武漢理工大學

数学建模暑期培训论文

第 4 题

基于帝国竞争算法求解派出所选址问题的随机规划模型

第 1 组

姓名	方向
阮滨（组长）	编程
王宇	建模
王家柯	写作

2020 年 8 月 20 日

摘要

派出所作为维持社会秩序基层组织，合理的选址对其执法的及时性有着重要作用。本文构建了人口密度、通行速度以及协同工作效益等指标，建立了派出所选址问题的随机规划模型，并通过**帝国竞争算法**求解，得到最佳的派出所选址地点。

针对问题一，从离散和连续两个角度分别建立派出所选址的优化模型，得到派出所最佳的选址。首先以节点 1 为原点，得到各节点的坐标，将数据汇总为节点 -速度矩阵，并分别构建了**基于通行时段的速度权重**和**基于人口密度的空间权重**对通行时间修正。以**最小化最长通行时间为目标函数**，考虑**选址节点的空间约束**，分别建立了**离散和连续的选址模型**。通过帝国竞争算法求解，得到离散条件下最佳选址坐标为 **(0,0)**，连续条件下则为 **(0,-0.4178)**，选址坐落于辖区中心，符合国家对于派出所建设的基本要求。最后通过灵敏度分析，认为结果受参数变化的影响较小。

针对问题二，结合附件中的犯罪数据，建立了派出所选址的**随机规划模型**。首先，引入节点处的**犯罪恶劣程度**以及派出所的**覆盖衰减函数**的概念，得到了**治安不稳定系数**。考虑到突发事件的不确定性，通过最大似然估计得到各节点未来犯罪事件服从的分布，建立**带有随机参数的不确定规划模型**，并在确定置信水平后，转化为**机会约束模型**。通过**帝国竞争算法**求解，得到最佳选址坐标为 **(-1,-3)**。通过与治安不稳定系数的分布比较，发现派出所坐落于违法相对集中、交通便捷的节点处。

针对问题三，在问题二已经选定派出所建址的基础上，增设一个新的派出所共同治安。考虑到派出所治安时效性，每个节点应先由通行时间最短的派出所负责，故建立**基于区域划分的选择指标**；此外，建立**基于协同执法的合作指标**，用于衡量两个派出所之间协作的效果。综合上述指标，建立新增派出所选址的**机会约束模型**。通过帝国竞争算法求解，得到新增的派出所选址为 **(2,-4)**，进而获得了每个派出所的管辖区域，区域划分的结果较为均匀。

本文的优点包括：1. 通过帝国竞争算法对选址问题的机会约束模型求解，帝国同化的操作使其全局搜索能力强，收敛速度快；2. 考虑到未来突发事件的不确定性，运用随机规划的相关理论，建立了机会约束模型，使结果具有较强的适应性；3. 从人口密度、覆盖效益衰减以及派出所之间的合作关系等角度建立了多个反映执法效益的指标，并参考《**公安派出所建设标准**》的相关要求核实派出所选址的合理性。

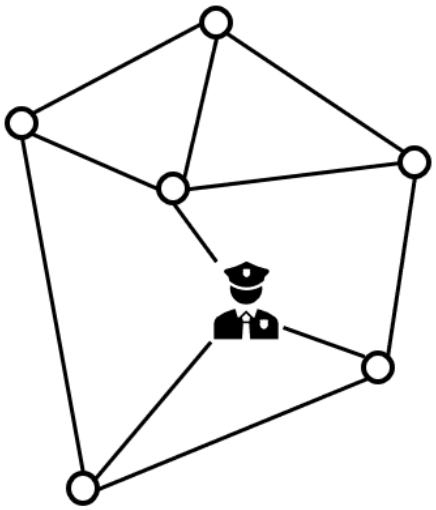
关键词： 帝国竞争算法　覆盖衰减函数　随机规划　机会约束模型

目录

1 问题重述

1.1 问题背景

派出所作为一个公安系统的基层组织，它在维护公共秩序，预防、制止犯罪活动，向群众宣传法制精神等方面具有重大作用。因此派出所的选址对民警能够快速有效地到达辖区内突发事件的地点十分重要。



1.2 问题重述

问题一：结合附件一的地图，通过建立数学模型求解出派出所的最佳选址，使得该派出所可以及时有效地到达突发事件发生的地点。

问题二：结合附件三中的最近十年来各个节点附近发生的违反犯罪次数，以及平均每次发生违法犯罪后损失的财产金额。以减少犯罪次数和财产损失为目的建立数学模型，求解派出所选址的最佳位置。

问题三：由于该市决定在该辖区增设一个新的派出所。在问题二原有派出所不变的前提下建立一个新的派出所，通过建立数学模型求解出新增派出所的位置，并规划出两所派出所的管理区域。

2 模型的假设

- 假设警车出行全程只在一个时间段；
- 假设派出所的选址只在道路上；
- 假设警车出行均以最大限制速度出行；
- 不存在由于需要同时处理多件事导致派出所警力不足的情况；

3 符号说明

符号	意义
$T'_{i,j}$	从节点 i 到节点 j 的时间
w_i	时间段 i 车速权重
u_j	节点犯罪恶劣程度
$H_{(ij)}$	覆盖衰减函数
u_{ij}	治安不稳定系数

注：其他未列出的符号以第一次出现时的解释为准。

4 问题一的模型建立与求解

4.1 问题分析

针对问题一，拟通过最短路径算法建立每个站点之间的距离矩阵，分析每个站点之间的道路类别带入算出每两个节点之间的通行时间矩阵。通过时间矩阵建立基于通行时间的选址优化模型。

问题中并未对派出所的选址范围进行规定，但 Zakaria 等人在《Hybrid Genetic Algorithm for Improving Fault Localization》中指出，只有当需求密度很大，或需求区域面积相对整个决策区域面积很小时，离散化模型的误差才能被控制在一个较低的水平。为更全面的分析问题，本文在离散模型的基础上对连续选址的情况也进行讨论[1]。

不同于一般的连续选址问题，本题的关注点在于如何对连续的解空间进行约束，即需要将选址的解空间固定在已知的可行道路上。

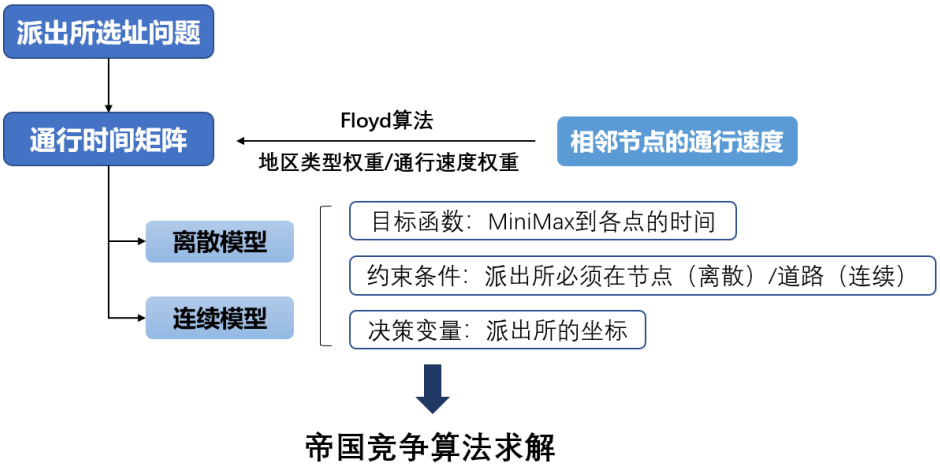


图 1　问题一思路图

4.2 数据预处理

将附件一图中的矩阵和道路信息用矩阵的形式进行表示，建立一个节点 -速度的矩阵示意图。并以节点 1 作为原点，建立平面直角坐标系，将各节点以坐标的形式进行表示，如图2：

