 | | | | √ |

## 10、附录

相关代码及资料请见附件。

蚁群算法软件是我们网上找到的已经实现好的软件，其余代码均为组员自己学习相关资料后编写。