全国第六届研究生数学建模竞赛

题 目 **110 警车配置及巡逻方案**

摘 要：

巡逻勤务是对社会治安进行动态管理的一种勤务方式,要求对社会面做到全时空控制。110 警车在街道上巡弋，既能够对违法犯罪分子起到震慑作用，降低犯罪率，又能够增加市民的安全感，同时也加快了接处警时间，提高了反应时效，为社会和谐提供了有力的保障。本文根据某一区域的数据地图，在满足所给要求 D1、D2、D3 的情况下，给出求解警车配置及巡逻方案。由于交叉点众多，导致线性方程组复杂难解，因此采用离散化策略，将地图细分为相邻点距离不超过 200 米图集。对于问题一我们考虑它的静态状况，将此问题转化问 P-中心问题，应用遗传算法求解。我们得到满足 D1 条件时该区最少需要配置 16 辆警车巡逻。对于问题二我们给出了评价巡逻效果显著程度的有关指标。而对于之后的问题，均牵扯巡逻问题，警车均为动态，我们采用贪婪策略来处理，得到了四种不同要求下的警车巡逻方案及其评价指标值。最后根据我们的模型和我们的理解，对当前警车巡逻等方面提出了一些具有实际意义的建议。

关键词： 巡逻方案；P-中心问题；遗传算法；贪婪策略

参赛队号 1190319

队员姓名 龚云路、张家健、周立刚

参赛密码 _____

（由组委会填写）

110警车配置及巡逻方案

一、 问题重述

110 警车在街道上巡弋，既能够对违法犯罪分子起到震慑作用，降低犯罪率，又能够增加市民的安全感，同时也加快了接处警（接受报警并赶往现场处理事件）时间，提高了反应时效，为社会和谐提供了有力的保障。

考虑某城市内一区域，为简化问题，假定所有事发现场均在下图的道路上。该区域内三个重点部位的坐标分别为：（5112，4806），（9126， 4266），（7434 ，1332）。

某城市拟增加一批配备有 GPS 卫星定位系统及先进通讯设备的 110 警车。设 110 警车的平均巡逻速度为 20km/h，接警后的平均行驶速度为 40km/h。警车配置及巡逻方案要尽量满足以下要求：

D1. 警车在接警后三分钟内赶到现场的比例不低于 90%；而赶到重点部位的时间必须在两分钟之内。

D2. 使巡逻效果更显著；

D3. 警车巡逻规律应有一定的隐蔽性。

请回答以下问题：

一. 若要求满足 D1,该区最少需要配置多少辆警车巡逻？

二. 请给出评价巡逻效果显著程度的有关指标。

三. 请给出满足 D1 且尽量满足 D2 条件的警车巡逻方案及其评价指标值。

四. 在第三问的基础上，再考虑 D3 条件，给出你们的警车巡逻方案及其评价指标值。

五. 如果该区域仅配置 10 辆警车，应如何制定巡逻方案，使 D1、D2 尽量得到满足？

六. 若警车接警后的平均行驶速度提高到 50km/h，回答问题三。

七. 你们认为还有哪些因素、哪些情况需要考虑？给出你们相应的解决方案。

二、背景与分析

警察勤务[1]是指警察机关完成警察任务，以最有效的方法，组织和运用警力，履行警察职责的一切有计划有规律的活动。纵观世界各国，警察勤务可分为五种：巡逻、临检、守望、值班、备勤。其中巡逻为最基本的勤务方式。警察巡逻勤务是

警察部门为完成自身任务的需要，运用巡逻工作方法，对社会面进行动态控制的一种勤务方式。当前，由于街面管理和防范工作相对滞后于社会经济的发展，各种隐藏的不安定因素增多，街面治安问题日趋复杂。一方面是不法分子在街面的违法犯罪活动越来越猖獗。街面的宽阔空间给不法分子以作案快、逃逸快的有利条件，所以一些不法分子越来越多地将作案地点选择在街面，严重威胁着老百姓的安全感。另一方面是街面刑事治安案件不断上升。这样就更显得进行街面巡逻的重要性及必要性。

要使巡逻勤务达到对社会治安进行良好动态管理的目的，就要求对社会面做到全时空控制。这样对巡区的设计与规划，就要根据城市的实际情况和需要，以及巡警的实际勤务能力来划分巡逻区域和各组巡逻范围、规定巡区的安全保卫等级、确定巡逻警力及编组、选择合适的巡逻执勤方式，最终形成适合的巡逻工作方案。

