交巡警服务平台的设置与调度

摘　要

本文针对交巡警服务平台的设置与调度问题，建立了基于 Floyd 算法的管辖范围分配模型、具有多目标的 0-1 规划模型、多目标优化模型、基于遗传算法的平台重新分配模型以及基于资源预分配思想的快速围堵模型，分别解决了 A 区交巡警服务平台管辖范围的分配、A 区交通要道的快速封锁、A 区平台的新增、全区现有交巡警服务平台设置方案的评价及改进、对嫌犯的快速围堵问题。

针对问题一第一小问，建立了基于 Floyd 算法的管辖范围分配模型模型，解决了 A 区交巡警服务平台管辖范围的分配问题。首先我们将该市的交通网络抽象为无向赋权图，并为每条道路赋权；其次利用 Floyd 算法计算出任意路口间的最短路径；最后，在尽量满足三分钟内到达案发地的条件下，以工作量均衡为目标。提出一种管辖范围划分算法。最终得到了相应的分配方案，在此方案下，A 区的道路覆盖率为 73.23%。

针对问题一第二小问，建立了具有多目标的 0-1 规划模型，解决了 A 区交通要道的快速封锁问题。首先我们确定 13 个平台调度的时间最短为第一目标，调度总路径尽可能短为第二目标，各平台工作量差异尽可能小为第三目标；然后我们建立了 0-1 规划模型；最后对模型进行编程求解，得到调度方案，以及封锁 13 个交通要道的总时间为 82.59 分钟

针对问题一第三小问，建立了多目标优化模型，解决了 A 区平台的新增问题。首先我们选取路口覆盖率、案件覆盖率、平均服务面积 3 个指标；其次确定目标函数为 2 个指标的线性加权和；最后用枚举法确定新增平台的插入区域，并利用问题一第一小问的管辖范围划分算法得到使目标函数最小的节点位置。最终得到在考虑新增平台代价的条件下，插入 4 个平台最合适，位置分别为 69-74 路段、75-76 路段、85-86 路段、87-88 路段。

针对问题二第一小问，建立了基于遗传算法的平台重新分配模型，解决了全区现有交巡警服务平台设置方案的评价及改进问题。首先，我们以将问题一第三小问得到的 3 个指标线性加权得到方案合理性的综合评价值来评判全区现有交巡警服务平台设置方案的合理性；其次，利用改进的遗传算法对平台进行重新分配。最终得到现有交巡警服务平台设置方案的评价值为 0.61，说明平台设置不合理，解得的最优平台设置方案的评价值为 0.83，相比现有方案有较大提高。

针对问题五，建立了基于资源预分配思想的快速围堵模型，解决了对嫌犯的快速围堵问题。首先，我们确定最短时间将嫌疑犯完全围堵和最少资源浪费为目标；其次我们根据嫌犯的可能逃跑路径采用预分配的方式提出交巡警服务平台的调度算法；最终得到完成围堵的时间为 9.67 分钟。

关键词： Floyd 算法 0-1 规划 多目标优化 遗传算法 资源预分配

一、问题重述

“有困难找警察”，是家喻户晓的一句流行语。警察肩负着刑事执法、治安管理、交通管理、服务群众四大职能。为了更有效地贯彻实施这些职能，需要在市区的一些交通要道和重要部位设置交巡警服务平台。每个交巡警服务平台的职能和警力配备基本相同。由于警务资源是有限的，如何根据城市的实际情况与需求合理地设置交巡警服务平台、分配各平台的管辖范围、调度警务资源是警务部门面临的一个实际课题。

试就某市设置交巡警服务平台的相关情况，建立数学模型分析研究下面的问题：

（1）附件 1 中的附图 1 给出了该市中心城区 A 的交通网络和现有的 20 个交巡警服务平台的设置情况示意图，相关的数据信息见附件 2。请为各交巡警服务平台分配管辖范围，使其在所管辖的范围内出现突发事件时，尽量能在 3 分钟内有交巡警（警车的时速为 60km/h）到达事发地。

对于重大突发事件，需要调度全区 20 个交巡警服务平台的警力资源，对进出该区的 13 条交通要道实现快速全封锁。实际中一个平台的警力最多封锁一个路口，请给出该区交巡警服务平台警力合理的调度方案。

根据现有交巡警服务平台的工作量不均衡和有些地方出警时间过长的实际情况，拟在该区内再增加 2 至 5 个平台，请确定需要增加平台的具体个数和位置。

（2）针对全市（主城六区 A，B，C，D，E，F）的具体情况，按照设置交巡警服务平台的原则和任务，分析研究该市现有交巡警服务平台设置方案（参见附件）的合理性。如果有明显不合理，请给出解决方案。

如果该市地点 P（第 32 个节点）处发生了重大刑事案件，在案发 3 分钟后接到报警，犯罪嫌疑人已驾车逃跑。为了快速搜捕嫌疑犯，请给出调度全市交巡警服务平台警力资源的最佳围堵方案

二、问题分析

2.1 问题一的分析

本题要求我们给出各交巡警服务平台的分配管辖范围，针对此问题，我们首先需要知道各节点之间的距离，以及各平台到节点之间的距离，构造一个最短距离矩阵。这类问题我们通常采用运筹学中的 Floyd 算法，抽象城区交通图为无向图，然后保证各平台到节点之间的距离小于 3km，最后得出各交巡警服务平台分配管辖范围。

2.2 问题二的分析

制定合理的调度方案使得 13 条交通要道实现快速封锁，这是运筹学中典型的指派问题。这类问题我们通常需要建立多级目标，然后利用 0-1 规划求解。因此我们需要算出各平台到节点之间的最短距离，通过结合最短距离和对目标的约束条件进行求解。

2.3 问题三的分析

为了解决交巡警服务平台的工作量不均衡和有些地方出警时间过长的实际问题，我们需要新增平台个数，即插入 2-5 个平台来减轻某些平台工作量的压力。对于此类问题，我们首先需要建立合适的指标来判断各区域的平台分配如何合理分配；其次通过计算在分配最不合理的地区增设平台；最后得出平台的具体位置和个数。

2.4 问题四的分析

要想分析该市交巡警服务平台设置的合理性，首先需要建立一个综合评价模型。其次，针对其现有的交巡警服务平台方案进行评价并提出改进的方法。通常情况下采用遗传算法来改进平台的分配方案。最后，对所改进的方案作出合理性的解释说明，并与之前的方案对比。

2.5 问题五的分析

要在重大事件发生时调动全市的警力在最快的时间内抓捕逃犯，首先我们需要知道其所在位置并推测其下一时间可能出现的位置，通过计算判断警力是否能在犯罪嫌疑人到达该点之前实现围捕。接着建立基于资源预分配思想的快速围堵模型，力求在最短时间围捕犯罪嫌疑人，分配交巡警服务平台的调度。

三、 基本假设

1、对于 6 个区域来说，其人口是均匀分布的；

2、假设罪犯逃跑的速度和交巡警的速度一致，都为 60km/h;

3、交巡警在接到报警后立即出发驶向围堵路口节点，期间无延迟；警车从平台节点到围堵路口节点是按最短路径行驶的。

4.道路没有单行道而且都是通畅的。

四、符号说明

p	每条路的权值
s	服务平台工作量
D_{min}	最短路径矩阵
d_i^+	正偏差变量
d_i^-	负偏差变量
f_i	案发率
a	总方差
l_{ij}	节点 i 到出口 j 的距离
P	路口覆盖率

A	案件覆盖率
E	平均服务面积
C_i	第 i 时段的警力资源

五、模型的建立与求解

5.1 问题一第一问的模型建立与求解

为了合理分配交巡警服务平台的管辖范围，使得其在所管辖的范围内出现突发事件时能有交巡警在 3 分钟内（警车的时速为 60km/h）到达事发地，就需要我们计算出城区内各节点之间的最短路径，以保证距离各节点 3km 之内尽量有服务平台。因此我们首先将城区交通图抽象为无向图，其次通过其邻接矩阵得到各节点之间的最短路径，最后建立基于 Floyd 算法的管辖范围分配模型得出合理的分配方案。