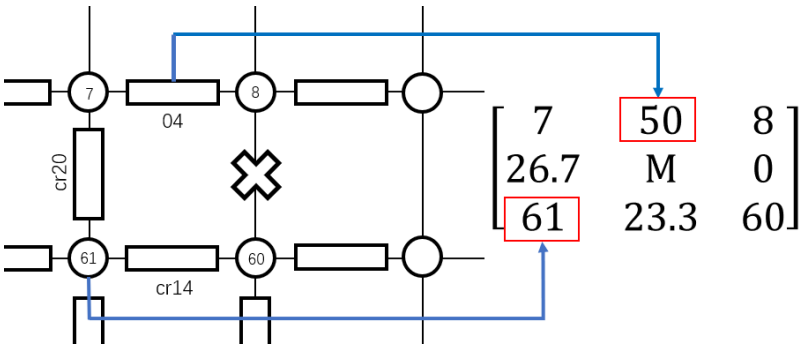


图 2　节点 -速度矩阵示意图

图中两个节点之间的数值表示两个节点之间道路所允许的最高速度，M 表示无意义点，0 表示为两个节点之间不存在实际意义的道路，无法直接通行。

4.2.1 基于时段的通行速度权重

在查阅相关资料后，可以发现在一天的不同时段下，犯罪率会有着明显的差异。于是可以依据不同时段犯罪率的差异，将一天中三段不同时间的峰值速度加权平均，最后得到一个一天中一段道路的平均最高峰值速度，

$$V = w_1v_1 + w_2v_2 + w_3v_3$$

. 其中 w_1, w_2, w_3 分别为三个时段所占的权重，v_1, v_2, v_3 则分别为三个时段的峰值速度，V 为这一段道路一天的平均峰值速度。权重的取值如表1。

表 1　速度权值取值表

	平峰期（w_1）	高峰期（w_2）	半夜（w_3）
参数	0.3	0.3	0.4

4.2.2 基于人口密度的空间权重

通过相关材料可知，一个空间中犯罪率的高低一定程度上取决于该空间中的人口密度，因此不同节点在对于派出所选址的重要程度会有差异，所以本文根据节点周围区域的类型所判断的人口密度来确定其所占权重，本文将权重设为 δ_i 为第 i 个节点所占的空间权重。权重的取值如表1[2]。

表 2　空间权值取值表

周边环境	村庄、森林、公园	居民区、商业区	工业区
权重	1	1.6	1

在后续的建模中，需要将连续分布在道路上的点首先修正至关键节点上进行分析。通过上述的节点 -速度矩阵，本文通过 Floyd（弗洛伊德）算法求出了 103 个节点任意两个节点之间通行所需要的最短时间。将其设为矩阵 $T_{i,j}$ 表示为从节点 i 到节点 j 的最短时间。

基于上述权重的设定，我们将距离矩阵重新定义为加空间和速度权值的节点间最短时间矩阵:

$$T'_{i,j} = \delta_j \frac{S_{i,j}}{\sum w_i v_i}. \tag{1}$$

4.3 基于最短时间的派出所选址优化模型

由于题目中未对派出所的选址范围进行规定，本文从离散模型和连续模型两个角度进行分析。

4.3.1 离散选址模型（P-Center 模型）

若派出所的选址被规定在附件的 103 个节点上，则可引入 0-1 变量表示该点是否为派出所选址：

$$x_i = \begin{cases} 1, \text{第 i 个节点被选为派出所建址} \\ 0, \text{第 i 个节点未被选为派出所建址} \end{cases} \tag{2}$$

显然，派出所的选址只能有一个，故有：

$$\sum_{i=1}^{103} x_i = 1 \tag{3}$$

选用 P-Center 模型中的 MiniMax 目标函数，可得总模型为：

$$Z = \min \max_{x,y} (T'_{ij});$$

$$s.t. \sum_{i=1}^{103} x_i = 1.$$

4.3.2 连续选址模型

决策变量

派出所选址的坐标：$(x, y) \quad x, y \in [0, 5].$

约束条件

1) 派出所数量限制

$$\begin{cases} P(x, y) \le x_i, \quad i = 1, 2, ..., 103; \\ \sum_{i=1}^{103} x_i = 1. \end{cases} \tag{4}$$

其中，x_i 为 0-1 变量，含义与上节相同

2) 道路约束

由于交通以及管理原因，派出所不能够建设在小区、商业区内部，所以仍应将连续的选址空间约束至附件所示的若干条道路上。通过归纳可得，道路上的横坐标点与

纵坐标中之多有一个点为小数，故可构建约束：

$$\begin{cases} x - \lfloor x \rfloor = u_x; \\ y - \lfloor y \rfloor = u_y; \\ \lceil u_x \rceil + \lceil u_y \rceil \leq 1. \end{cases} \tag{5}$$

此外，备选点所在的道路必须是可通行的，因此将坐标点映射至数据预处理的通行速度矩阵中，要求通行速度必须大于 0：

$$V[(\lceil x \rceil, \lceil y \rceil), (\lfloor x \rfloor, \lfloor y \rfloor)] > 0.$$

3) 小数坐标修正

在数据预处理的工作中，本文已经获取了任意两个重要节点之间的时间，但是对于道路上的任意一点，首先需要将其修正至重要节点处，再对通行时间进行运算。

$$\begin{cases} y_1 + y_2 = 1; \\ \delta \dfrac{\lceil x \rceil - x + \lceil y \rceil - y}{\sum_{i=1}^{3} w_i v_i} = t_1; \\ \delta \dfrac{\lfloor x \rfloor - x + \lfloor y \rfloor - y}{\sum_{i=1}^{3} w_i v_i} = t_2. \end{cases} \tag{6}$$

其中，t_1 和 t_2 分别为两种不同方案下的修正时间。要求修正点处为附件上可行的重要节点：

$$\begin{cases} P(\lceil x \rceil, \lceil y \rceil) \leq y_1; \\ P(\lfloor x \rfloor, \lfloor y \rfloor) \leq y_2. \end{cases} \tag{7}$$

因此总通行时间可以被修改为:

$$T'_{ij} = \sum_{k=1}^{2} t_k x_k + T_{ij}. \tag{8}$$

目标函数

考虑派出所需要兼顾公平性与有效性原则，本文选用了 Minimax 形式的目标函数作为模型的目标函数:

$$Z = \min \max_{x,y} (T'_{ij}). \tag{9}$$

模型总述

$$
\begin{aligned}
&Z = \min\max_{x,y}(T'_{ij});\\
&s.t.\begin{cases}
P(x,y) \le x_i, \quad i = 1,2,...,103\\
\sum_{i=1}^{103} x_i = 1;\\
x - \lfloor x \rfloor = u_x; y - \lfloor y \rfloor = u_y;\\
\lceil u_x \rceil + \lceil u_y \rceil \le 1;\\
y_1 + y_2 = 1;\\
\delta\frac{\lceil x \rceil - x + \lceil y \rceil - y}{\sum_{i=1}^{3} w_i v_i} = t_j, \quad j = 1,2;\\
P(\lceil x \rceil, \lceil y \rceil) \le y_1; P(\lfloor x \rfloor, \lfloor y \rfloor) \le y_2;\\
T'_{ij} = \sum_{k=1}^{2} t_k x_k + T_{ij}.
\end{cases}
\end{aligned}
\tag{10}
$$

4.4 派出所选址问题的模型求解

4.4.1 基于帝国竞争算法求解派出所选址问题

1) 初始化帝国

首先随机生成若干解，每个解对应一个国家。解 (x, y) 中分量分别表示派出所选址的横纵坐标, 解的优劣根据上述模型中的 MiniMax 拉式目标函数决定, 解越优, 对应的国家势力越大。