警察巡逻活动的目的有两个:一是即时打击犯罪；二是即时服务群众。其出发点都是追求“第一时间的有效快速反应”，即警情反应速度。警情反应速度，是指从接警到警察赶赴现场的时间。警情反应速度越快，制止和抓获犯罪的效率就越高。正是警情反应速度对犯罪的重要控制作用，使得世界第三次警务革命的立足点将警力重点摆在街面上，以缩短警察赶赴现场的时间。从城市治安安全角度而言，警情反应速度是考察城市治安控制力状况的重要指标。

随着我国三十年的改革开放，经济和社会状态都发生了很大的变化，各地社会治安形势日益动态化、复杂化，以往那种在计划经济体制下形成的，以静态管理为主的治安管理方式、方法和手段，已越来越不适应新形势的发展和治安状况的需求。因此，在目前新形势下急需治安管理方式、方法进行转变。另外，根据西方警界的研究结果，公众的公共安全感和满意度，很大程度上并非源于警察的破案数、破案率或其它业务行动的结果，因为大多数情况下，公众对这些结果没有直接的感受，他们的安全感和满意率往往来源于身边的巡逻警察[2]。因此，构建巡逻格局就是要扩大警察时空的覆盖面，使警察无处不在，无时不在。通过街面巡逻，加强社会面控制和威慑，维护公共安全和公共秩序；通过街面巡逻，及时发现和调节群众纠纷，开展社会救援活动，为群众排忧解难，树立人民警察的良好形象；通过街面巡逻，充分宣传、发动和组织群众参与社会治安管理，及时掌握治安动向，把专门工作与群众工作有机的结合起来，变有限警力为无限警力。

由于科技的发展，现代巡逻方式和巡逻所需配备设置都有了很大的改变。现在很多城市都用上了GPS巡逻车系统[3]。GPS巡逻车系统是公安部门指挥调度系统的一个重要组成部分，是一个功能完善、技术先进、设备可靠的系统，他的建立对车辆的指挥调度、实时监控、防盗反劫等方面起到积极的促进作用。通过警用GPS管理系统，可以实现对警车的统一监控、管理、指挥，提高警务工作的效率。实现了对警车的精确指挥调度、实时监控。保障了人民警察能够安全、迅速、准确地执行治安巡逻、治安防范的任务。

因此，该城市警方在增加一批配备有GPS卫星定位系统及先进通讯设备的110警车后，通过配置合理的巡逻路线，对于接报、处理治安、刑事案件的效率将会有很大的提高。

在对模型的建模和求解过程中，由于交叉点众多，导致线性方程组复杂难解，我们采用离散化策略，将地图划分为相邻距离不超过200米的点集。另外，对于三个重点部位，由于都不在道路上，我们假设这三个点和它们附近的交叉点都有直接通路可以到达。

对于问题一中的警车配置问题，由于D1条件中不牵扯警车的巡逻，所以我们可以在问题一中假设警车是处于静止的状态，进而可将此问题转化问P-中心问题求解。而对于之后的问题，均牵扯巡逻问题，警车均为动态，我们采用贪婪策略来处理。

二、问题假设

1.警车出行路线的路面状况应是通畅、良好的，每条道路都是可双向行驶的。

2.相邻两个交叉路口之间的道路是直线。

3.警车巡逻时应是正常的，如不因警车抛锚而耽搁正常巡逻，而且不考虑路面情况和天气状况对正常巡逻的影响。

4.警车配备有的GPS卫星定位系统及先进的通讯设备都是良好的、正常运行的。警车可通过这些设备随时获得整个城市的治安状况。

5.社会治安处于一个正常状态，如在每辆警车巡逻区域内不会同时有多起事件发生，即每次在警车接处警时，该巡逻区域内再有其它事件发生的概率很小。

6.本文中除了考虑3个重点部位外，其他地方都是一样的对待，不考虑重点要害区域和案发重点区域，巡逻频率在每个点上是同等处理。并且在时间上也考虑一样，不区分重点时段与其它时间。