5.1.1 模型的准备

为了便于分配 A 区平台的管辖范围，我们运用运筹学中的图论把 A 区的网络交通图抽象为无向图 G(V,E)。V 为各顶点集，代表城区内所有的节点；E 为边集，代表城区内两节点之间路段的长度。接着得到该无向图的邻接矩阵。邻接矩阵是表示顶点之间相邻关系的矩阵，对于无向图而言，其邻接矩阵是对称的，而且主对角线一定为零。

5.1.2 模型的建立

1、题目分析

根据题目中所给要求，为了尽量满足有交巡警在 3 分钟之内赶到事发地(速度为 60km/h)的条件，就需要平台与节点之间的距离尽可能小于 3km。并且，为了均匀分配各交巡警服务平台的管辖范围，我们决定选用“按路分配”的方法。而“按路分配”可以分为以下 3 种情况：

(1)A、B 节点处有一个节点设置了服务平台 1，则 A、B 所在路段为平台 1 的管辖范围；

(2)在节点 A、B 处都没有设置服务平台：

①节点 A、B 距离服务平台 1 都不到 3km，则 A、B 所在路段都属于平台 1 的管辖范围；

②A、B 中 A 到距 B 最近的平台 1 的距离超过 3km，而 B 到距 A 最近的平台 2 的距离不足 3km，则 A、B 所在路段属于平台 2 的管辖范围；

③A、B 中 A 到距 B 最近的与 B 到距 A 最近的的距离都大于 3km，则 A、B 节点所在路段属于平台 1 与平台 2 中工作量小的平台管理；

(3)A、B 节点处分别设置了服务平台 1、2：

①A、B 间路段长度小于 3km，则该路段属于平台 1 与平台 2 中工作量小的平台管理；

②A、B 间路段长度大于 3km，则我们将该路段分为两段，小于 3km 的路段属

于平台 1 与平台 2 中工作量小的平台管理，其余路段属于工作量大的管理。

针对以上的情况，我们发现分配方案过程中可能会出现平台工作量不均匀的问题，为了更适合的制定分配方案，我们引入以下两个定义：

(1)每条路段权值的定义：

$$\mathrm{p}=\frac{a+b}{2l}$$

其中，a、b 为该路段两节点的案发率，l 为该路段的长度。

(2)服务平台工作量的定义：

$$\mathrm{s}=\sum_{i=1}^{n} l_i{}'\frac{a+b}{2l_i}$$

其中，l_i是第 i 条路段的长度，$l_i{}'$是第 i 条路段中按比例分配给平台管辖的路段长度。

由此，我们建立基于 Floyd 算法[1]的管辖范围分配模型。

2、基于 Floyd 算法的管辖范围分配模型

(1)为了方便计算管辖的范围，需要计算出各节点之间的最短路径，我们选用运筹学中的 Floyd 算法，即通过一个图的邻接矩阵求出它的每两点间的最短路径矩阵D_{min}。Floyd 算法计算最短路径矩阵的步骤如下：

步骤一：从任意一条单边路径开始。所有两点之间的距离是边的权，如果两点之间没有边相连，则权为无穷大；

步骤二：对于每一对顶点 u 和 v，看看是否存在一个顶点 w 使得从 u 到 w 再到 v 比已知的路径更短，如果是更新它。

于是，我们得到了城区 A 的最短路径矩阵D_{min}：

$$D_{min}=\begin{bmatrix} t_{11} & t_{12} & \cdots & t_{1,\,582} \\ t_{21} & t_{22} & \cdots & t_{2,\,582} \\ \cdots & \cdots & \cdots & \cdots \\ t_{20,1} & t_{20,2} & \cdots & t_{20,582} \end{bmatrix}$$

(2)我们根据以上引入的定义算出各路段的权值和工作量。

(3)首先，我们考虑 A、B 节点处有一个平台的情况。其次，考虑在节点 A、B 处都没有设置服务平台的情况。最后考虑 A、B 节点处均设置服务平台的情况。

5.1.3 模型的求解

根据以上数据，通过 matlab 求解得到 A 区各平台的管辖区范围和工作量如表 1 表 2 所示：

表 1 A 区各平台管辖范围

平台	分配路段个数	具体管辖的路段
A1	16	1-69,1-74,1-75,1-78,67-68,68-69,68-75,69-70,69-71,71-72,71-74,73-74,74-80,75-76,76-77,78-79
A2	9	2-40,2-43,2-44, 2-70,39-40, 43-72,43-70,44-67,72-73

A3	9	3-45, 3-65, 3-44, 3-55, 54-55, 54-63, 65-66, 66-67, 66-76
A4	12	4-39, 4-57, 4-62, 4-63, 57-58, 57-60, 60-62, 60-61, 62-85, 63-64, 64-65, 64-76
A5	13	5-49, 5-50, 5-47, 49-53, 49-50, 50-51, 51-52, 51-59, 52-56, 52-53, 53-54, 56-57, 58-59
A6	2	6-47, 6-59
A7	9	7-15, 3-30, 7-32, 7-37, 7-47, 30-48, 32-33, 47-48, 48-61
A8	6	8-9, 8-47, 8-33, 8-46, 33-34, 46-55
A9	7	9-34, 9-35, 31-32, 31-34, 35-45, 35-36, 45-46
A10	2	10-26, 10-34,
A11	4	11-22, 11-25, 11-26, 26-27
A12	2	12-25, 12-27
A13	5	13-22, 13-23, 13-24, 21-22, 24-25
A14	2	14-16, 14-21,
A15	4	15-28, 15-31, 28-29, 29-30
A16	6	16-36, 16-38, 36-37, 36-39, 38-39, 38-41
A17	6	17-40, 17-41, 17-42, 17-81, 41-92, 42-43
A18	8	18-73, 18-80, 18-81, 18-83, 81-82, 82-83, 82-90, 83-84
A19	4	19-77, 19-79, 77-78, 79-80
A20	14	20-85, 20-86, 20-89, 84-85, 84-89, 86-87, 86-88, 87-88, 87-92, 88-89, 88-91, 89-90, 90-91, 91-92

表 2 各平台工作量

平台	工作量
A1	16.8000
A2	12.8000
A3	11.1000
A4	13.1000
A5	14.4000
A6	3.7500
A7	13.6322
A8	8.3500
A9	10.8000
A10	3.0500
A11	7.0000
A12	3.6000
A13	8.5000
A14	3.0847
A15	8.4178
A16	9.6153
A17	10.6000
A18	10.0000

A19	4.2000
A20	16.4500

结合表 1 表 2 中的数据我们可以看出，各平台管辖的路段数量并不均衡，其中 A1、A4、A5、A20 管辖的路段数量较多，分别为 16、12、13、14 个；而平台 A6、A10、A12、A14 管辖的路段数量最少，都为 2 个。并且各平台的工作量差别也很大，平台 A1 的工作量最大，为 16.800；而 A14 的工作量最小，为 3.0847。由此可以看出，我们分配的方案并不能很均匀的把所有路段都分配给平台，即此分配方案存在一定的不合理性。

5.2 问题一第二问的模型建立与求解

在一个平台只能封锁一个出口的条件下，为了实现快速封锁，且我们要求交巡警到达最远平台的时间最短。由于本题还需要满足交巡警资源的合理分配条件，即这是运筹学中典型的指派问题，因此我们选择建立多目标线性规划模型。此外，我们建立了三级目标，通过优先级的大小确定目标的重要程度，然后列出约束条件进行求解。

5.2.1 模型的建立

想要快速实现 13 条交通要道的全封锁，即要求 20 个交巡警服务平台的警察能够在最短时间内到达管辖范围内距各平台最远的节点。这就需要从以下几个方面考虑，即需要满足的目标：