将目标函数最小的几个国家定义为殖民国家, 通过 $C_k = max\{c_l\} - c_k$ 对目标函数值归一化后，根据其势力大小为每个殖民国家分配殖民地, 构建若干初始帝国：

$$
Emipre_i = \{(x_I, y_I); (x_{c1}, y_{c1}), (x_{c2}, y_{c2})...\}. \tag{11}
$$

其中，(x_I, y_I) 为殖民国家对应的坐标。

2) 帝国同化

同化过程可以描述为殖民地沿其自身和殖民国家的连线方向向殖民国家随机移动一段距离 $D \sim U(0, \beta \times d)$，同时为保证解的多样性，引入偏角 θ。对于二维坐标，进一步迭代的公式为：

$$
(x + D\cos\theta, y + D\sin\theta). \tag{12}
$$

其中 d 为殖民地和殖民国家之间的距离；$1 < \beta \le 2$, 设置 $\beta > 1$ 是为了使殖民地向殖民国家方向移动。

另外，由于本问题的特殊性质使得需要对小数坐标进行修正，具体计算过程已经在上节中阐明，示意图如下：

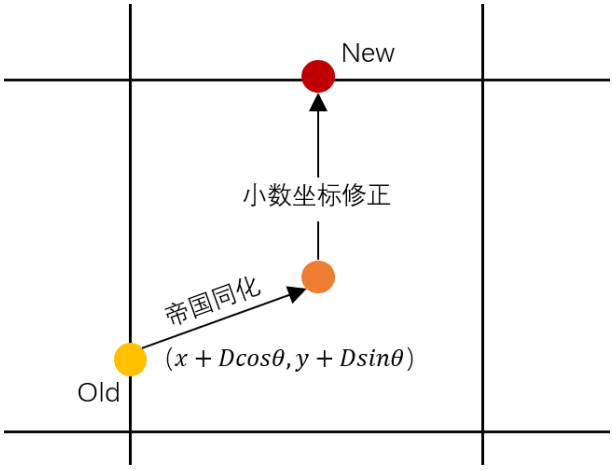


图 3　帝国同化示意图

3) 帝国革命

在迭代过程中对于每一个帝国随机选取一定数量的殖民地，将其坐标随机改变。

4) 殖民国家更新

迭代过后，在一个帝国内选择结果最优的国家作为新的殖民国家。

5) 帝国竞争

根据帝国的总成本，利用轮盘赌算法，将最弱帝国的最弱国家按照帝国势力重新划分，每个国家能够获得最弱帝国的概率与其势力有关:

$$P(k) = \frac{C_k + \sum_i C_{ik}/N.C.}{\sum_k (C_k + \sum_i C_{ik}/N.C.)}. \tag{13}$$

6) 帝国消亡

如果一个帝国内无任何成员，则直接剔除这个帝国。

以下是帝国竞争算法伪代码：

Algorithm 1: Function ImperialistCompetitveAlgorithm(x)

Initialize /* 初始化：设置多个初始国家解 $X_n, n = 1, ..., n_max$。从中选取 N 个最好质量的解使其作为 N 个帝国的殖民国家，$X_1, X_2, ..., X_N$，为每个殖民国家分配殖民国家势力比例数量的殖民地得到 N 个帝国 $E_1, E_2, ..., E_N$ */

iteration = 1; **Repeat for** $i \in [1, N]$ **do**

    EmpireAssimilate(E_i) /* 帝国同化 */

    ColoniesRevolve(E_i) /* 殖民地革命 */

    EmpirePosses(E_i) /* 殖民国家替换 */

    CaculateNewCost(E_i) /* 重新计算国家权力 */

end

EmpireUnite($E_1, E_2, \ldots, E_N$) /* 合并相似帝国 */

EmpireCompetition($E_1, E_2, ..., E_N$) /* 帝国竞争，分配殖民地 */

iteration ← iteration+1; 进入下一次迭代

until 只有一个帝国剩下/达到迭代次数。

通过 MATLAB 编程实现了帝国竞争算法。图4为帝国竞争算法迭代结果图：

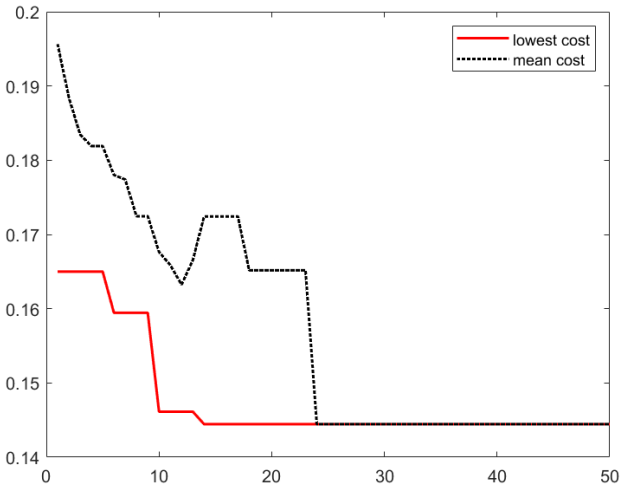


图 4　帝国竞争迭代图

通过帝国竞争算法，最后得出来在离散情况下和连续情况下的最优选址方案和目标函数的最小值。如表3：

表 3　结果呈现

	离散情况	连续情况
选址坐标	(0,0)	(0,-0.4178)
目标函数/h	0.1556	0.1532

4.4.2 算法对比

本文将传统的粒子群算法与帝国竞争算法进行求解对比。迭代结果如图5

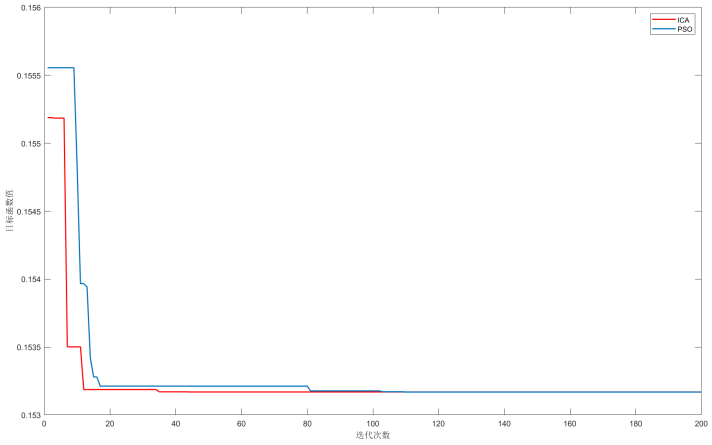


图 5　算法对比图

(其中蓝色线为传统粒子群算法迭代曲线，红色线为帝国竞争算法迭代线，两者相对比可以明显看出帝国竞争算法收敛速度更快。)

4.5 结果分析

通过分别对离散型和连续性的派出所选址问题求解，可以看到连续问题的求解决策空间更大，求解出的结果优于离散型。

根据公安部发布的《公安派出所建设标准》（2016 修订版）第十二条：公安派出所的选址，宜处于辖区中心区域，交通便捷，临靠城市道路，便于出警或群众求助，有较好的市政设施条件 [3]。本文通过帝国竞争算法求解出了派出所的最优地点在节点 7 处，如下图6：

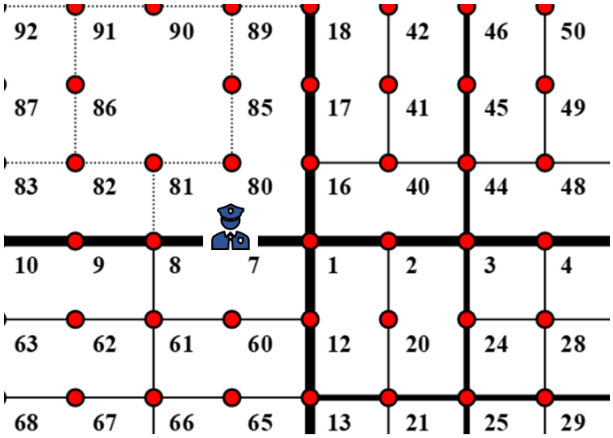


图 6　问题一结果

通过查询附件一中的地图我们发现此地点在大道西路上。符合《公安派出所建设标准》所规定的交通便利，临靠城市道路的情况。

而如果不依据人口密度的空间权重直接按照最短时间进行求解，则最佳选址地点则在节点 89 上。将此点与《公安派出所建设标准》的建设指导意见相比较则会发现此点建设在乡村公路上不符合靠近城市道路的原则，据此可以反映出权重的重要性。

4.6 灵敏度分析

随着时间的变化，城市人口密度的分布也会发生变化，于是针对基于人口密度的空间权重做灵敏度分析，对城市人口密度对应的空间权重做 ±5% 的变化，并比较产生的此变化后对结果的影响。图7 为权重变化后的最优目标结果图。

