高等运筹学

期末专题

管理科学与工程学系

仲维礼

题目

星巴克 新门市位置

问题描述:星巴克企业想要在北京开设新的实体门店，根据它们的市场调查，发现只要有图书馆地方，人们就会有买咖啡的消费行为，因此在开设有限数量的门市下，希望这些店离图书馆距离不远。

模型建立与假设:这是经典的选址问题，我们采用 P-中位方法建构以下数学式，

$$\underset{S}{arg\ \min} \sum_{i=1}^{K} \sum_{X \in Si} \| \ X - \ \mu_i \ \|^2 \ , i = 1,2, \ldots \ldots K$$

$$s.t. \quad \mu_1 + \mu_2 + \cdots + \mu_K \ = \ \mathrm{n}$$

$$\mu_i \ are \ bounded \ in \ X$$

Assumption: Each library is linked exactly one coffeeshop

Each coffeeshop can link not only one library

Coffeeshop cadidate locations are same as library location

数据集:透过百度地图开放资源我们可以获得全北京市图书馆的据点，但观察数据集后我们集中在北京的海淀区、东城区、西城区、朝阳区这四个区域，因为这四个区域的加总占全部数据集的十分之七，并且可以降低 outliner 点的影响。

原数据(400 笔)

	area	name	lat	lng
0	海淀区	中国国家图书馆	39.949641	116.330065
1	朝阳区	首都图书馆	39.875885	116.469182
2	西城区	北京图书大厦	39.914075	116.383456
3	通州区	通州区图书馆	39.918552	116.693865
4	海淀区	海淀区图书馆	39.984178	116.315950

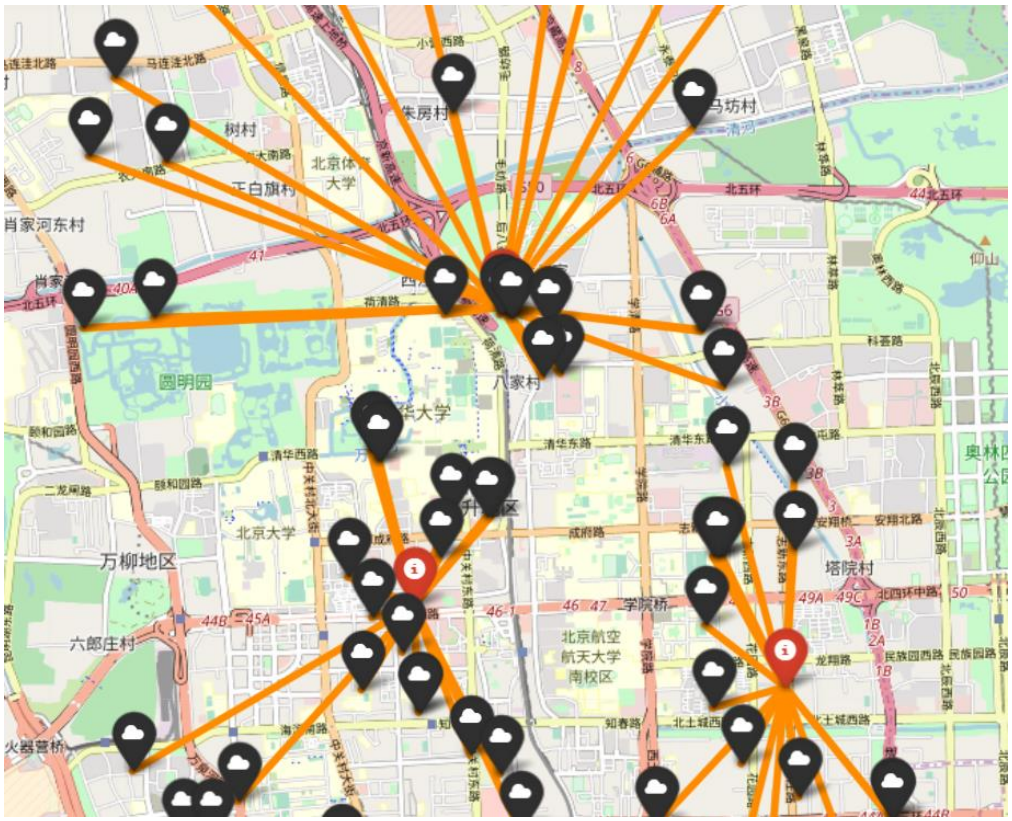
海淀区(95 笔)