(1) 最长路径最短，即使平台到达各自出口路径中的最长路径尽可能短；

(2) 总路径最短即各平台到达各自封锁出口的路径和最小；

(3) 各平台到达出口的路径差异最小，即总方差最小。

由于本题解决的是一个平台封锁一个路口的合理方案问题，即运筹学中典型的指派问题，因此我们选择建立多目标规划模型。

在多目标规划中，为了更好的解决问题，我们引入以下 4 个定义：

(1)正、负偏差变量：

设$f_i(i=1,2,\dots l)$为第 i 个目标函数，它的正偏差变量$d_i^{+}=max\{f_i-{d_i}^0,0\}$，表示决策值超过目标值的部分，负偏差变量$d_i^{-}=-min\{f_i-{d_i}^0,0\}$表示决策值未达到目标值的部分，这里${d_i}^0$表示$f_i$的目标值。决策值不可能既超过目标值同时又未达到目标值，即恒有$d_i^{+}\times d_i^{-}=0$。

(2)绝对约束和目标约束：

绝对约束是指必须严格满足的等式约束和不等式约束，如线性规划问题的所有约束条件，不能满足这些约束条件的解称为非可行解，所以他们是硬约束。在达到此目标值时允许发生正或负偏差，因此在这些约束中加入正、负偏差变量，它们是软约束。线性规划问题的目标函数，在给定目标值和加入正、负偏差后可变换为目标约束。也可根据问题的需要将绝对的约束变为目标约束。

(3)优先因子

一个规划问题常常有若干个目标。但决策者在要求达到这些目标时，是有主

次或轻重缓急之分的。凡是要求第一位达到的目标赋予优先因子 P1，，次位的目标赋予优先因子 P2，…，Pn，并规定$p_k \gg p_{k+1}$，表示p_k比p_{k+1}有更大的优先权。

(4)目标规划的目标函数

目标规划的目标函数是按各目标约束的正、负偏差变量和赋予相应的优先因子构造的。当每一目标值确定后，决策者的要求是尽可能的缩小偏离目标值。其基本形式有三种：

第 i 个目标要求恰好达到目标值，即正、负偏差变量都要尽可能的小，这时：

$$\min \ \ {w_i}^- {d_i}^- + {w_i}^+ {d_i}^+$$

第 i 个目标要求不超过目标值，即允许达不到目标值，即正偏差变量都要尽可能的小，这时：

$$\min \ \ {w_i}^+ {d_i}^+$$

第 i 个目标要求超过目标值，即超过量不限，但偏差变量都要尽可能的小，这时：

$$\min \ \ {w_i}^- {d_i}^-$$

由此，我们建立多目标线性规划模型。

设变量x_{ij}为 0-1 规划，即：

$$x_{ij} = \begin{cases} 1 & 第i个平台封锁第j个出口 \\ 0 & 第i个平台不封锁第j个出口 \end{cases}$$

从以上三个目标进行分析，我们发现在分配合理方案时应先满足第一个目标，其次尽可能满足第二个目标，最后考虑第三个目标。因此我们采用求解目标规划算法中的序贯算法。

序贯算法是求解目标规划的一种早期算法，其核心是根据优先级的先后次序，将目标规划问题分解成一系列的单目标规划问题，再依次进行求解。

根据以上 3 个目标，我们要尽可能的满足第一个目标；对于第 2 个目标来说，我们允许实际总路径的长度略长于最小总路径的长度；也允许各路径的实际偏差较最小总方差偏大。根据正负偏差变量的定义，我们建立目标函数如下：

$$\min z = \mathrm{P1}\left({d_1}^- + {d_1}^+\right) + \mathrm{P2}{d_2}^- + P3{d_3}^-$$

其中，其优先级分别为 P1、P2、P3。

记节点 i 的案发率为f_i,则该地区管辖范围内总案发率的平均值为：

$$\bar{f} = \frac{\sum_{i=1}^{20} f_i}{20}$$

考虑到第三个目标中各平台到达出口的路径差异最小，即总方差最小,我们需要求解：

$$\min a = \sum_{i=1}^{20} \sum_{j=1}^{13} (x_{ij} a_j - \bar{a})^2$$

约束条件如下：

$$\begin{cases} l_{ij} + {d_1}^+ - {d_1}^- = 27.686 \\ \sum_{i=1}^{NUM1}\sum_{j=1}^{NUM2} l_{ij} + {d_2}^+ - {d_2}^- = 403.293 \\ \sum_{i=1}^{NUM1}\sum_{j=1}^{NUM2} (l_{ij} - \overline{l_{ij}})^2 + {d_3}^+ - {d_3}^- = 365.2511 \\ i = 1,2,\dots 20; j = 1,2,\dots 13 \\ \sum_{i=1}^{NUM1} x_{ij} = 1 \\ \sum_{j=1}^{NUM2} x_{ij} \le 1 \\ x_{ij} = 0 \text{ 或 } 1 \end{cases}$$

其中，l_{ij}为节点 i 到出口 j 的距离。

通过以上条件求解合理的调度方案。

5.2.2 模型的求解

根据上述约束条件，解得调度方案如表 3 所示：

表 3 调度方案

交通要道	交巡警服务平台警力	距离	所用时间
12	A12	0	0
14	A9	82.74	1.379
16	A16	0	0
21	A14	32.65	0.544
22	A10	77.08	1.285
23	A13	5	0.083
24	A11	38.05	0.634
28	A15	47.52	0.792
29	A8	104.93	2.348
30	A7	5.83	0.097
38	A2	39.82	0.664
48	A5	24.76	0.411
62	A4	3.5	0.058

由表 3 我们得知，除了上表分配的 13 个平台，还有 7 个平台没有被分配，而且各平台到达案发地的时间不同，最短的达到时间为 0min，最长的到达时间为 2.348min。因此这种分配方案虽然能够满足在 3min 之内到达事发地，但并不能很好的解决合理警力资源的分配问题，即存在一定的缺陷。

5.3 问题一第三问的模型建立与求解

由第二问得到的分配结果，我们发现只考虑到在最短时间内到达事发地的条件并不全面，很可能会出现工作量不均衡和某些地方出警时间过长的情况。因此

我们首先新增 2-5 个平台来弥补这两个问题，即插入 2-5 个平台来减轻某些平台工作量的压力。其次，我们建立增设平台的最优化模型来评价此平台的分配方案是否合理。最后，得出新增平台的具体位置和个数，制定合理的分配方案。

5.3.1 模型的建立

从第一问我们可以看出，现有的交巡警分配方案并不能使得所有交巡警资源得到最优的分配。因此，我们考虑在现有分配方案的基础上增设 2-5 个服务平台，解决交巡警服务平台工作量不均衡和有些地方出警时间过长的问题。在不扩大平台出警时间的前提下使各平台的工作量均衡，就要通过在工作量较大的平台处插入新的平台来减小其出警时间，达到工作量均衡的目的。为了得到合理的分配方案，我们建立评价指标体系进行更深入的分析：

从同一区域内考虑

(1) 路口覆盖率 P:

即各平台管辖范围内 3min 内可到达的路口占总路口数的百分比。之前我们将该地区的交通网络图抽象为无向图，并得到了任意两路口见的距离 d。基于此无向赋权图，我们计算其路口覆盖率的步骤如下：

步骤一：该地区交通路口数共有 N=582 个，x_j为第 j 个交通路口，初始时，j=0；

步骤二：遍历第 j 个交通路口所属区域中的所有平台，若$d \le 3km$，则将x_j加入集合 U，U 表示平台 3min 内可以到达的路口的集合；

步骤三：若 j<N，则 j=j+1，转步骤二；若 j=N,转步骤四；

步骤四：统计集合 U 中的元素个数 k，计算出覆盖率函数 P。

(2)案件覆盖率 A：

即统计各平台管辖范围内的能在 3min 内到达的路口的案发次数之和，计算出案发次数之和占案发总次数的百分比，即：

$$A = \frac{\sum_{v_i \in N_{ac}} a(v_i)}{\sum_{v_i \in N} a(v_i)}$$

其中，$N_{ac} = \{v_i | d(v_i, v_j) \le 3, v_i \in N\}$，即 3min 内能到达的节点构成的集合，$a(v_i)$表示节点$v_i$的案发率，N 表示所有节点的集合。