7.在我们考虑警车巡逻路线时不考虑突发事件，或者不考虑处理事故对正常巡逻的影响。

8.对巡逻方式不做考虑，只考虑有先进通讯设备的110警车，以前的设备不做考虑。

9.在安排巡逻路线时，不考虑换班的影响。

三、符号说明

n：表示所给城市道路交叉口的总数目，即 n = 307；

v_i：表示数据表中给出的第 i 个城市道路交叉口，其中 $i=1,2,3,...,n$；

$e_{i,j}$：表示相邻两个交叉口 $v_i, v_j (i \neq j)$ 之间的道路，其中 $i, j = 1,2,3,...,n$；

$s_{i,j}$：表示道路 $e_{i,j}$ 的长度，其中 $i, j = 1,2,3,...,n$；

$d(u,v)$：表示从点 u 到达 v 的最近距离；

$d(u,S)$：若 S 是一非空点集，则 $d(u,S)$ 表示点 u 与 S 中最近的一个点的距离；

$V = \{v_i \mid i = 1,2,3,...,n\}$ 表示所有道路交叉口的集合（$|V| = n$）；

$E=\{e_{i,j}\mid i\neq j且i,j=1,2,3,...,n\}$ 表示所有相邻道路的集合；

$G(V,E)$：表示所给城市道路网络图；

p：满足巡逻要求时所需警车的数目（ $p<n$ ）；

x_i：巡逻中的第 i 辆警车，其中 $i=1,2,3,...,p$；

$X=\{x_1,x_2,....,x_p\}$：由 p 辆巡逻警车构成的集合（ $|X|=p$ ）；

$X\subset G$ ：表示集合 X 中的点可以在图 G 的顶点上或边上，即警车可以在道路 $e_{i,j}$ 上，也可以在道路交叉口 v_i 处；

N：把图 G 上的边按要求进行细分后得到的所有新的顶点总数目；

$L_{\max}^{t}$：表示警车接警后，t 分钟能走的最远路程；

$P(d(v,X)\le L_{\max}^{3})$：表示所有警车与任意点的距离小于 $L_{\max}^{3}$ 的概率，即一旦发生事故，警车接警后三分钟内赶到现场的比例；

a：表示坐标为（5112，4806）的重点部位；

b：表示坐标为（9126，4266）的重点部位；

y：表示坐标为（7434，1332）的重点部位；

$d(a,X)\le L_{\max}^{2}$：表示某一警车与重点部位 a 的距离不超过 $L_{\max}^{2}$；

$d(b,X)\le L_{\max}^{2}$：表示某一警车与重点部位 b 的距离不超过 $L_{\max}^{2}$；

$d(y,X)\le L_{\max}^{2}$：表示某一警车与重点部位 y 的距离不超过 $L_{\max}^{2}$；

四、模型的建立与求解

4.1 问题一的模型建立与求解

对本文的第一个问题，从理论上可以认为是利用最少的警车数实现满足约束要求的巡逻服务，对巡逻警车的位置设置的优化布局属于图论中的p-中心问题，是NP—难问题[8]，要设计一个多项式时间算法求其精确解是比较困难的。本文中提出了用遗传算法求解这一问题，从而得到近似解，该算法的全局高效搜索特性使得到的优化结果能接近全局最优解。

4.1.1 p-中心问题

p-中心问题[8]是一种选址问题，在实际中有很好的应用。对选址问题的研究已有十分丰富内容，按照候选点的特性分类(Scaparra[5])，可分为连续选址问题、离散选址问题和网络选址问题。网络选址问题要求在某个网络上设置若干个服务中心，以达到特定的目的。例如，城市中设置若干个119消防站，以使到达最远的报警点的时间尽可能的短。又如：物流配送中心的选址，以便配送商品到所有超市的总的距离最短。网络选址问题在物流设施规划、通讯系统设计等诸多领域具有十分广阔的应用背景。自Hakimi[6]1964年首次提出中心问题与中位问题以来，对网络选址问题的研究范围和领域不断拓展。

p-中心问题(p-center problem)在物流网络规划以及公共服务设施选址中具有广

泛的应用背景，最早是由Hakimi于1964年[7]提出，其一般定义是：设 X 为图 G 的一点集， $|X|=p, X=\{x_1,x_2,\cdots,x_p \mid x_i \in G, i=1,2,\cdots,p\}$，对任意给定函数 $f_i, \forall v_i \in V$，定义 $f(X)=\max_{v_i \in V}\{f_i(d(v_i,P))\}$，求： $X^*=\{x_1^*,x_2^*,\cdots,x_p^* \mid x_i^* \in G, i=1,2,\cdots,p\}$ 以及 r_p，满足 $r_p=f(X^*)=\min\{f(X):|X|=p, X\subset G\}$。若其中 $X\subset V$，则模型称为顶点p-中心问题；相应地，若其中 $X\subset G$，则称为完全p-中心问题(若无特别说明，以下简称中心问题)；若其中 $f_i, f(X)$ 定义在 G 上，则称为连续p-中心问题。在大部分文献中，f_i 定义为距离的某种线性函数（除非特殊说明，以下讨论均含此假定），即 $f_i=w_i d(x_i,X)+a_i$，其中，$\forall i, w_i$ 称为权重，a_i 称为常数项。当 $w_i=1$ 时称为不加权型。