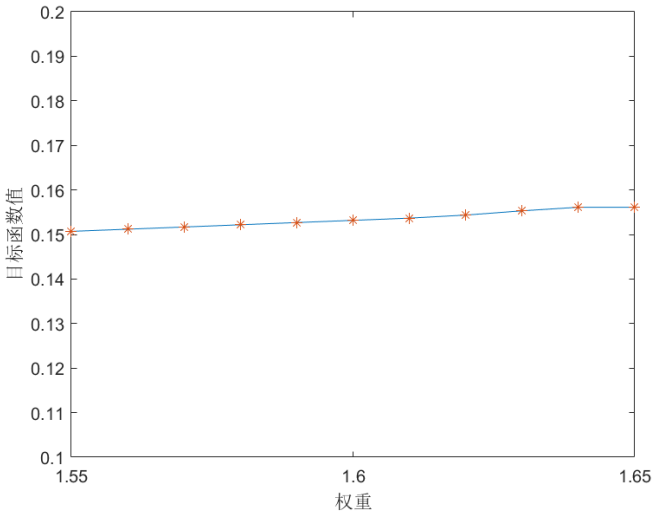


图 7　灵敏度分析结果

从灵敏度分析的结果图看来，随着城市人口密度对应的权重发生波动时，目标函数的变化并不明显。最后得到的派出所地址也在很小的范围内移动，由此可以说明模型具有较强的稳定性。

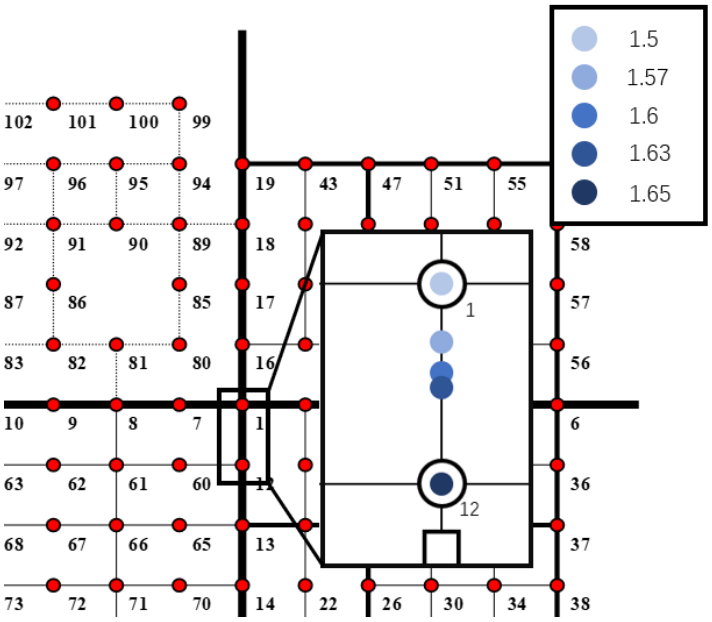


图 8　权重变化下选址变化结果

从图8中可以发现在人口密度对应的空间权重发生变化时，派出所的最佳选址地点集中在节点 1 和节点 12 之间，且变化范围很小，这说明部分参数在一定变化范围内，原始最优解仍然保持其良好性质。故可以认为此选址位置比较可靠，具有一定抵抗参数波动的能力。

5 问题二模型的建立与求解

5.1 问题分析

与前文给出了基于人口密度的空间权重类似，本题需要根据附件中提供的犯罪数据重新建立犯罪恶劣指数。同时，根据核函数的基本思想，若派出所出警到达目标地点的时间越短，能够减少更多的损失，因此我们还需要引入覆盖衰减函数的并通过速节点 -速度矩阵按问题一的方式建立基于通行时间矩阵优化模型。为保证解的效果较优，本文仅对连续选址的情况进行讨论。

注意到附件中的数据为前十年犯罪数据的统计，而实际上未来发生犯罪事件的次数以及恶劣程度都为不确定的。容易将该情况归纳为事前分析，因此可以在给定指标分布的前提下，进行随机规划。

另外，在随机规划中又可分为期望模型、机会约束模型等。因为期望模型得到结果很可能是不可行的，故选择了机会约束模型对此问题进行分析 [4]。

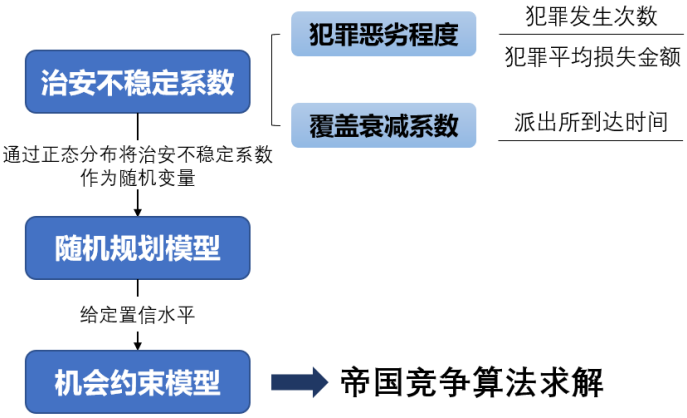


图 9　问题二思路图

5.2 数据预处理

5.2.1 治安不稳定系数

由于每次违法损失的金额数存在一定的偶然性，所以损失的金额不能够完全等比例的反映违法的恶劣程度，所以本文认为损失金额对反映违法恶劣程度的效用存在边际递减的关系。而违法次数则反映了人们对此处治安程度的预期，所以本文认为违法次数反映违法恶劣程度的效用存在边际递增的关系。基于此关系，我们可以引入每个节点的犯罪恶劣程度 $u_j(m,n)$，m 表示节点 j 总损失金额数，n 表示节点 i 犯罪次数：

$$u_j(m,n) = [lg(nm+1)+1] \times (n+1) \tag{14}$$

同时本文考虑派出所出警的及时性，本文认为当违法事件发生之后，派出所在一

定时间内赶到就能减少一定损失，基于此本文引入覆盖衰减函数 [5]$H_{(ij)}$：

$$H(t_{ij}) = \begin{cases} 2, & t_{ij} < D^{\Lambda}; \\ [1 - \dfrac{t_{ij} - D^{\Lambda}}{\max t_{ij} - D^{\Lambda}}]^{\alpha} + 1, & t_{ij} \geq D^{\Lambda} \end{cases} \tag{15}$$