因为要同时满足以上两个条件，所以我们采用线性加权和法将以上两个目标转化为单目标，建立如下组合目标函数：

$$f = min[\alpha \bullet \gamma \bullet \frac{A' - A}{A} - (1 - \alpha) \bullet \frac{P' - P}{P}]$$

其中，α用以确定这两个因素各自的权重，满足$0 < \alpha < 1$，γ用以调整使这两个因素具有可比性。在本题中，我们可以近似的认为路口覆盖率 P 和案件覆盖率 A 同等重要，因此在这里我们取$\alpha = 0.5$。

从不同区域内考虑

在考虑区域之间分配方案合理性时，我们还需要计算各区域各平台的平均服务面积 E，即

$$E = \frac{X}{S}$$

其中，X 代表该区域内的平台个数，S 代表该区域的面积。

通过满足以上 3 个方面的条件来增设平台，建立增设平台的最优化模型。

5.3.2 模型的求解

根据以上条件并结合基于 Floyd 算法的管辖范围分配模型，解得平台的插入方案如表 4 所示：

表 4 插入路段

平台	添加位置
A21	43
A22	51
A23	64
A24	89

5.4 问题二第一问的模型建立与求解

5.4.1 现有平台设置方案评价

在问题一第三小问中，我们已经得到了衡量交巡警平台设置方案的三个指标，并进行线性加权得到方案合理性的综合评价值，从而评判全区现有交巡警服务平台设置方案的合理性。

通过计算，各区的各项指标如表 5 所示。

表 5 现有平台设置方案六个城区指标对比

区域	A 区	B 区	C 区	D 区	E 区	F 区
路口覆盖率 P	0.73	0.51	0.52	0.54	0.41	0.46
平均服务面积 E	0.91	0.078	0.077	0.023	0.035	0.040
案件覆盖率 A	0.81	0.76	0.70	0.72	0.68	0.64

由此，我们算出现有平台设置方案的评价值为 0.61，该市现有平台设置不合理。

5.4.2 问题四的模型建立与求解

由于该市现有平台设置不合理，所以我们要在全市范围内重新分配警务资源，也就是重新为每个区分配交巡警服务平台的数目。遗传算法[2]-[5]借助生物进化理论，体现了优胜劣汰思想，通过交叉及变异操作保证了种群的多样性，具有较强的全局搜索能力和并行处理能力。因此，我们建立了基于遗传算法的平台重新分配模型。

5.4.2.1 模型的建立

（1）遗传算法概述

遗传算法是一种基于自然选择原理和自然遗传机制的搜索和优化方法，它是模拟自然界中的生命进化机制，在人工系统中实现特定目标的优化。

①染色体：在使用遗传算法时，需要把问题的解编成一个适合的码子。这种具有固定结构的符号串即是染色 体。符号串的每一位代表一个基 因。符号串的总位数称为染色体的长度。一个染色体就代表问题的一个解，每个染色体也被 称为一个个体。

②群体：每代所产生的染色体总数称为群体，一个群体包含了该问题在这一代的一些解的集合。

③适应度：对群体中每个染色体进行编码后，每个个体对应一个具体问题的解，而每个解对应于一个函数值。该函数值 即适应函数，就是衡量染色体对环境适应度的指标，也是反映实际问题的目标函数 。

（2）模型建立

我们以第三问建立评价指标体系为目标函数：

$$f = min[\alpha \frac{A' - A}{A} - \beta \frac{P' - P}{P} - \gamma \frac{E' - E}{E}]$$

约束条件为

$$\sum_{1}^{582} w_i = 80$$

5.4.2.2 模型的求解

遗传算法的参数设定如下：

染色体长度：582

种群大小： M=100

最大代数：G=1000

1.编码策略

采用十进制编码，用随机数列$w_1 w_2 \cdots w_{582}$作为染色体，染色体基因数为582，染色体的第j个基因用w_i来表示。用w_i=1 或 0 分别表示路口j是否安置平台，由于该城市的平台数量不变，所以$\sum_1^{582} w_i = 80$。

2.始种群

利用改良圈算法求得一个较好的初始种群。对于随机产生的初始圈

$$C = \pi_1 \cdots \pi_{u-1} \pi_u \pi_{u+1} \cdots \pi_{v-1} \pi_v \pi_{v+1} \cdots \pi_{582},$$

$$(2 \le u < v \le 581, 2 \le \pi_u < \pi_v \le 581),$$

交换u，v之间的顺序，此时的新路径为

$$\pi_1 \cdots \pi_{u-1}\pi_v\pi_{v-1} \cdots \pi_{u+1}\pi_v\pi_{v+1} \cdots \pi_{582}$$

记$\Delta f = \left(d_{\pi_{u-1}\pi_v} + d_{\pi_u\pi_{v+1}}\right) - (d_{\pi_{u-1}\pi_u} + d_{\pi_v\pi_{v+1}})$，若$\Delta f < 0$，则以新路径修改旧路径，直到不能修改为止，就得到一个比较好的可行解。

直到产生M个可行解，并把这M个可行解转换成染色体编码。

3.目标函数

$$f = min[\alpha\frac{A' - A}{A} - \beta\frac{P' - P}{P} - \gamma\frac{E' - E}{E}]$$

4.交叉操作

首先，对父代个体进行配对，即对父代以适应度函数（目标函数）值进行排序，目标函数值小的与小的配对，目标函数值大的与大的配对。然后利用混沌序列确定交叉点的位置，最后对确定的交叉项进行交叉。

5.变异操作

变异是实现种群多样性的一种手段，是寻求全局最优的重要保证。首先根据给定的变异率，随机的选取 2 个整数，对这两个数对应位置的基因进行变异，变异时利用混沌序列把这两个位置的基因换成新的基因值，从而得到新的染色体。

6.选择采用确定性的选择策略，也就是在父代种群和子代种群中选择目标函数值最小的个个体，以保证父代的优良特性被保存下来。

通过编程求解，得到最优平台设置方案如表所示 6。

表 6　6 个城区优化后各平台位置

A	1、2、3、4、5、6、7、8、9、11、13、14、15、17、18、20、26、52、66、73、90
B	93、94、95、96、97、121、145、63
C	166、167、171、172、173、180、181、182、191、201、231、245、254、256、261、268、272
D	320、321、323、324、326、327、328、331、368
E	372、373、375、376、378、389、380、381、383、384、386、401、421、423、443
F	475、476、478、480、482、502、511、531、542、567、570

计算求得，最优平台设置方案的评价值为 0.83，相比现有方案有较大提高，因此我们得到的最优平台设置方案是比较合理的。

5.5 问题二第二问的模型建立与求解

为了在发生重大事件时调动全市警力资源实现最佳围堵，首先要推测其可能的逃跑路线；其次，建立基于资源预分配思想[6]-[7]的快速围堵模型，以在最短时间围捕犯罪嫌疑人为目标，提出交巡警服务平台的调度算法，最后，得到交巡警围捕罪犯的最短时间。

5.5.1 模型的建立

根据题意，当重大事件发生之后，交巡警需要尽快去围捕正驾车逃跑的犯罪

嫌疑人。为了更好的实现围捕，本题中我们采用“两头堵”的围捕方法，即分配警力围捕犯罪嫌疑人所在路段的两节点。犯罪嫌疑人在逃跑的过程中，会想要尽快逃离其所在城区，即找到距离其最近的出城路口且该路口没有设立交巡警服务平台。在罪犯逃跑的过程中，交巡警想要在最快时间内抓住罪犯并使其不出城，就需要赶在罪犯之前到达其接下来可能到达的点，实现围捕。但在追捕的过程中，有可能存在某些点罪犯一定先于警察到达，对于这种情况，就需要继续预测罪犯下一时间可能到达的地方，再分配警力实现围捕。