针对本文要在城市道路网络中去搜索p个最少警车所处位置，是一个近似于加权的完全p-中心问题模型，这是一个NP—难问题。因此，在实际处理中，我们只能寻求简单启发式算法或近似算法来求解，本文中采用的是遗传算法求解。

4.1.2 遗传算法

遗传算法[10]是一种以自然选择和遗传理论为基础，将生物进化过程中适者生存规则与种群内部染色体的随机信息交换机制相结合的优化搜索算法。

设求解的优化问题为：

$$f:\prod_{i=1}[u_i,v_i]\to R$$

式中 $\prod_{i=1}[u_i,v_i]\subset R^m$，这里 $[u_i,v_i]$ 是第 i 个变量的范围，用遗传算法求解问题时，首先对问题的解进行编码，构成“染色体”，不同的染色体构成种群。每个染色体叫做种群的个体，每个个体根据适应函数有一个适应度，然后通过选择、交叉和变异算子构成新一代更好的种群，这样不断进化，直到求出问题的满意解。其基本执行过程如下：

(1) 随机选择 n 个初始个体 $x_0^1, x_0^2, \cdots, x_0^n$ ，每个个体有 L 个基因位置 $t=0$；

(2) 计算每个个体的适应度 $f(x_t^i), i=1,2,\cdots,n;$

(3) 选择操作 从种群中选择出 n 个个体，选择的方法有精英算法，赌轮盘算法等等，个体 x_0^i 在下一代中被选择的概率为 $f(x_t^i)\Big/\sum_{i=1}^{n} f(x_t^i)$，这 n 个新个体记为 $x_t^{'1}, x_t^{'2}, \cdots, x_t^{'n}$；

(4) 交叉操作：从 $x_t^{'1}, x_t^{'2}, \cdots, x_t^{'n}$ 中以相同的概率 P_c 独立地选出两个个体，在 $1\sim L-1$ 之间随机选择一个或多个交叉点进行交叉操作，在交叉点互换两个个体的码段，产生两个新个体。重复这一过程，直到形成新种群 $x_{t+1}^1, x_{t+1}^2, \cdots, x_{t+1}^n$

(5) 变异操作：按变异概率 P_m 随机地改变每个个体的每一个基因位．形成新一代种群 $x_{t+2}^1, x_{t+2}^2, \cdots, x_{t+2}^n$；

(6) 是否满足结束条件，如果满足则停止；否则 $t=t+1$，转(2)；

4.1.3 模型分析和求解

由于巡逻警车接警后的平均行驶速度为 40km/h，这也就意味着接警后，对于离警车的距离小于或等于 $L^3_{\max}(L^3_{\max}=2000m)$ 的事故点，巡逻警车接警后，完全能够在三分钟之内赶到事故现场。而对三个重点部位的事故，巡逻警车接警后必须在两分钟之内赶到重点部位现场，即巡逻警车离重点部位的距离在任意巡逻时刻都不能超过 $L^2_{\max}(L^2_{\max}=4000/3\,m)$.

通过计算寻找得知：重点部位 a 点附近的交叉路口为 $v_{101},v_{103},v_{110},v_{112}$ ，b 点附近的交叉路口为 v_{123},v_{141}，y 点附近的交叉路口为 v_{277} 。在这我们假设重点部位和它们附近的交叉点都有直接通路可以到达。因此，从邻近警车到达重点部位的距离，可以考虑为警车先到达与他们邻近的交叉口的距离与此交叉口与重点部位距离之和。

由以上约束条件的分析，我们可以得到如下使巡逻警车尽可能少的优化模型：

$$\begin{aligned}&\min\ p\\&\min_X F(X)\\&\text{s.t.:}\\&P(d(v,X)\le L^3_{\max})\ge 90\%,\\&d(a,X)\le L^2_{\max},\\&d(b,X)\le L^2_{\max},\\&d(y,X)\le L^2_{\max}.\end{aligned}\qquad(1)$$