其中，t_{ij} 为派出所 i 到违法节点 j 所需时间，D^{Λ} 为衰减阈值。

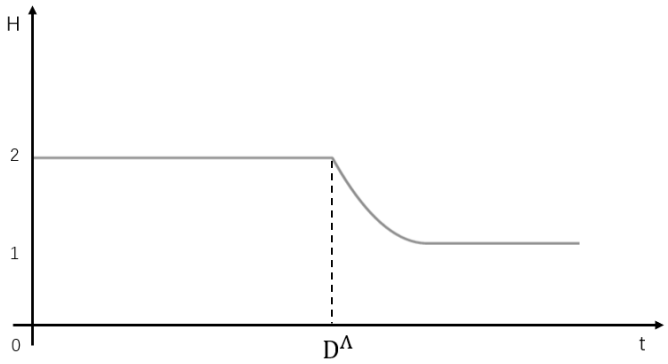


图 10　覆盖衰减函数图

此后我们基于犯罪恶劣程度和覆盖衰减函数可以得到计算治安不稳定系数 $\xi_{(x,y),j}$：

$$\xi_{(x,y),j} = \frac{u_j}{H_{(ij)}} \tag{16}$$

图15是每个节点的不稳定系数散点热力图：

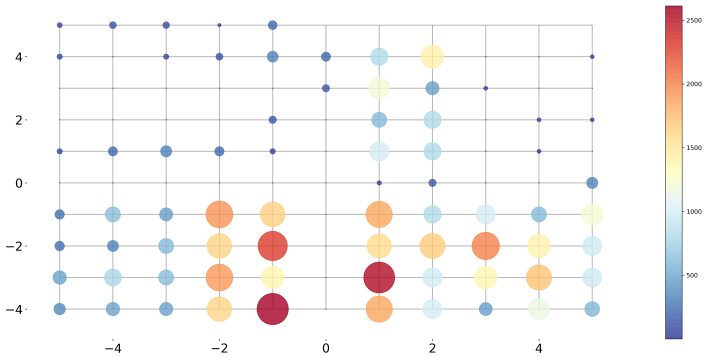


图 11　治安不稳定系数图

（图中每个节点圆越大颜色越深的地方其不稳定系数越高）

5.3 建立派出所选址的机会约束规划模型

1) 带有随机参数的数学规划模型

决策变量

派出所选址的坐标：$(x,y)\quad x\in[-4,5],y\in[-5,5].$

约束条件

针对本问，拟采取连续的选址模型进行分析。因此，除上文中对于治安不稳定系数的计算方法外，其余约束都与问题一中的连续选址模型相同。

由于过去十年内犯罪的数据仅可作为先验数据，故在对派出所进行选址时，需要考虑未来犯罪情况，建立含有随机变量的约束。认为治安不稳定系数的服从正态分布：

$$\xi_{(x,y),j}\sim\ N(\mu_j,\sigma_j), \tag{17}$$

其中，μ_j,σ_j 分别表示对于 j 点处治安不稳定系数服从分布的均值和方差，由于犯罪的恶劣程度具有一定空间聚集性，故认为每个点的犯罪情况都与附近四个点相关，故参数的估计将邻近的四个点及其本身作为样本点，通过最大似然估计法对其进行计算。

需要强调，对于离散分布在道路上的派出所选址可行解，首先需要将其修正至离散的关键节点上，故在此仅对两个离散点之间的治安不稳定系数进行了计算，但通过约束和相关计算，可以转化为连续情况下的执法成本。

目标函数

$$\min\max_{(x,y),j}\xi_{(x,y),j}. \tag{18}$$

2) 建立机会约束模型

机会约束模型采用的基本原则为：允许决策在一定程度上不满足约束条件，但是应该使得约束条件成立的概率不小于置信水平 α，针对本题，随机参数实际上仅在目标函数里出现，故对其进行修改，即用式（19）代替（25）：

$$\begin{aligned}&\min_{(x,y),j}f,\\ s.t.\quad &P\{\max_{(x,y)j}(\xi_{(x,y),j})<f\}>\alpha.\end{aligned} \tag{19}$$

其中，本文选取 $\alpha=90\%$。从目标函数最小化的角度来看，即为在确保置信水平为 α 的条件下，所取 f 的最小值。

模型总述

$$
\begin{aligned}
&Z = \min f;\\
&s.t.\begin{cases}
P(x,y) \le x_i, \quad i = 1,2,...,103;\\
\sum_{i=1}^{103} x_i = 1;\\
x - \lfloor x \rfloor = u_x; y - \lfloor y \rfloor = u_y;\\
\lceil u_x \rceil + \lceil u_y \rceil \le 1;\\
y_1 + y_2 = 1;\\
\delta \frac{\lceil x \rceil - x + \lceil y \rceil - y}{\sum_{i=1}^{3} w_i v_i} = t_j, \quad j = 1,2;\\
P(\lceil x \rceil, \lceil y \rceil) \le y_1; P(\lfloor x \rfloor, \lfloor y \rfloor) \le y_2;\\
P\{\max_{(x,y)j}(\xi_{(x,y),j}) < f\} > \alpha.
\end{cases}
\end{aligned}
\tag{20}
$$

5.4 模型求解

此模型与问题一的模型类似，所以本文仍然采用上文所提的帝国竞争算法进行求解。图12为问题二的求解迭代图:

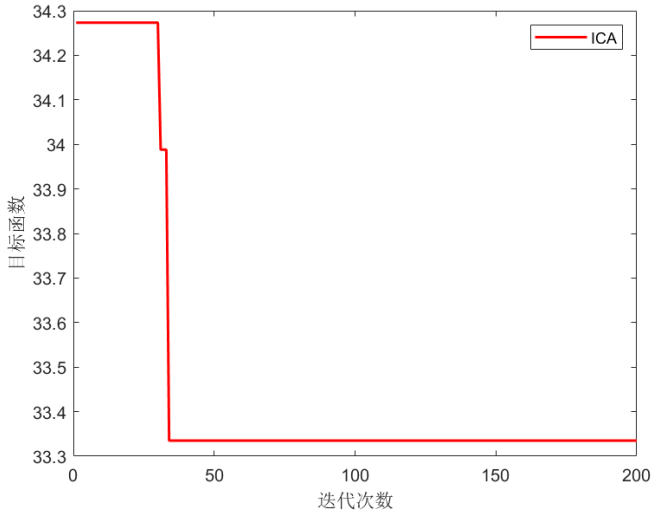


图 12　第二问求解迭代图

图13为派出所选址的最优点示意图。在地图中位于节点 70 处。

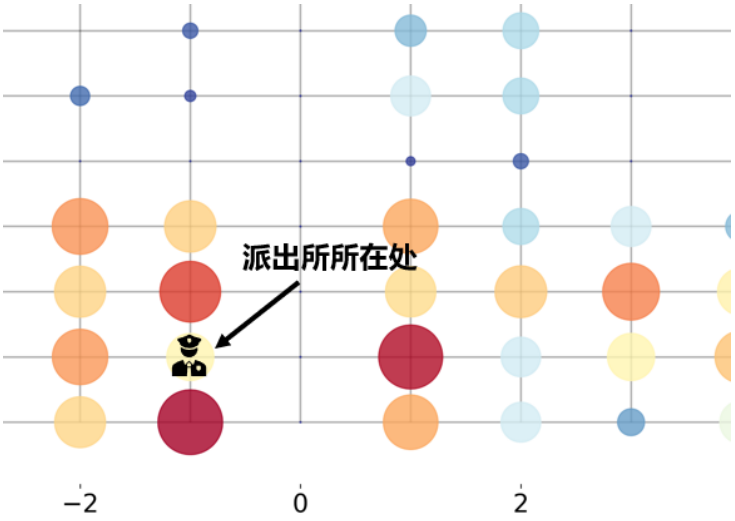


图 13　问题二派出所选址示意

通过图13我们可以发现派出所选址位置主要在治安不稳定系数较高的节点，对这些节点的治理效果会较好。与问题一求得的派出所选址地点相比较，此派出所的地点更加靠近居民区和商业区，这是由于此区域人口密度较高导致的犯罪率和治安不稳定系数较高。所以在居民区和商业区附近修建派出所可以更有效得去处理违法事件。

5.5 灵敏度分析

由于派出所的覆盖衰减系数对，派出所的选址会对目标函数造成影响，进而会对派出所的选址造成影响。所以我们有必要知道覆盖衰减系数对最终目标函数的影响和最后选址的影响。图14为覆盖衰减函数阈值上下波动 5% 后对目标函数的影响。

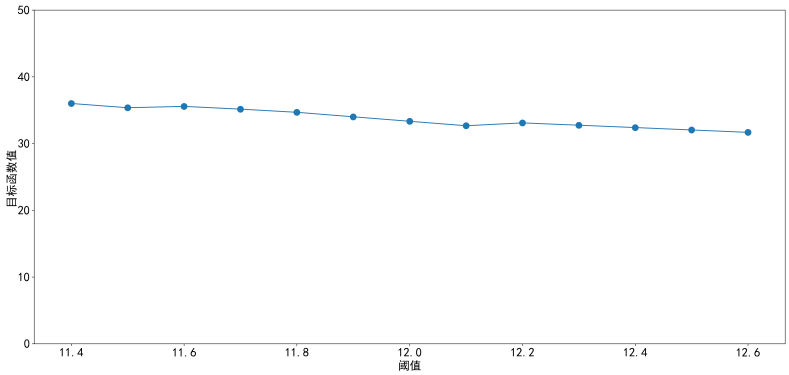


图 14　第二问灵敏度分析结果

图中可以看出随着覆盖衰减函数阈值的变化，随着阈值的增大治安不稳定系数略有下降且下降幅度在 8% 内，符合实际条件下可追回覆盖面积的增大而减小的趋势，且幅度稳定。

6 问题三模型建立与求解

6.1 问题分析

本题需要在问题二的基础上，新设一个派出所，使得两个派出所能够达到最大执法效益。我们注意到对于现实生活中，派出所管辖区域的划分往往意味着一个事件仅由一个派出所进行处理；但派出所分布密集的区域又能够实现多个派出所之间的灵活调度，达到协同执法的效果。由此，拟分别对应两种情况，分别建立基于区域划分的选择指标和基于协同执法的合作指标，并引入不确定性，构建随机规划模型对问题进行分析 [6]。