但是，传统的调度方案，即根据逃犯的逃跑路线立即分配警力实施追捕，如果在预期的节点不能先于逃犯到达，则继续沿路追捕的方案，不仅浪费了警力资源，也使得围捕的时间变得更长。因此，我们选用资源预分配的思想进行调度。

预分配即是在有序分配的过程中，合理的预防死锁的产生，实际上是一种资源进行的静态管理调度，需要从目标预测轨迹全程角度考虑管理方法，优于弥补和再分配。其目的是为各量化时段选择最优处理的资源组合。考虑到每一时段对目标可实现有效覆盖的资源组合是不断变化的，因此预分配就是从某时段可覆盖的资源集合中选择若干个资源对目标进行处理，实际上是一个约束最优化问题。

假定矢量集$\mathrm{A}=\{A_1，A_2，\ldots,A_n\}$表示每一量化时段所选择的资源组合，则资源与分配的优化模型可表示为：

$$\begin{cases} Min\{f(A)\},\mathrm{A}=\{A_1，A_2，\ \ldots,A_N\} \\ \quad s.t.\quad A_i \subseteq C_i, i=1,2,\ldots,N \end{cases}$$

其中，$f(A)$为优化目标函数；C_i为第 i 时段对目标可实现有效覆盖的资源集合，即本题中的警力资源。

5.5.2 模型的求解

根据预分配的方法，我们得出警力资源的最佳围堵方案如下表所示：

表 7　最优方案的围堵路段

围堵路段	围堵距离
8-33	5.1
9-34	5.02

即最优围堵路段分别为 8-33、9-34；围堵距离分别为 5.1km、5.02km，最优围堵时间为 9.67 分钟。

六、模型的推广与评价

6.1 模型的优点

（1）问题一第一小问中的基于 Floyd 算法的管辖范围分配模型较好的解决了交巡警平台的管辖范围，且考虑到了案件发生在道路上而不只是路口的情况。

（2）多目标规划将多个目标分级处理，考虑的比较全面，更符合实际情况。

（3）遗传算法具有较强的全局搜索能力和并行处理能力，求解迅速。

（4）本模型的设置与调度问题，当事故发生时，交巡警可以第一时间到达事故发生地点，有效的改善了交巡警在执行任务中的效率。除此之外，本模型还解决了遇到突发事件时的最佳围堵方案、调度方案，以及对警力的合理分配。

6.2 模型的缺点

（1）问题一第一小问中的基于 Floyd 算法的管辖范围分配模型虽然可以在本题应用，但并不适用于数据较多的情况，其算法复杂度较高。

（2）问题三在增加新平台时，只是在工作量较大的平台周围选取新平台，可能不是全局最优解。

6.3 模型的改进

问题的求解将模型理想化，如现实中不能时刻都保证道路的畅通性。也不可能转弯时不用时间，即不能保证出警的时间总是维持在 3 分钟之内。为了更具有普遍性，则应考虑道路的畅通性以及其他因素对出警所用时间的影响，本模型忽略了实际生活中存在的不定因素。

6.4 模型的推广

结合基于遗传算法的平台重新分配模型的特点，该模型也可运用到其他最优选址问题中去，比如消防站、医院、超市等一些重要公共设施的选址问题、重大生产事故应急援救、公共交通的最优路径问题等。

参考文献

[1]左秀峰,沈万杰，基于 Floyd 算法的多重最短路问题的改进算法[J]，计算机科学,2017,(05):232-234+267.

[2]Pingxian Wu,Yanzhi Jiang,Li Zhu,Xuewei Li,Guoqing Tang,Optimizing the gain of social genetic effect under the control of inbreeding using genetic algorithm[J], Livestock Science,2016.

[3]崔珊珊，遗传算法的一些改进及其应用[D]，中国科学技术大学，2010.

[4]汪松泉，遗传算法在组合优化中的应用研究[D]，安徽大学,2010.

[5]胡飞虎, 马贝龙, 杨丽等,基于改进遗传算法的应急物资配送车辆调度优化问题研究[J]，计算机应用研究, 2014, 31(10):2928-2932.

[6]王博,刘海军,安玮,周一宇,基于粒子群优化的传感器预分配方法[J],信号处理,2010,(04):486-491.

[7]栾德杰,倪宏,潘梁,吴丽彬,孙鹏,一种基于启发式算法的数据块扩展预分配策略[J],微计算机信息,2010,(35):70-71+101.

附录

附录一：最短距离计算函数

```
function [d,path]=floyd(a)
[n,m]=size(a);
d=a;
path=zeros(n,n);

for k=1:n
    for i=1:n
        for j=1:n
            if d(i,j)>d(i,k)+d(k,j)
                d(i,j)=d(i,k)+d(k,j);
                path(i,j)=k;
            end
        end
    end
end
```


附录二：管辖范围划分算法

```
clc,clear
% 全市交通网络中路口节点节点坐标
A=load('F:\数学建模\暑假培训\11B\数据\全市交通路口节点.txt');
% 全市交通路线 4:权值
B=load('F:\数学建模\暑假培训\11B\数据\全市交通路线.txt');
C=load('F:\数学建模\暑假培训\11B\数据\服务台位置编号.txt');
n=size(B,1);
D=ones(size(A,1));%距离矩阵
D(:)=inf;
for i=1:size(A,1)
    D(i,i)=0;
end
for i=1:n
    a=B(i,1);   % 起点标号
    b=B(i,2);% 终点标号
    d=sqrt((A(a,2)-A(b,2))^2+(A(a,3)-A(b,3))^2);
    %fprintf(shuchu,'%d    %d    %6.2f\r\n',a,b,d);
    D(a,b)=d;
    D(b,a)=d;
end
%求最小路
[dmin,path]=floyd(D);
%每条路赋值
```

```
for i=1:size(B,1)
    a1=B(i,1);
    b1=B(i,2);
    B(i,4)=(A(a1,5)+A(b1,5))/(2*D(a1,b1));
end
b2=1;
S=zeros(size(dmin));
for a2=1:582
    for b2=1:582
        if path(a2,b2)==0&&a2~=b2
            S(a2,b2)=(A(a2,5)+A(b2,5))/(2*D(a2,b2));
        end
    end
end
%两节点之间路赋值
for a2=1:582
    for b2=1:582
        B2=b2;
        while path(a2,b2)~=0
            h=path(a2,b2);
            S(a2,B2)=S(a2,B2)+S(b2,h);
            b2=h;
        end
    end
end
%%
%划分 A 区交巡警服务平台
E=load('F:\数学建模\暑假培训\11B\数据\A 区道路.txt');
t=140;%要改成 A 区道路数
fp1=zeros(t,6);%1 入口  2 出口 3 路编号，4 是否分配，5 分配到哪个平台 6 路长
fp1(:,1)=E(:,1);
fp1(:,2)=E(:,2);
fp1(:,3)=E(:,3);
fp1(:,6)=inf;
gzl1=zeros(20,2);%1 平台编号，2 此平台总工作量
gzl1(:,1)=C(1:20,1);
%首先分配能直接到达平台的路
for i=1:t
    m=E(i,1);n=E(i,2);
    if m<=20&&n<93&&n>20   %m 为平台
            fp1(i,4)=1;
            fp1(i,5)=m;
            fp1(i,6)=D(m,n);
            gzl1(m,2)=gzl1(m,2)+S(m,n)*fp1(i,6);
```

```
    end
end
%再分配
for i=1:t
    j1=0;
    j2=0;
    if fp1(i,4)==0&&fp1(i,1)>20
        m=E(i,1);n=E(i,2);
        dm1=inf;dm2=inf;
        for j=1:20 %找出距离 m 最近的平台 j1
            if dm1>dmin(j,m)
                dm1=dmin(j,m);
                j1=j;
            end
        end
        for j=1:20 %找出距离 n 最近的平台 j2
            if dm2>dmin(j,m)
                dm2=dmin(j,m);
                j2=j;
            end
        end
        if (dmin(j2,m)<30&&dmin(j1,n)<30)||(dmin(j2,m)>30&&dmin(j1,n)>30)
            fp1(i,4)=1;
            if gzl1(j1,2)<=gzl1(j2,2)
                fp1(i,5)=j1;
                fp1(i,6)=dmin(j1,n);
                gzl1(j1,2)=gzl1(j1,2)+S(m,n)*D(m,n);
            end
            if gzl1(j1,2)>gzl1(j2,2)
                fp1(i,5)=j2;
                 fp1(i,6)=dmin(j2,m);
                gzl1(j2,2)=gzl1(j2,2)+S(m,n)*D(m,n);
            end
        end
        if dmin(j2,m)>30&&dmin(j1,n)<30
            fp1(i,4)=1;
            fp1(i,5)=j1;
            fp1(i,6)=dmin(j1,n);
            gzl1(j1,2)=gzl1(j1,2)+S(m,n)*D(m,n);
        end
        if dmin(j2,m)<30&&dmin(j1,n)>30
            fp1(i,4)=1;
            fp1(i,5)=j2;
            fp1(i,6)=dmin(j2,m);
```

```
                gzl1(j2,2)=gzl1(j2,2)+S(m,n)*D(m,n);
        end
    end



for i=1:t
    if fp1(i,4)==0&&fp1(i,2)<=20
        m=E(i,1);n=E(i,2);
        fp1(i,4)=1;
        if D(m,n)<30
            if gzl1(m,2)<gzl1(n,2)
                fp1(i,5)=m;
            else
                fp1(i,5)=n;
            end
            fp1(i,6)=D(m,n);
        end
        if D(m,n)>30
            fp1(i,5)=n*10+m;
            if gzl1(m,2)<gzl1(n,2)
                gzl1(m,2)=gzl1(m,2)+S(m,n)*30;
                gzl1(n,2)=gzl1(n,2)+S(m,n)*(D(m,n)-30);
            else
                gzl1(m,2)=gzl1(m,2)+S(m,n)*(D(m,n)-30);
                gzl1(n,2)=gzl1(n,2)+S(m,n)*30;
            end
        end
    end
End
```