其中：

$$F(X)=\max_{v_i\in V} d(v_i,X)\ ;\qquad(2)$$

$$P(d(v,X)\le L^3_{\max})=\frac{\sum_{v_i\in V,\text{且}\,d(v_i,X)\le L^3_{\max}} d(v_i,X)}{\sum_{i\ne j,i,j=1,2,\ldots,307} s_{i,j}}\ ;\qquad(3)$$

$d(a,X)\le L^2_{\max}$(或$d(b,X)\le L^2_{\max}, d(y,X)\le L^2_{\max}$)表示警车与重点部位a（或b，y）的距离不超过 $L^2_{\max}$ 。

把以上分析中的(2)和(3)两个式子代入模型(1)，这样就建立起一个在满足D1所给要求后，可求解巡逻警车数最少的优化模型。我们采用遗传算法对此模型进行求解。

因为警车配置及巡逻方案要尽量满足 D1，即要保证警车在接警后三分钟内赶到现场的比例不低于 90%；而赶到重点部位的时间必须在两分钟之内。在对本模型求解过程中，我们考虑的方法是把道路线段离散化，也即对城市道路段进行加点细分，使得任意一条新的道路段的距离设置在 100 米到 200 米之间。细分方法为：若对有道路直接连接的两个点 v_i,v_j 之间的距离 $s_{i,j}$ 大于 200 米，则用它来除以 200 并取其整数 $[s_{i,j}/200]$，即可在 v_i,v_j 这两个相邻道路交叉口间可取 $[s_{i,j}/200]$ 个点，这样就将 $s_{i,j}$ 平均分成 $[s_{i,j}/200]+1$ 份。

经过计算，我们发现三个重点部位都不在网络线段或定点上，我们为了简化求解过程，把离重点部位最近的顶点近似的当成重点部位来处理。警车在接警后的速度为 40km/h，则它在三分钟赶到现场所走的路程为 $40\times3/60=2km=2000$米。由于把它离散了，在某一时刻，我们把总体的接处警（接受报警并赶往现场处理事件）时间转变为离此刻警车位置的距离为 2000米 以内的所有点的总数 N_1（除了重复的点），则要求 $N_1/N>90\%$。而赶到重点部位的时间必须在两分钟之内，则在重点部位周围必须有警车的路程为 $40\times2/60=4/3km=1333.3$米。根据 D1 我们可以得到静态状态下的警车配置情况，由此得到满足 D1 时该区最少需要配置多少辆警车巡逻。

在对地图进行离散化后，最终形成 1350 个点（即 N=1350），使得任意相邻两点间的路程都在 100 到 200 米之间。

下面是求解 P-中心的遗传算法的适应度函数：

$$Fitness = rate + \beta_{satisfy} \tag{4}$$

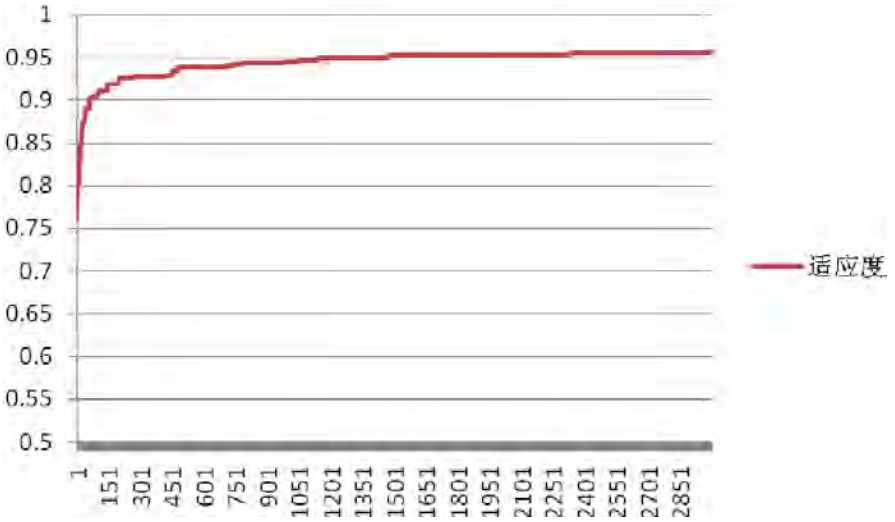
其中：

$rate$： 警车在接警后三分钟赶到现场的比例；

$\beta_{satisfy}$ ：如果满足两分钟内赶到重点部位，$\beta_{satisfy}$值为 0；如果不满足，加入罚值 0.5，即$\beta_{satisfy}=-0.5$;