	area	name	lat	lng
0	海淀区	中国国家图书馆	39.949641	116.330065
1	海淀区	海淀区图书馆	39.984178	116.315950
2	海淀区	中国科学院文献情报中心	39.993046	116.329603
3	海淀区	清华大学法律图书馆楼	40.005203	116.335359
4	海淀区	北京大学-图书馆	39.997899	116.316801

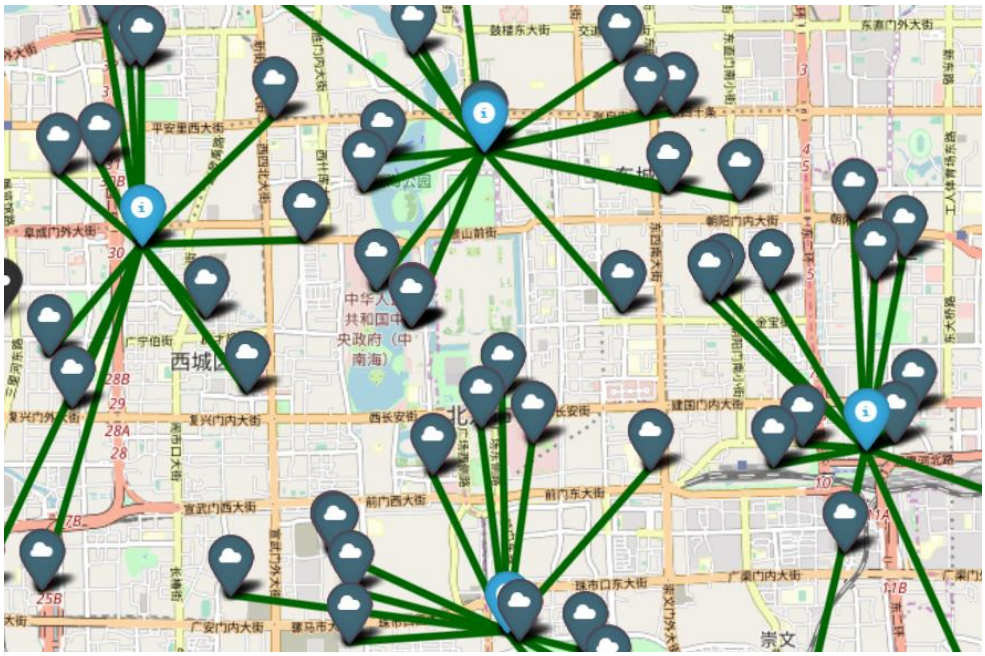
求解方法:我们采用两种方法进行求解，一个是透过 cplex 另一个是透过机器学习中 K 均值聚类，cplex 由于软件建模方式，从原本线性规划问题变为混合整数规划，μ_i 是经纬度不是整数的，而过程会利用二元矩阵代表这图书馆附近是有开还是没开，但由于此矩阵规模太大了，运算上无法负荷，因此我们采取分区运算有海淀区、东西城区、朝阳区分三部分求解。

求解结果 cplex:

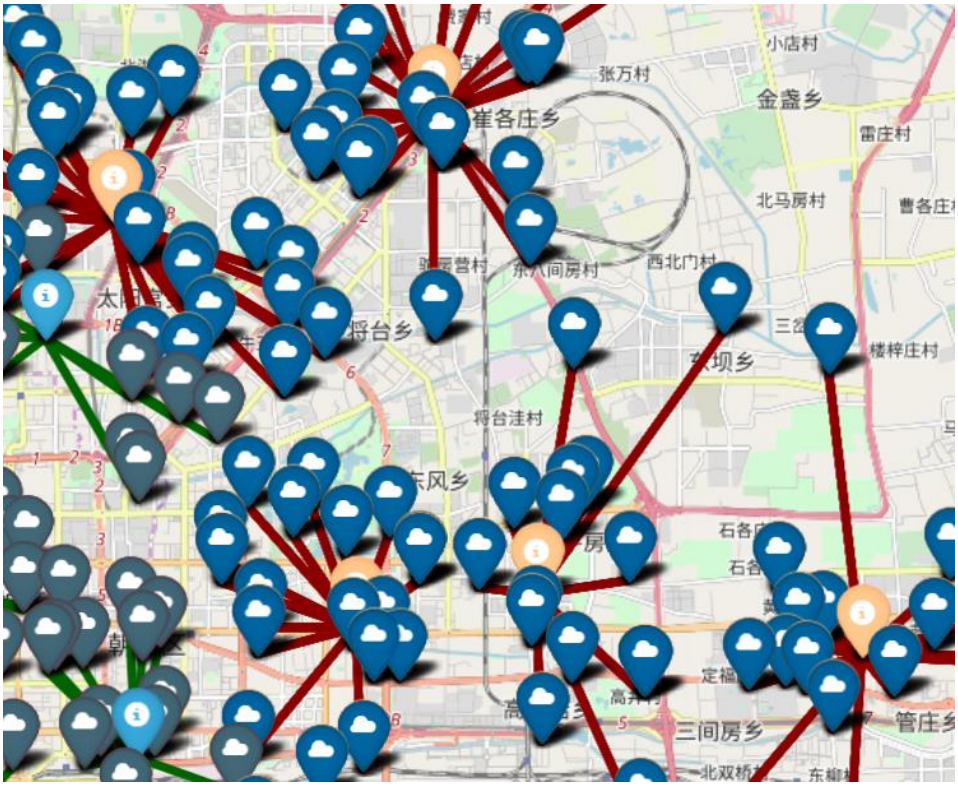
海淀区(红色为新设咖啡门市据点；黑色为图书馆；n=5)



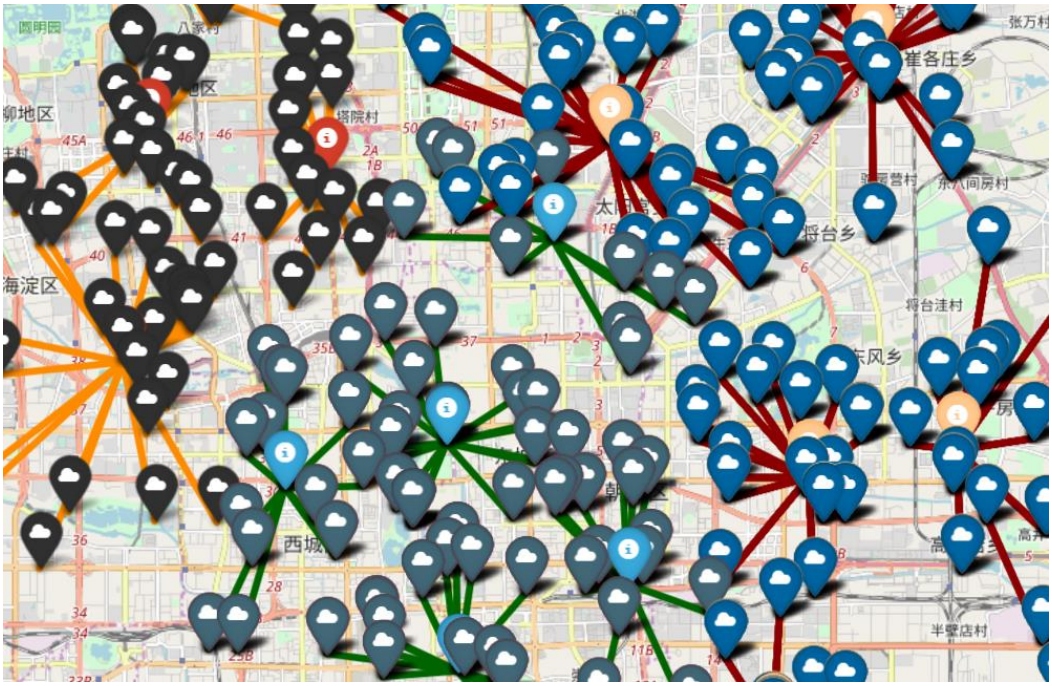
东西城区(浅蓝色为新设咖啡门市据点；深蓝色为图书馆；n=5)