附录三：0-1 规划模型代码

```
model:
sets:
pingtai/1..13/:b;
chukou/1..20/:d;
link(pingtai,chukou):a,x;
S_Con_Num/1..3/:dplus,dminus;
endsets
data:
a=
222.36204.64183.52219.97176.28176.59149.15140.93130.1175.87 37.91 0 59.77 119.5 170.3 145.43 218.92 242.47 225.47 269.46
160.28 141.3 127.67 150.09 129.7 130 109.01 94.34 82.74 127.76 83.37 119.5 59.73 0 132.98 67.42 149.03 185.14169.61
212.13 92.87 73.88 60.26 82.67 62.28 62.59 41.6 26.92 15.33 69.57 113.95 145.43 127.15 67.42 65.56 081.62 117.73 102.2
144.71 192.93 173.95 160.32 182.73 162.35 162.65141.66126.99115.3995.11 50.72 86.85 27.08 32.65 165.63 100.07181.68
```

```
217.79202.26244.78 210.96 191.97178.35200.76 177.5 177.8 150.36 142.14 131.32 77.08 32.7 68.83 9.06 50.68 171.51
118.09199.71235.82220.29262.81 225.02 206.03192.41214.82191.55 191.86 164.42156.19145.3891.13 46.75 64.77 5 64.73
185.56 132.1 213.77249.88234.35276.86 228.93 211.21190.09226.54182.85183.16 155.72 147.5 136.68 82.44 38.05 35.92
23.85 83.59 176.87 151 225.49 249.04 232.04276.03 190.01 172.29151.17162.27113.07113.3785.7 102.28 97.76 141.95
186.33217.81228.08180.5 47.52 113.08 186.57 210.12 193.12230.11 195.16 177.44156.32155.35106.15106.46
80.15 104.93 107.24151.44195.82227.3 237.57 189.1757.01 121.75 195.24 215.27 198.26223.19 120.83 103.1182
81.03 31.83 32.14 5.83 30.61 34.92 79.11 123.5 154.98 165.25114.8444.01 47.43 120.92 140.94 123.94 148.87 58.81 39.82 60.94 48.61
94.21 94.52 73.53 58.85 47.26 101.5 145.88 177.36161.21101.4897.5 34.06 47.56 83.67 76.39 110.66 118.5 103.1 81.98 73.96
24.76 25.06 12.9 30.99 41.99 86.19 130.57 162.05172.32121.9151.09 54.5 127.99 136.99119.99141.8
48.85 60.35 43.93 3.5 52.55 53.37 79.92 86.77 93.37 147.61 191.99 223.47 213.32153.59118.1 86.17 78.21
67.34 50.34 64.49;
enddata
min=dplus(1)
@for(S_Con_Num(k):@sum(link(i,j):a*x)-dminus(k)=0);
@for(pingtai(i):@sum(chukou(j):x(i,j))=1;);
@for(chukou(j):@sum(pingtai(i):x(i,j))<=1;);
End
```