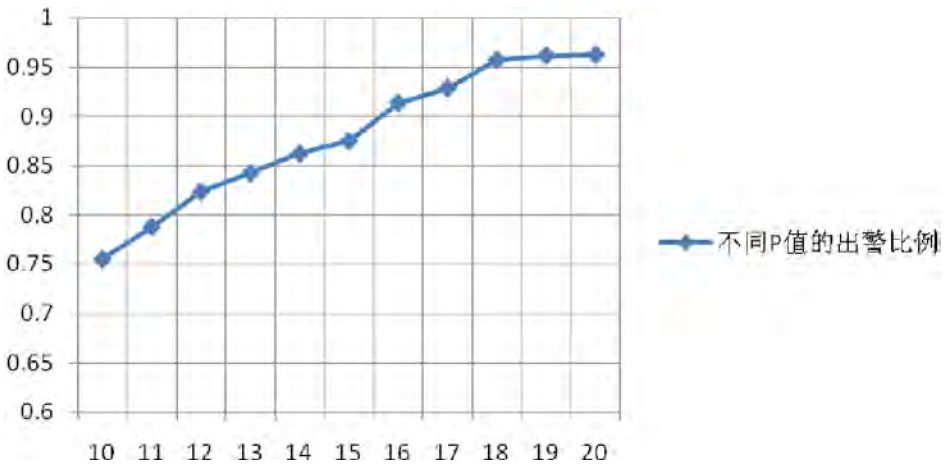
遗传算法的染色体个体编码采用一个 p 维数组，每一个数组内存放一个点；群体个数为 120 个；进化 3000 代；交叉概率为 0.5，变异概率为 0.01；

下面为遗传算法最优个体适应度进化的曲线图（以 p=18 为例）：



图（1）：适应度进化图

下面为 p 从 10 到 20 的最优解情况，其中每种情况均满足两分钟内赶到重点部位，纵坐标为警车在接警后三分钟赶到现场的比例：（具体数据详见附录）



图（2）：不同 P 值情况下警车在接警后三分钟赶到现场的比例

从图中可以看出，随着警车数量 p 的增加，警车在接警后三分钟赶到现场的比例*rate*不断增加。其中当 $p<16$ 时，*rate*的值小于 90%；当 $p=16$ 时，*rate*=0.9133。则我们认为满足 D1 时该区最少需要配置 16 辆警车巡逻。

4.2 问题二的模型建立与求解

评价巡逻效果显著程度与反应时间，工作负荷，单位警车出警效果，公众安全感，巡逻的区域覆盖率等相关。警察巡逻对辖区治安状况、发案率降低、人民群众安全感普遍增强起到了积极促进作用。在第二节中我们介绍了警情反应速度对犯罪的重要控制作用，是考察城市治安控制力状况的重要指标。在本文我们考虑的警情反应速度即反应时间也就是接处警时间。这里单位警车出警效果我们考虑警车巡逻经过的区域覆盖率。工作负荷是指出警的数量，在这里我们考虑的是警车的数目。

本文中，巡逻效果显著程度主要从下面三个方面衡量：

1. 警车接警后三分钟赶到现场的比例：因为这个指标直接与接警反应时效相关，所以比例越高，反应速度越快。

   警车接警后三分钟赶到现场的比例计算公式如下：

$$rate=\frac{\text{距最近警车距离小于 2km 的点数目}}{\text{城市内所有点数目（1350 个）}}$$

2. 警车巡逻经过的区域覆盖率：警车经过的区域越多，对违法犯罪分子的震慑作用越大，市民的安全感越强。

   警车巡逻经过的区域覆盖率计算如下：

$$cover\_rate = \frac{\text{警车巡逻经过的点数目}}{\text{城市内所有点数目（1350个）}}$$

3. 警车的数目：我们知道，只要增加警车数目，上述两个指标都能有效的增加，但警车数目的增加也代表着成本和工作负荷的增加，所以，我们要尽可能的少使用警车。

   警车数目定义如下：

$$car\_num = \text{警车数目}$$

从第一题的计算过程可以看出，警车数目在大于 16 到 20 的时候是可以满足 D1 的条件，所以，我们可以将警车数目的范围定在[16, 20]，下面是结合上述三种指标的综合指标：

$$indicator = rate + k_1 cover\_rate + k_2\left(1 - \frac{car\_num - \min\_num}{\max\_num - \min\_num}\right)$$