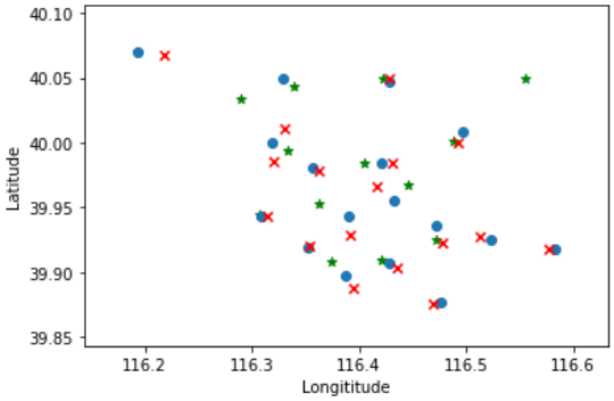
朝阳城区(橘色为新设咖啡门市据点；蓝色为图书馆；n=7)



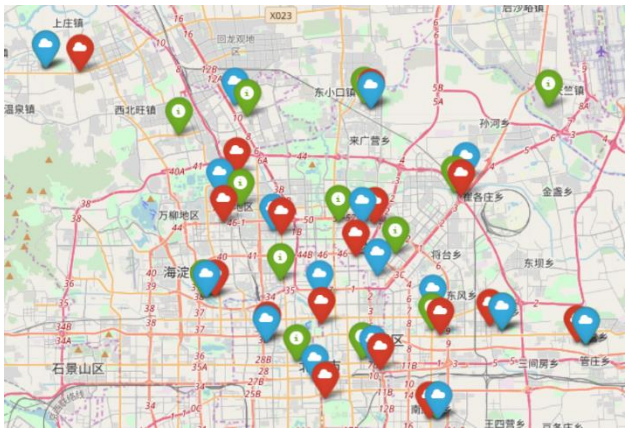
总图



cplex、k-means 分区、k-means 不分区 新设咖啡门市据点结果比较:



cplex ［红色］
kmeans 分区 ［蓝色］
kmeans 不分区 ［绿色］

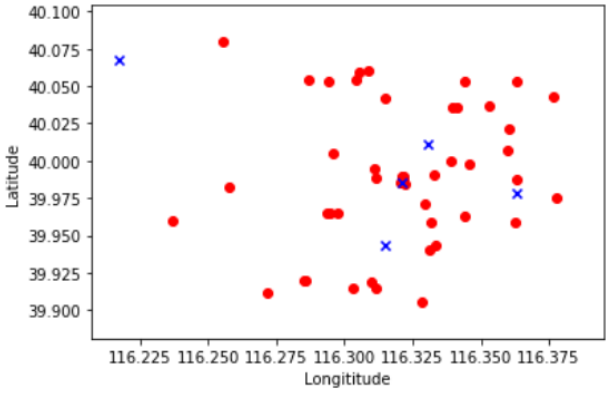


cplex ［红色］
kmeans 分区 ［蓝色］
kmeans 不分区 ［绿色］

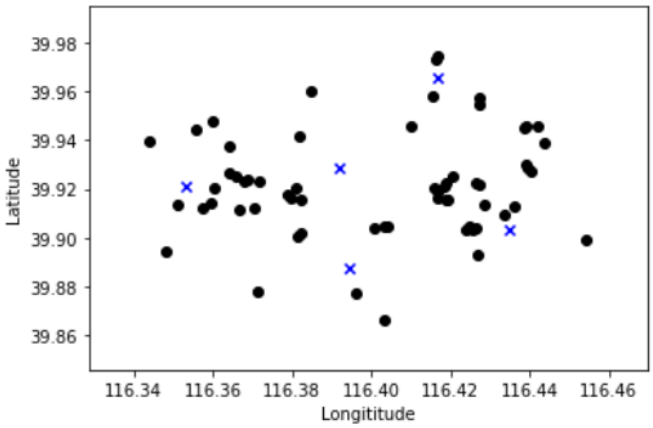
可以发现 cplex 求解结果(红色)和 kmeans 分区(蓝色)几乎重迭，不完全重迭的原因是 cplex 的假设，其中一条，门市位置和图书馆位置相同，而 kmeans 并没有这条约束，至于 kmeans 不分区(绿色)结果与另两者不同是因为它没有分区的约束，所以相差更多。(详见附注距离计算)

cplex 求解结果与真实星巴克据点比较:

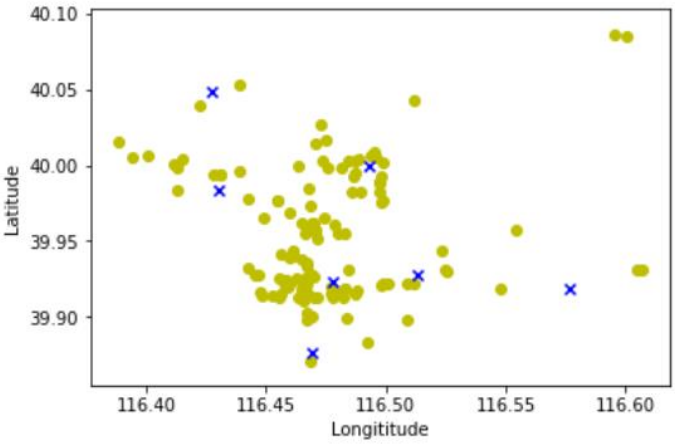
海淀区(红色为实际星巴克据点；蓝色为 cplex 求解结果；n=5)



东西城区(黑色为实际星巴克据点；蓝色为 cplex 求解结果；n=5)



朝阳区(黄色为实际星巴克据点；蓝色为 cplex 求解结果；n=7)



可以发现几件事情，我们预设的开店数目远远小于实际开的，毕竟星巴克要抢占市场不可能只开几间店，以及星巴克密度越高的地方，都有一个求解的结果或者星巴克据点的中心也有求解的结果，代表如果真的只能限制几家的话，或许求解结果真是不错的选择，当然星巴克企业每开一间门市是会考虑很多的，不可能仅仅只因为距离图书馆去设计，说不定其实是因为商圈去设计的，这些都可以去探讨的。

cplex 求解算法与代码实践:

我们的目标是要距离最小，所以肯定要设计距离函数:

```
def get_distance(p1, p2):
    return great_circle((p1.x, p1.y), (p2.x, p2.y)).miles
```

约束一 (μ_i *are bounded in* X):

```
BIGNUM = 999999999
for c_loc in coffeeshop_locations:
    for b in libraries:
        if get_distance(c_loc, b) >= BIGNUM:
            mdl.add_constraint(link_vars[c_loc, b] == 0,"ct_forbid_
              {0!s}_{1!s}".format(c_loc, b))
```