附录四：问题一第三小问代码

```
function [fp1,gzl1]=q3(C,biaohao,pts)
% 全市交通网络中路口节点节点坐标
A=load('F:\数学建模\暑假培训\11B\数据\全市交通路口节点.txt');
B=load('F:\数学建模\暑假培训\11B\数据\全市交通路线.txt');
n=size(B,1);
D=ones(size(A,1));%距离矩阵
D(:)=inf;

for i=1:size(A,1)
    D(i,i)=0;
end
for i=1:n
    a=B(i,1);  % 起点标号
    b=B(i,2);% 终点标号
    d=sqrt((A(a,2)-A(b,2))^2+(A(a,3)-A(b,3))^2);
    %fprintf(shuchu,'%d    %d    %6.2f\r\n',a,b,d);
    D(a,b)=d;
    D(b,a)=d;
end
%求最小路
[dmin,path]=floyd(D);
%每条路赋值
for i=1:size(B,1)
    a1=B(i,1);
    b1=B(i,2);
```

```
        B(i,4)=(A(a1,5)+A(b1,5))/(2*D(a1,b1));
    end
    b2=1;
    S=zeros(size(dmin));
    for a2=1:582
        for b2=1:582
            if path(a2,b2)==0&&a2~=b2
                S(a2,b2)=(A(a2,5)+A(b2,5))/(2*D(a2,b2));
            end
        end
    end
    %两节点之间路赋值
    for a2=1:582
        for b2=1:582
            B2=b2;
            while path(a2,b2)~=0
                h=path(a2,b2);
                S(a2,B2)=S(a2,B2)+S(b2,h);
                b2=h;
            end
        end
    end
    %%
    %划分 A 区交巡警服务平台

    E=load('F:\数学建模\暑假培训\11B\数据\A 区道路.txt');
    t=140;%A 区道路数
    fp1=zeros(t,6);%1 入口  2 出口  3 路编号，4 是否分配，5 分配到哪个平台 6 路长
    fp1(:,1)=E(:,1);
    fp1(:,2)=E(:,2);
    fp1(:,3)=E(:,3);
    fp1(:,6)=inf;
    gzl1=zeros(pts,2);%1 平台编号，2 此平台总工作量
    gzl1(:,1)=biaohao(:,1);
    %首先分配能直接到达平台的路

    for i=1:t
        m=E(i,1);n=E(i,2);
        if ismember(m,biaohao)&&n<93&&(ismember(n,biaohao)==0)   %m 为平台标号,判断此道路属于哪一类
                fp1(i,4)=1;
                fp1(i,5)=m;
                fp1(i,6)=D(m,n);
                gzl1(find(biaohao==m),2)=gzl1(find(biaohao==m),2)+S(m,n)*fp1(i,6);
        end
```

```
end
%再分配
for i=1:t
    j1=0;
    j2=0;m=E(i,1);n=E(i,2);
    if fp1(i,4)==0&&(ismember(m,biaohao)==0)
        dm1=inf;dm2=inf;
        for j=1:pts %找出距离 m 最近的平台 j1
            if dm1>dmin(biaohao(j),m)
                dm1=dmin(biaohao(j),m);
                j1=biaohao(j);
            end
        end
        for j=1:pts %找出距离 n 最近的平台 j2
            if dm2>dmin(biaohao(j),m)
                dm2=dmin(biaohao(j),m);
                j2=biaohao(j);
            end
        end

        if (dmin(j2,m)<30&&dmin(j1,n)<30)||(dmin(j2,m)>30&&dmin(j1,n)>30)
            fp1(i,4)=1;
            if gzl1(find(biaohao==j1),2)<=gzl1(find(biaohao==j2),2)
                fp1(i,5)=j1;
                fp1(i,6)=dmin(j1,n);
                gzl1(find(biaohao==j1),2)=gzl1(find(biaohao==j1),2)+S(m,n)*D(m,n);
            end
            if gzl1(find(biaohao==j1),2)>gzl1(find(biaohao==j2),2)
                fp1(i,5)=j2;
                 fp1(i,6)=dmin(j2,m);
                gzl1(find(biaohao==j2),2)=gzl1(find(biaohao==j2),2)+S(m,n)*D(m,n);
            end
        end
        if dmin(j2,m)>30&&dmin(j1,n)<30
            fp1(i,4)=1;
            fp1(i,5)=j1;
            fp1(i,6)=dmin(j1,n);
            gzl1(find(biaohao==j1),2)=gzl1(find(biaohao==j1),2)+S(m,n)*D(m,n);
        end
        if dmin(j2,m)<30&&dmin(j1,n)>30
            fp1(i,4)=1;
            fp1(i,5)=j2;
            fp1(i,6)=dmin(j2,m);
            gzl1(find(biaohao==j2),2)=gzl1(find(biaohao==j2),2)+S(m,n)*D(m,n);
```

```
            end
        end
    end



for i=1:t
    m=E(i,1);n=E(i,2);
    if fp1(i,4)==0&&ismember(n,biaohao)==1
        fp1(i,4)=1;
        if D(m,n)<30
            if gzl1(find(biaohao==m),2)<gzl1(find(biaohao==n),2)
                fp1(i,5)=m;
            else
                fp1(i,5)=n;
            end
                fp1(i,6)=D(m,n);
        end
        if D(m,n)>30
            fp1(i,5)=n*10+m;
            if gzl1(find(biaohao==m),2)<gzl1(find(biaohao==n),2)
                gzl1(find(biaohao==m),2)=gzl1(find(biaohao==m),2)+S(m,n)*30;
                gzl1(find(biaohao==n),2)=gzl1(find(biaohao==n),2)+S(m,n)*(D(m,n)-30);
            else
                gzl1(find(biaohao==m),2)=gzl1(find(biaohao==m),2)+S(m,n)*(D(m,n)-30);
                gzl1(find(biaohao==n),2)=gzl1(find(biaohao==n),2)+S(m,n)*30;
            end
        end
    end
    z=gzl1(:,2);
    z=z';
    std(z)
end


clc,clear
% 全市交通路线 4:权值
C=load('F:\数学建模\暑假培训\11B\数据\服务台位置编号.txt');
%F=load('F:\数学建模\暑假培训\11B\数据\A 区非平台节点编号.txt');
A1=[71;72;73];
A2=[89;90;91;87;82;84];
biaohao=zeros(22,1);
for k=1:20
    biaohao(k)=k;
end
```

```
for i=1:3
            C=[C;A1(i)];
            biaohao(21)=A1(i);
        for j=1:6
            C=[C;A2(j)];
            biaohao(22)=A2(j);
            q3(C,biaohao,22);
            break;
        end
End
```

附录五：问题二第一小问代码

```
tic    %¼Æ Ê±¿ªÊ¼
clc,clear
sj0=load('sj.txt');              %¼ÓÔØ100 ¸öÄ¿±êµÄÊý¾Ý
x=sj0(:,1:2:8); x=x(:);
y=sj0(:,2:2:8); y=y(:);
sj=[x y]; d1=[70,40];
sj=[d1;sj;d1]; sj=sj*pi/180;    %µ¥Î»¯Éè¡¶È
d=zeros(102); %¾àÀë¾ØÕódµÄ³õÊ¼Öµ
for i=1:101
  for j=i+1:102
    d(i,j)=6370*acos(cos(sj(i,1)-sj(j,1))*cos(sj(i,2))*cos(sj(j,2))+sin(sj(i,2))*sin(sj(j,2)));
  end
end
d=d+d'; w=50; g=100; %w ÎªÖÖÈº¹æ ¸öÊý£¬g Î¾ø»¯µÄ ´úÊý
rand('state',sum(clock)); % ³õÊ¼»¯Ëæ»úÊý·¢Éú¼ ÷
for k=1:w    % Í¨¹ý ÄÁ¼È Ë㷨Ñ ¦È ¡³õÊ¼ÖÖÈº
      c=randperm(100); % ²úÉú1 £¬... £¬100 µÄÒ» ¸öÈ«ÅÅÁÐ
      c1=[1,c+1,102]; %Éú³É ³õÊ¼½â
      for t=1:102 % Ã ²âÑ ·»·ÊÇÐÞ ÄÈ ¦
          flag=0; %ÐÞ ÄÈ ¦Ë ³ö±êÖ¾
      for m=1:100
        for n=m+2:101
          if d(c1(m),c1(n))+d(c1(m+1),c1(n+1))<d(c1(m),c1(m+1))+d(c1(n),c1(n+1))
              c1(m+1:n)=c1(n:-1:m+1);    flag=1; %ÐÞ ÄÈ ¦
          end
        end
      end
    if flag==0
        J(k,c1)=1:102; break %¼ÇÂ¼ ÏÂ½ ÏªÃ µÄ½â ³Ë Ë ³öµ±Ç °²âÑ ·»·
    end
    end
end
J(:,1)=0; J=J/102; % °ÑÕûÊýÐòÁÐ×ª»»³É[0,1]Çø¼ äÉ ÏµÄÊýÊ£¬¼´ ×ª»»³ÉÈ¾É«±ìâÂ ë
```

```
for k=1:g   % ¸Ã ²ãÑ»·½øÐÐÒÅ´«Ëã·¨µÄ¸Ù×÷
    A=J; %½»Åä²úÉú×Ó´úB µÄ³õÊ¼È¾É«Ìå
    for i=1:2:w
        ch1(1)=rand; %»ìãçÐòÁÐµÄ³õÊ¼Öµ
        for j=2:50
            ch1(j)=4*ch1(j-1)*(1-ch1(j-1)); % ²úÉú»ìãçÐòÁÐ
        end
        ch1=2+floor(100*ch1); % ²úÉú½»²æ¸Ù×÷µÄµØÖ·
        temp=A(i,ch1); %ÖÐ¼ä±äÁ¿µÄ±£´æÖµ
        A(i,ch1)=A(i+1,ch1); %½»²æ¸Ù×÷
        A(i+1,ch1)=temp;
    end
    by=[];   % ÎªÁË·ÀÖ¹ÏÂÃæ²úÉú¿ÕµØÖ·£¬Õâ ìïÈ ³õÊ¼»¯
while ~length(by)
    by=find(rand(1,w)<0.1); % ²úÉú±äÒì¸Ù×÷µÄµØÖ·
end
num1=length(by); B=J(by,:); % ²úÉú±äÒì¸Ù×÷µÄ³õÊ¼È¾É«Ìå
ch2=rand;   % ²úÉú»ìãçÐòÁÐµÄ³õÊ¼Öµ
for t=2:2*num1
        ch2(t)=4*ch2(t-1)*(1-ch2(t-1)); % ²úÉú»ìãçÐòÁÐ
end
for j=1:num1
    bw=sort(2+floor(100*rand(1,2)));   % ²úÉú±äÒì¸Ù×÷µÄ2 ¸öµØÖ·
    B(j,bw)=ch2([j,j+1]); %bw ´¦µÄÁ½ ¸ö»ùÒò·¢ÉúÁË±äÒì
end
    G=[J;A;B]; % ¸¸´úºÍ×Ó´úÖÖÈº ºÏÔÚÒ»Æð
    [SG,ind1]=sort(G,2); % ÑÈ¾É«Ìå·ÒëÉ1£¬...,102 µÄÐòÁÐind1
    num2=size(G,1); long=zeros(1,num2); %Â·¾¶³¤¶ÈµÄ³õÊ¼Öµ
    for j=1:num2
        for i=1:101
            long(j)=long(j)+d(ind1(j,i),ind1(j,i+1)); %¼ÆËãÃ¿ÌõÂ·¾¶³¤¶È
        end
    end
      [slong,ind2]=sort(long); %¶ÔÂ·¾¶³¤¶È°´ÕÕ´ÓÐ¡µ½´óÅÅÐò
      J=G(ind2(1:w),:); %¾«Ñ¡³öw ¸ö½Ï¶ÌµÄÂ·¾¶¶ÔÓ¦µÄÈ¾É«Ìå
end
path=ind1(ind2(1),:), flong=slong(1)   %½âµÄÂ·¾¶¼°Â·¾¶³¤¶È
toc   %¼ÆËãÊ±¼ä
xx=sj(path,1);yy=sj(path,2);
plot(xx,yy,'-o') %»³öÂ·¾¶
```