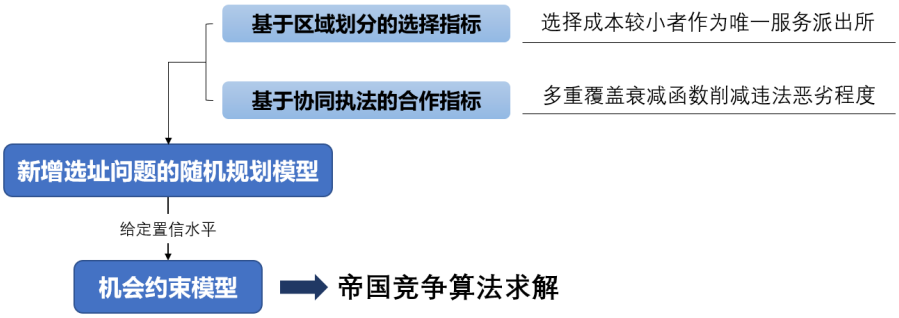


图 15　问题三思路图

6.2 建立新增派出所选址问题的随机规划模型

1）基于区域划分的选择指标

对于大部分违法事件，需要强调其时效性，因此在具体执法的过程中，往往由执法成本较小的派出所对该节点的事件进行处理。故在新建派出所后，每个节点实际的治安不稳定系数为：

$$\mu_{ij} = \min(\xi_{0j}, \xi_{1j}). \tag{21}$$

其中，ξ_{1j} 表示新建派出所 (x_1, y_1) 负责 j 节点时的治安不稳定系数；xi_{0j} 则表示原始派出所负责 j 节点时的治安不稳定系数，根据第二问，原始派出所为 70 号点，对应坐标为 $(-1, -3)$。ξ 具体的计算方法见上一章。

2）基于协同执法的合作指标

对于一部分规模较大的突发事件，以及日常生活中派出所起到的震慑作用，都可以在附近派出所的数量中有所体现，即表现为派出所协同执法的效用 [7]。因此，本文通过多重覆盖衰减函数对节点处的治安不稳定系数进行修正：

$$\tau_{ij} = \frac{[\lg(n_j m_j + 1) + 1](1 + n_j)}{H(T_{0j})H(T_{1j})}. \tag{22}$$

其中，T_{0j} 和 T_{1j} 分别表示原始派出所和新增派出所到达执法地点 j 的时间。上式通过将犯罪恶劣程度除以两个覆盖衰减函数，在数学形式上体现了协同执法的作用。

结合上两节，实际过程中派出所之间的合作与选择时相互矛盾的，具体分析时需要考虑到决策者的偏好，故将两个指标加权求和：

$$T_j = \alpha\tau_j + (1-\alpha)\mu_j. \tag{23}$$

其中，α 为赋予的权重，其值越大表明越看重派出所之间的协同执法。

3）新增选址问题机会约束模型

本问题的基本约束与问题二中相同，不同之处在于本文的成本指标以第二问的治安不稳定系数作为基础，转化为同时表达派出所的选择和合作的指标。考虑到第三问中仍具有不确定性，故需要对治安不稳定系数进行转化：

$$R_j = \alpha\frac{\xi_{0j}}{H(T_{1j})} + (1-\alpha)\min(\xi_{0j}, \xi_{1j}). \tag{24}$$

针对上式中随机变量所服从的分布，估计方式与问题二中相同，仍以中心点附近的四个节点上的数据作为样本，通过最大似然估计得到正态分布的参数。

将上式通过 $\beta = 90\%$ 的置信水平转化为确定的约束条件：

$$P\{\max_{(x_1,y_1)j}(R_j) < f\} > \beta. \tag{25}$$

模型总述

$$\begin{aligned} &Z = \min(f); \\ &s.t. \begin{cases} P(x,y) \le x_i, \quad i = 1,2,...,103; \\ \sum_{i=1}^{103} x_i = 1; \\ x - \lfloor x \rfloor = u_x; y - \lfloor y \rfloor = u_y; \\ \lceil u_x \rceil + \lceil u_y \rceil \le 1 \\ y_1 + y_2 = 1; \\ \delta\frac{\lceil x \rceil - x + \lceil y \rceil - y}{\sum_{i=1}^{3} w_i v_i} = t_j, \quad j = 1,2; \\ P(\lceil x \rceil, \lceil y \rceil) \le y_1; P(\lfloor x \rfloor, \lfloor y \rfloor) \le y_2; \\ P\{\max_{(x_1,y_1)j}(R_j) < f\} > \beta. \end{cases} \end{aligned} \tag{26}$$

6.3 模型求解

通过帝国竞争算法求解出第二个派出所的最佳位置为节点 27 处。表4为每个节点所属辖区示意。图16为二派出所的辖区图。

表 4　各节点所属辖区（节选）

所属派出所编号		所属派出所编号	
节点 1	1	节点 10	1
节点 2	2	节点 11	1
节点 3	2	节点 12	1
节点 4	2	节点 13	1
节点 5	2	节点 14	1
节点 6	2	节点 15	2
节点 7	1	节点 16	1
节点 8	1	节点 17	1
节点 9	1	节点 18	1

（其中编号为 1 归属于原有派出所，编号为 2 归属于新建派出所）

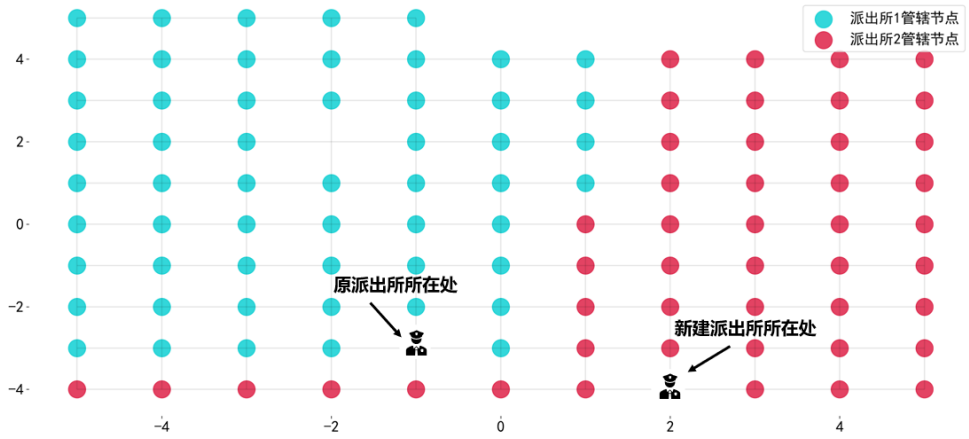


图 16　辖区规划图

其中蓝色的节点由在节点 70 处的派出所管辖，红色的节点由节点 23 处的派出所管辖。

可以发现个各派出所的辖区的区分较为明显，主要呈现在左右分布的态势。两个派出所均选址于居民区，并设置在城市道路上。

6.4 灵敏度分析

由于两所派出所的既有基于区域划分的选择情况，又有相互合作的协同执法情况，因此本文两者之间权重的变化会对最终目标函数产生影响，为确定合适的权重需要对其进行灵敏度分析。图7为不同权值下目标函数的变化趋势图：