在这里，$\min\_num = 16$，$\max\_num = 20$，其中k_1是警车巡逻经过的区域覆盖率的权重，k_1越大表示警车巡逻经过的区域覆盖率越重要；k_2表示警车数目的权重，可以根据实际情况调整：在经济发达，或者重点保护的区域，增加警车的负担不是很重，k值可减少；在经济欠发达，或者不是很重要的区域，不需要很多警车巡逻的情况下，增加警车对指标的影响就较大，这是k值可适当增大。

我们认为综合指标越大，巡逻效果越显著。

4.3 问题三的模型建立与求解

4.3.1 贪婪算法

贪婪算法（Greedy Algorithm）是采用逐步构造最优解的方法，即在每一个阶段，都做出一个看上去最优的决策；决策一旦作出，就不可再更改。贪婪算法一般可以快速得到满意的解，因为它省去了为找最优解要穷尽所有可能而必须耗费的大量时间。

贪婪算法的五个组成部分是：

1）候选方案，有一套可以解决问题的候选方案
2）选择函数，选择最优候选方案
3）可行性函数，确定是否可用某个候选方案
4）目标函数，用于判定整体的或者某个阶段的结果好坏
5）结果函数，用于结束。

4.3.2 巡逻方案的建模与分析

在警车进行巡逻的时候，我们需要满足 D1，即警车在接警后三分钟内赶到现场的比例不低于 90%，赶到重点部位的时间在两分钟之内，然后我们针对问题二中提

出的指标值进行优化，得到巡逻方案。

在本问中，我们采用贪婪策略进行警车巡逻方案设计，具体如下：

1. 由第一问中所述方法，得到 P（P={16,17,18,19,20}）辆警车的初始最优位置。
2. 警车巡逻时，对下一时刻所有可能前进方向进行下述评价：

$$Score = rate + a_{90p} + b_{important} + c_{new}$$

其中：

$rate$：警车在接警后三分钟赶到现场的比例；

$$a_{90p} = \begin{cases} 1, & \text{警车在接警后三分钟赶到现场的比例达到 } 90\% \\ 0, & \text{其它} \end{cases};$$

$$b_{important} = \begin{cases} 2, & \text{三个重点部位均在两分钟之内到达} \\ 0, & \text{其它} \end{cases};$$

$$c_{new} = \begin{cases} c, & \text{下一方向为没有巡逻到的区域} \\ 0, & \text{其它} \end{cases};$$

在这里，描述“三个重点部位均在两分钟之内到达”的变量$b_{important}$的范围是[0，2]，描述“警车在接警后三分钟赶到现场”的变量（$rate + a_{90p}$）范围也是[0，2]，但由于$rate$的值很难到达 1，所以其实在这个策略，优先保证“三个重点部位均在两分钟之内到达”和“警车在接警后三分钟赶到现场”，然后“警车在接警后三分钟赶到现场的比例”越高的方向越优先。

c_{new}用来判断下一方向是否曾被巡逻到，由于警车巡逻经过的区域覆盖率越大越好，所以警车需要尽可能巡逻没有巡逻过的区域。这里的c值可以进行调整，c值过小，会导致警车经常在保证三分钟能赶到的区域大的地方徘徊，c值过大，警车巡逻经过的区域覆盖率会增大，但可能很难保证高反应时效。所以，我们需要尝试不同的c值以得到合适值。

3. 选择$Score$最高的方向作为前进方向。回到步骤 2，不断循环。

我们针对不同的车辆数目和不同的c值进行了计算，最后得到最优结果如下：

车辆数	是否满足 D1	接警三分钟赶到现场比例	巡逻到达区域覆盖率	最优 c 值范围
16	是	0.9177	0.1844	[0.001,0.0016]
17	是	0.9279	0.3711	[0.0018,0.0024]
18	是	0.9422	0.5733	[0.0026,0.003]
19	是	0.9508	0.6615	[0.0018,0.0024]
20	是	0.9505	0.7333	[0.0026,0.003]

按照第二问中提出的指标评价方法，我们对上述进行对了评价，得到结果如下：

Car_num	rate	Cover_rate	indicator
16	0.9177	0.1844	2.2865
17	0.9279	0.3711	2.4201
18	0.9422	0.5733	2.5888