附录六：问题二第二小问代码

```
#include<stdio.h>

#define MAX_VERTEX_NUM 20
```

```
#define INFINITE 10000   //当做无穷大
#define Num_PingTai 4
#define Num_CuKou 2
#define Num_JieDian 6
#define v 0.6


char pingtai[Num_PingTai]={'A','B','C','D'};
char chukou[Num_CuKou]={'A','E'};
double ToPingShorst[Num_PingTai-Num_CuKou];
char lujing[Num_PingTai-Num_CuKou][Num_JieDian]={};
int NUM=0;

double amin[Num_JieDian][Num_JieDian]={{1,2,3,4,5,6},{1,1,1,1,1,1},{2,2,2,2,2,2},{3,3,3,3,3,3},{4,4,4,4,4,4},{5,5,5,5,5,5}};//读取数
据

//图的定义
typedef struct
{
        int vertexNum;
        char vertex[MAX_VERTEX_NUM];
        int arc[MAX_VERTEX_NUM][MAX_VERTEX_NUM];
}Graph,*PGraph;

//辅助数组中的元素定义
typedef struct
{
        int distance;
        int path[MAX_VERTEX_NUM];
}ArrayNode;


//构造有向网
void createdGraph(PGraph g)
{
        int i,j;
      g->vertexNum=6;
      for(i=0;i<g->vertexNum;i++)
                g->vertex[i]='A'+i;
        for(i=0;i<g->vertexNum;i++)
                for(j=0;j<g->vertexNum;j++)
                  g->arc[i][j]=0;
        g->arc[0][2]=10;
        g->arc[0][4]=30;
```

```
    g->arc[0][5]=100;
    g->arc[1][2]=5;
    g->arc[2][3]=50;
    g->arc[3][5]=10;
    g->arc[4][3]=20;
    g->arc[4][5]=60;
    g->arc[3][4]=20;
}


//迪杰斯特拉算法
double Dijkstra(PGraph g,int from,int to,int num)
{
    int i,j,index=-1;
    int n=1;//记录已经求出的两个点之间的最短距离的个数
    int flag[MAX_VERTEX_NUM]={0};//标记，为 1 表示到这个顶点的最短距离已求出
    ArrayNode shortestPath[MAX_VERTEX_NUM];
    //1.求 from 到各个顶点的直接距离，即初始化 shortestPath 数组
    for(i=0;i<g->vertexNum;i++){
        if(from==i){
            shortestPath[i].distance=0;
            shortestPath[i].path[0]=i;
            flag[from]=1;
        }
        else if(g->arc[from][i]>0){
        shortestPath[i].path[0]=from;
        shortestPath[i].path[1]=i;
            shortestPath[i].distance=g->arc[from][i];
        }else
        shortestPath[i].distance=INFINITE;
    }
    //2.每次求一个最短路径
    while(n<g->vertexNum){
        //选择 shortestPath 中距离最小的，求出 from 到这个顶点的最短路径
        index=-1;
        for(i=0;i<g->vertexNum;i++){
            if(i==from)
                continue;
            if(flag[i]==0 && index==-1 && shortestPath[i].distance!=INFINITE)
                index=i;
            if(flag[i]==0 && index!=-1 && shortestPath[i].distance<shortestPath[index].distance)
                index=i;
        }
        flag[index]=1;
```

```
            //修改到各个顶点的最短路径
            for(i=0;i<g->vertexNum;i++){
                if(i==from)
                    continue;
                if(g->arc[index][i]>0 && g->arc[index][i]+shortestPath[index].distance<shortestPath[i].distance){
                    shortestPath[i].distance=g->arc[index][i]+shortestPath[index].distance;
                    //修改路径
                    j=0;
                while(1){
                        shortestPath[i].path[j]=shortestPath[index].path[j];
                        if(shortestPath[index].path[j]==index)
                            break;
                        j++;
                    }
                shortestPath[i].path[j+1]=i;
                }
            }
            n++;
        }
        //输出 from 到 to 的最短路径及长度
        if(shortestPath[to].distance==INFINITE){
            printf("%c 到%c 没有路径\n",from+'A',to+'A');
            return 0;
        }
        printf("%c 到%c 的最短路径长度是：%d\n",from+'A',to+'A',shortestPath[to].distance);
        ToPingShorst[num]=shortestPath[to].distance;
        printf("经过的顶点：");
        i=0;
        while(1){
            printf("%c",shortestPath[to].path[i]+'A');
            lujing[num][i]=shortestPath[to].path[i]+'A';
            NUM++;
       if(shortestPath[to].path[i]==to)
                break;
            i++;
        }
        printf("\n");
        return shortestPath[to].distance;
}

//检查是否是平台，不是返回 1，是返回 0
int checkPT(char a){
        int j;
        for(j=0;j<Num_PingTai;j++){
```

```
            if(a==pingtai[j]){
                return 0;
            }
        }
        return 1;
}

//找到数组最小值 ， 返回下标
int findmin(double a[]){
        double temple;
        int i,t;
        temple=a[0];
        t=0;
        for(i=1;i<Num_PingTai-Num_CuKou;i++){
            if(temple>a[i]){
                temple=a[i];
                t=i;
            }
        }
        return t;
}

//找到距离最短的出口和路径
int findShoutestP(PGraph g,int from){
        int i,j,k=0;
        double temp;
        for(i=0;i<Num_CuKou;i++){
            if(checkPT(chukou[i])){
                temp=Dijkstra(g,from-'A',chukou[i]-'A',k);
                ToPingShorst[k]=temp;
                k++;
            }
        }
        return findmin(ToPingShorst);
}

int main()
{
        Graph graph;
        int i,b;
        int x,y,z;
        char from;
        scanf("%c",&from);
        createdGraph(&graph);
```

```
    for(i=0;i<Num_PingTai-Num_CuKou;i++){
        ToPingShorst[i]=INFINITE;
    }
    b=findShoutestP(&graph,from);
    for(i=0;i<NUM;i++){
        printf("路径经过结点%c",lujing[b][i]);
    }
    printf("\n 路径长%lf",ToPingShorst[b]);

    //判断 3min 后在哪两个结点之间
    for(i=0;i<NUM-1;i++){
        printf("%d",i);
        if(i=0){
            x=(int)lujing[i]-'A';
            y=(int)lujing[i+1]-'A';
            printf("x=%d y=%d",x,y);
            if((v*3<amin[x][y])){
            printf("%c 之前",lujing[i]);
            break;
            }
        }else{
           x=(int)lujing[i]-'A';
            y=(int)lujing[i+1]-'A';
            z=(int)lujing[i+2]-'A';
           if((amin[x][y]<=v*3)&&(v*3<amin[y][z])){
            printf("%c %c 之间",lujing[i+1],lujing[i+2]);
            break;
           }
        }
    }
    return 0;
}
```