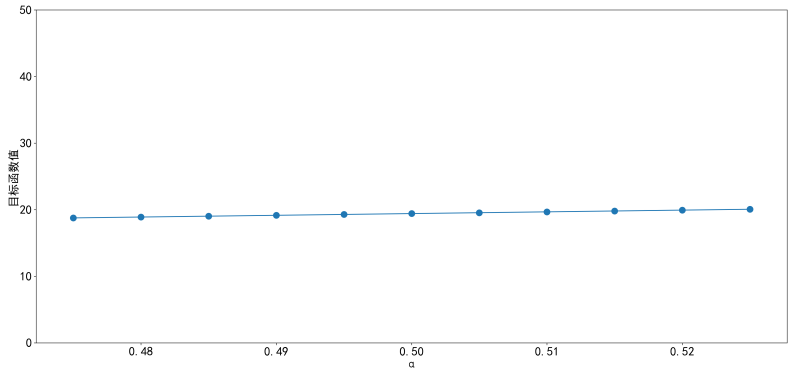


图 17　第三问灵敏度示意图

可以看出随着权值的变化，目标函数的变化波动不超过 3.3%。说明权值的选择对新派出所的选址影响不大，证明了模型的可靠性。

7 模型总结与评价

7.1 模型优点

1. 考虑到选址问题必须在观测到实际犯罪情况前进行实施，本文通过运用随机规划的相关理论，建立了机会约束规划模型，使结果具有更广泛的适应性。

2. 求解问题时采用了帝国竞争算法，该算法相较于传统的粒子群算法收敛速度更快，收敛精度更高。

3. 从人口密度、覆盖效益衰减以及派出所之间的合作关系等角度建立了多个反映执法效益的指标，并参考《公安派出所建设标准》的相关要求核实派出所选址，使模型更符合实际情况。

7.2 模型缺点

1. 未考虑犯罪事件同时发生导致单一派出所警力不足的情况

7.3 模型改进

由于附件中的数据仅为过去十年的数据，所以需要考虑到未来发展的不确定性。派出所作为保障人民安全的重要基层组织，在选址问题中应考虑最坏的情况 (worst-case) 进行分析。若时间允许，可以建立相应的鲁棒优化模型，从最坏情况的角度对不确定因素进行分析。

参考文献

[1] 张喆. 基于连续需求的区域最大覆盖设施选址问题的研究 [D]. 北京交通大学, 2019.

[2] 王星. 随机、容错和厌恶型设施选址的算法研究 [D]. 天津大学, 2012.

[3] 王星, 徐大川. 带次模惩罚和随机需求的设施选址问题 [J]. 运筹学学报, 2013, 17(2):1-9.

[4] 蒋晨琛, 霍宏涛, 刘克俭, 等. 空间数据驱动的 B 市主城区犯罪时空分布及其影响因素分析 [J]. 科学技术与工程, 2019, 019(026):384-394.

[5] 肖俊华, 侯云先. 大规模突发事件应急设施选址模型及算法 [J]. 计算机工程与应用, 2013.

[6] 王璨璨. 连续设施选址模型的快速算法研究 [D]. 南京航空航天大学, 2012.

[7] 王惠珠, 周建勤. 基于不确定需求的铁路救援基地选址问题研究 [J]. 北京交通大学学报 (社会科学版),2020,19(03):100-107.

附录 A 代码

A.1 帝国竞争 matlab 源程序

```
close all
clc; clear
global dots
global dismat
global path_var
load dots
load('../dismat1')
load path_var
%% Problem Statement
ProblemParams.CostFuncName = 'BenchmarkFunction'; % You should state the
    name of your cost function here.
ProblemParams.CostFuncExtraParams = '';
ProblemParams.NPar = 2;                    % Number of optimization
    variables of your objective function. "NPar" is the dimention of the
    optimization problem.
ProblemParams.VarMin = -5;                  % Lower limit of the
    optimization parameters. You can state the limit in two ways. 1) 2)
ProblemParams.VarMax = 5;                % Lower limit of the optimization
    parameters. You can state the limit in two ways. 1) 2)

% Modifying the size of VarMin and VarMax to have a general form
if numel(ProblemParams.VarMin)==1
   ProblemParams.VarMin=repmat(ProblemParams.VarMin,1,ProblemParams.NPar);
   ProblemParams.VarMax=repmat(ProblemParams.VarMax,1,ProblemParams.NPar);
end

ProblemParams.SearchSpaceSize = ProblemParams.VarMax - ProblemParams.VarMin;

%% Algorithmic Parameter Setting
AlgorithmParams.NumOfCountries = 50;        % Number of initial countries.
AlgorithmParams.NumOfInitialImperialists = 10; % Number of Initial
    Imperialists.
AlgorithmParams.NumOfAllColonies = AlgorithmParams.NumOfCountries -
    AlgorithmParams.NumOfInitialImperialists;
AlgorithmParams.NumOfDecades = 200;
AlgorithmParams.RevolutionRate = 0.3;       % Revolution is the process in
    which the socio-political characteristics of a country change suddenly.
AlgorithmParams.AssimilationCoefficient = 2; % In the original paper
    assimilation coefficient is shown by "beta".
```

```
AlgorithmParams.AssimilationAngleCoefficient = .5; % In the original paper
    assimilation angle coefficient is shown by "gama".
AlgorithmParams.Zeta = 0.02;                % Total Cost of Empire = Cost
    of Imperialist + Zeta * mean(Cost of All Colonies);
AlgorithmParams.DampRatio = 0.99;
AlgorithmParams.StopIfJustOneEmpire = false;  % Use "true" to stop the
    algorithm when just one empire is remaining. Use "false" to continue the
    algorithm.
AlgorithmParams.UnitingThreshold = 0.02;     % The percent of Search Space
    Size, which enables the uniting process of two Empires.

zarib = 1.05;                   % **** Zarib is used to prevent the weakest
    impire to have a probability equal to zero
alpha = 0.1;                    % **** alpha is a number in the interval of
    [0 1] but alpha<<1. alpha denotes the importance of mean minimum compare
    to the global mimimum.

%% Display Setting
DisplayParams.PlotEmpires = false; % "true" to plot. "false" to cancel
    ploting.
if DisplayParams.PlotEmpires
   DisplayParams.EmpiresFigureHandle = figure('Name','Plot of
       Empires','NumberTitle','off');
   DisplayParams.EmpiresAxisHandle = axes;
end

DisplayParams.PlotCost = true; % "true" to plot. "false"
if DisplayParams.PlotCost
   DisplayParams.CostFigureHandle = figure('Name','Plot of Minimum and Mean
       Costs','NumberTitle','off');
   DisplayParams.CostAxisHandle = axes;
end

ColorMatrix = [1 0  0  ; 0 1  0   ; 0  0 1   ; 1  1  0  ; 1  0 1   ; 0 1  1
      ; 1 1 1     ;
           0.5 0.5 0.5; 0 0.5 0.5 ; 0.5 0 0.5 ; 0.5 0.5 0 ; 0.5 0 0 ; 0 0.5
              0 ; 0 0 0.5 ;
           1   0.5 1 ; 0.1*[1 1 1]; 0.2*[1 1 1]; 0.3*[1 1 1]; 0.4*[1 1 1];
              0.5*[1 1 1]; 0.6*[1 1 1]];
DisplayParams.ColorMatrix = [ColorMatrix ; sqrt(ColorMatrix)];

DisplayParams.AxisMargin.Min = ProblemParams.VarMin;
DisplayParams.AxisMargin.Max = ProblemParams.VarMax;
```

```
%% Creation of Initial Empires
InitialCountries = GenerateNewCountry(AlgorithmParams.NumOfCountries ,
    ProblemParams);

% Calculates the cost of each country. The less the cost is, the more is
    the power.
if isempty(ProblemParams.CostFuncExtraParams)
   InitialCost = feval(ProblemParams.CostFuncName,InitialCountries);
else
   InitialCost =
       feval(ProblemParams.CostFuncName,InitialCountries,ProblemParams.CostFuncExtraParams
end
[InitialCost,SortInd] = sort(InitialCost);                % Sort the cost
    in assending order. The best countries will be in higher places
InitialCountries = InitialCountries(SortInd,:);           % Sort the
    population with respect to their cost.
InitialCost = InitialCost';
Empires =
    CreateInitialEmpires(InitialCountries,InitialCost,AlgorithmParams,
    ProblemParams);

%% Main Loop
MinimumCost = repmat(nan,AlgorithmParams.NumOfDecades,1);
MeanCost = repmat(nan,AlgorithmParams.NumOfDecades,1);

if DisplayParams.PlotCost
   axes(DisplayParams.CostAxisHandle);
   if any(findall(0)==DisplayParams.CostFigureHandle)
      h_MinCostPlot=plot(MinimumCost,'r','LineWidth',1.5,'YDataSource','MinimumCost');
      hold on;
%
    h_MeanCostPlot=plot(MeanCost,'k:','LineWidth',1.5,'YDataSource','MeanCost');
      hold off;
      pause(0.05);
   end
end

for Decade = 1:AlgorithmParams.NumOfDecades
   AlgorithmParams.RevolutionRate = AlgorithmParams.DampRatio *
       AlgorithmParams.RevolutionRate;

   Remained = AlgorithmParams.NumOfDecades - Decade
```

```
    for ii = 1:numel(Empires)
        %% Assimilation; Movement of Colonies Toward Imperialists
            (Assimilation Policy)
        Empires(ii) =
            AssimilateColonies(Empires(ii),AlgorithmParams,ProblemParams);

        %% Revolution; A Sudden Change in the Socio-Political Characteristics
        % 该殖民地改革操作可能导致势力较强的殖民地丢失，导致寻优精度降低
        Empires(ii) =
            RevolveColonies(Empires(ii),AlgorithmParams,ProblemParams);

        %% New Cost Evaluation
        if isempty(ProblemParams.CostFuncExtraParams)
            Empires(ii).ColoniesCost =
                feval(ProblemParams.CostFuncName,Empires(ii).ColoniesPosition);
            Empires(ii).ColoniesCost = Empires(ii).ColoniesCost';
        else
            Empires(ii).ColoniesCost =
                feval(ProblemParams.CostFuncName,Empires(ii).ColoniesPosition,ProblemParams.Co
        end

        %% Empire Possession (****** Power Possession, Empire Possession)
        Empires(ii) = PossesEmpire(Empires(ii));

        %% Computation of Total Cost for Empires
        Empires(ii).TotalCost = Empires(ii).ImperialistCost +
            AlgorithmParams.Zeta * mean(Empires(ii).ColoniesCost);

    end

    %% Uniting Similiar Empires
    Empires = UniteSimilarEmpires(Empires,AlgorithmParams,ProblemParams);

    %% Imperialistic Competition
    Empires = ImperialisticCompetition(Empires);

    if numel(Empires) == 1 && AlgorithmParams.StopIfJustOneEmpire
        break
    end

    %% Displaying the Results
    DisplayEmpires(Empires,AlgorithmParams,ProblemParams,DisplayParams);
```

```
    ImerialistCosts = [Empires.ImperialistCost];
    MinimumCost(Decade) = min(ImerialistCosts);
    MeanCost(Decade) = mean(ImerialistCosts);

    if DisplayParams.PlotCost
       refreshdata(h_MinCostPlot);
%         refreshdata(h_MeanCostPlot);
       drawnow;
       pause(0.01);
    end

end % End of Algorithm
MinimumCost(end)
```