约束二 (Each library is linked exactly one coffeeshop):

```
mdl.add_constraints(mdl.sum(link_vars[c_loc, b] for c_loc in
coffeeshop_locations) == 1
                    for b in libraries)
```

约束三 (Each coffeeshop can link not only one library):

```
mdl.add_constraints(link_vars[c_loc, b] <= coffeeshop_vars[c_loc]
                    for b in libraries
                    for c_loc in coffeeshop_locations)
```

约束四 (μ1＋μ2＋…＋μK ＝ n):

```
mdl.add_constraint(mdl.sum(coffeeshop_vars[c_loc] for c_loc in
coffeeshop_locations) == 5)
```

约束五 (Coffeeshop cadidate locations are same as library location):

```
libraries = set(libraries)
coffeeshop_locations = libraries
```

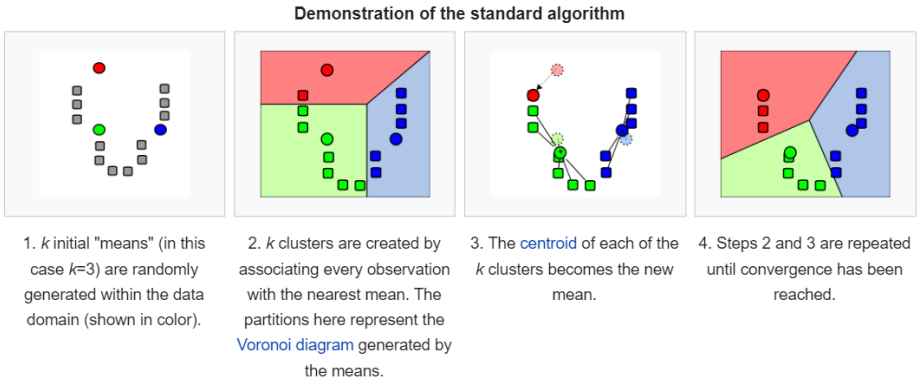
求解

```
total_distance = mdl.sum(link_vars[c_loc, b] * get_distance(c_loc, b)
for c_loc in coffeeshop_locations for b in libraries)
mdl.minimize(total_distance)
```

二元矩阵(撷取片段)

Minimize:
obj:
Subject To:
c1: x75 - x1 <= 0
c2: x149 - x2 <= 0
c5489: x87 + x161 + x235 + x309 + x383 + x457 + x531 + x605 + x679 + x753 + x827 + x901 + x975 + x1049 + x1123 + x1197 + x1271 + x1345 + x1419+ x1493 + x1567 + x1641 + x1715 + x1789 + x1863 + x1937 + x2011 + x2085+ x2159 + x2233 + x2307 + x2381 + x2455 + x2529 + x2603 + x2677 + x2751+ x2825 + x2899 + x2973 + x3047 + x3121 + x3195 + x3269 + x3343 + x3417+ x3491 + x3565 + x3639 + x3713 + x3787 + x3861 + x3935 + x4009 + x4083 + x4157 + x4231 + x4305 + x4379 + x4453 + x4527 + x4601 + x4675 + x4749+ x4823 + x4897 + x4971 + x5045 + x5119 + x5193 + x5267 + x5341 + x5415 + x5489 = 1
0 <= x35 <= 1
0 <= x36 <= 1
Binaries
 x1 x2 x3 x4 x5 x6 x7·····

kmeans 求解算法与代码实践:




```
kmeans=KMeans(n_clusters=5,random_state=0)
kmeans.fit(data_train)
```

附注距离计算

```
distances1 = np.linalg.norm(cplexAns - kmeans_sperate, axis=0)
distances2 = np.linalg.norm(cplexAns - kmeans_nosperate, axis=0)
distances1 = 0.3389108896235739
distances2 = 0.6523480768210991
```