A.2 粒子群算法 matlab 源程序

```
%% Particle Swarm Optimization
clc
clear
global dots
global dismat
global path_var
load dots
load('../dismat1')
load path_var
%% Problem Statement
Npar = 2;
FN = 'BenchmarkFunction';
FunNumber = 6;
VarLow = -5;
VarHigh = 5;

%% Parameters
C1 = 1.5;
C2 = 4-C1;
Inertia = .3;
DampRatio = .95;
ParticleSize = 10;
MaxIter = 200;

GlobalBest = 0;
GlobalBestCost = 10000;
```

```
%% Initialization
GB = [];

for ii = 1:ParticleSize
   Particle{ii}.Position = rand(1,Npar) * (VarHigh - VarLow) + VarLow;
   Particle{ii}.Position = fix_coordinate(Particle{ii}.Position);
   Particle{ii}.Cost = feval(FN,Particle{ii}.Position);
   Particle{ii}.Velocity = rand(1,Npar);
   Particle{ii}.LocalBest = Particle{ii}.Position;
   Particle{ii}.LocalBestCost = Particle{ii}.Cost;
   if Particle{ii}.Cost < GlobalBestCost
      GlobalBest = Particle{ii}.Position;
      GlobalBestCost = Particle{ii}.Cost;
   end
end

%% Main Loop
for jj = 1:MaxIter
   for ii = 1:ParticleSize
      Inertia = Inertia * DampRatio;
      Particle{ii}.Velocity = rand * Inertia * Particle{ii}.Velocity + C1 *
          rand * (Particle{ii}.LocalBest - Particle{ii}.Position) + C2 *
          rand * (GlobalBest - Particle{ii}.Position);
      Particle{ii}.Position = Particle{ii}.Position + Particle{ii}.Velocity;

      Particle{ii}.Position(Particle{ii}.Position > VarHigh) = VarHigh;
      Particle{ii}.Position(Particle{ii}.Position < VarLow) = VarLow;
      Particle{ii}.Position = fix_coordinate1(Particle{ii}.Position);
      Particle{ii}.Cost = feval(FN,Particle{ii}.Position);
      if Particle{ii}.Cost < Particle{ii}.LocalBestCost
         Particle{ii}.LocalBest = Particle{ii}.Position;
         Particle{ii}.LocalBestCost = Particle{ii}.Cost;

         if Particle{ii}.Cost < GlobalBestCost
            Particle{ii}.Cost
            GlobalBest = Particle{ii}.Position;
            GlobalBestCost = Particle{ii}.Cost;
         end
      end
   end
   GB = [GB GlobalBestCost];
end
```

```
  % %% Function Plot
  % x = -10:.05:10;
  % y = -10:.05:10;
  % [X,Y] = meshgrid(x,y);
  % Z = 60 + X.^2 + Y.^2 - 30*(cos(20* X) + cos(20*Y));
  % surf(X,Y,Z)
  %%
  plot(GB,'LineWidth',1.5)
  GlobalBest
  GlobalBestCost
```

A.3 fix coordinate matlab 源程序

```
  function x = fix_coordinate(x)
global dots
for i = 1:size(x,1)
    if x(i,2) < -4
        x(i,2) = -4;
    end
    if x(i,1) > -1 && x(i,1) < 0 && x(i,2) > 3
        x(i,2) = 3;
    end
    ra = rand;
    if x(i,1) > -3 && x(i,1) < -1 && x(i,2) > 1 && x(i,2) < 3
        if ra < 0.25
            x(i,1)=-3;
        elseif ra >= 0.25 && ra < 0.5
            x(i,1)=-1;
        elseif ra >=0.5 && ra < 0.75
            x(i,2)=1;
        else
            x(i,2)=3;
        end
    end
    if x(i,1) >= 0 && x(i,1) <=5 && x(i,2) > 4
        x(i,2) = 4;
    end
    if rand > 0.5
        x(i,2) = round(x(i,2));
        x1 = floor(x(i,1));
```

```
        x2 = ceil(x(i,1));
        y1 = x(i,2);
        y2 = x(i,2);
    else
        x(i,1) = round(x(i,1));
        x1 = x(i,1);
        x2 = x(i,1);
        y1 = floor(x(i,2));
        y2 = ceil(x(i,2));
    end
    % 存在和重合的情况dot1dot2
    dot1 = find(dots(:,1)==x1 & dots(:,2)==y1);
    dot2 = find(dots(:,2)==x2 & dots(:,2)==y2);
    if isempty(dot1) && isempty(dot2)
        ra = rand;
        if ra <0.25
            x(i,1) = x(i,1)+1;
        elseif ra >=0.25 && ra < 0.5
            x(i,1) = x(i,1)-1;
        elseif ra >= 0.5 && ra < 0.75
            x(i,2) = x(i,2)+1;
        else
            x(i,2) = x(i,2)-1;
        end
    elseif isempty(dot1)
        x(i,1) = x2;
        x(i,2) = y2;
    elseif isempty(dot2)
        x(i,1) = x1;
        x(i,2) = y1;
    end
end
```