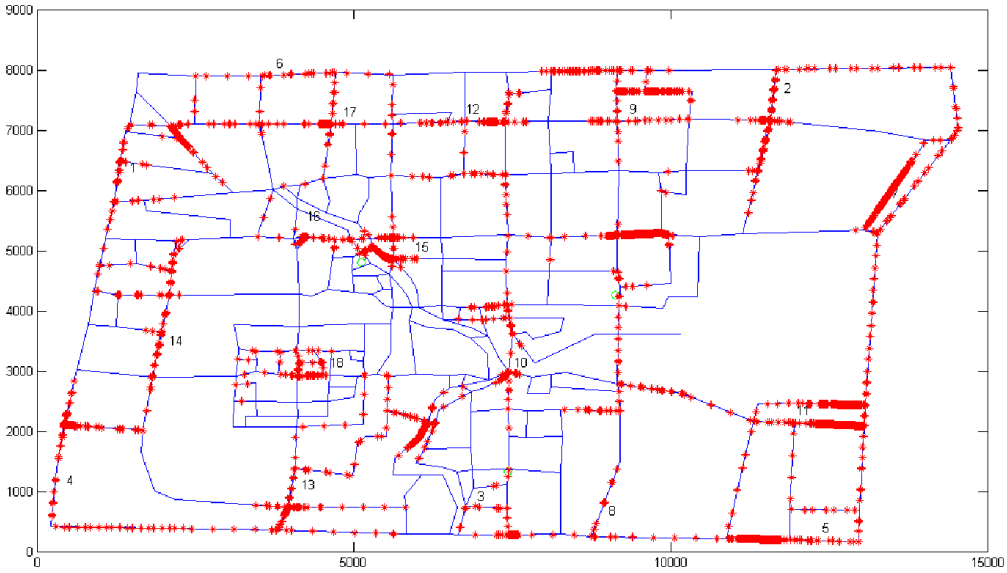
19	0.9508	0.6615	2.5238
20	0.9505	0.7333	2.4171

在这里，我们认为巡逻到达区域覆盖率对于巡逻效果显著程度很重要，所以取 $k_1 = 2$，而警车数目由于不了解所建模地图区域情况，所以做一般情况处理，取 $k_2 = 1$。

由上表可以看出，综合评价各项指标，当警车数为 18 辆这一情况时，巡逻方案较好，所以，我们的巡逻方案及其评价指标值如下：

车辆数目	18
巡逻方案	见上述算法
接警三分钟赶到现场比例	94.22%
重点部位两分钟赶到比例	100%
巡逻到达区域覆盖率	57.33%
综合指标评价	2.5888

巡逻方案的效果如图所示，其中红色部分代表巡逻覆盖区域：



图（3）：问题三的巡逻方案效果图

4.4 问题四的模型建立与求解

D3 条件要求警车巡逻规律应该有一定的隐蔽性，这里就要求警车的巡逻路线不能一成不变，警车线路不确定性强。行车线路和时间难以捉摸。所以，我们考虑在巡逻路线中加入随机因素，即在满足 D1 条件后，警车的巡逻有一定概率是考虑增加巡逻效率，也有一定概率是随机巡逻，增加隐蔽性。

增加隐蔽性的警车巡逻算法如下：

1. 由第一问中所述方法，得到 P（P={16,17,18,19,20}）辆警车的初始最优位置。
2. 警车巡逻时，对下一时刻所有可能前进方向进行下述评价：

$$Score = rate + a_{90p} + b_{important}$$

其中：

$rate$：警车在接警后三分钟赶到现场的比例；

$$a_{90p} = \begin{cases} 1, & \text{警车在接警后三分钟赶到现场的比例达到 90\%} \\ 0, & \text{其它} \end{cases};$$

$$b_{important} = \begin{cases} 2, & \text{三个重点部位均在两分钟之内到达} \\ 0, & \text{其它} \end{cases};$$

$$c_{new} = \begin{cases} c, & \text{下一方向为没有巡逻到的区域} \\ 0, & \text{其它} \end{cases};$$

3. 选定一个概率值σ，警车有σ的概率做随机运动，$1-\sigma$的概率优先向未巡逻到区域运动，具体方法如下：
   a. 随机一个数值k，$k \in (0,1)$
   b. 如果$k < \sigma$，随机选择一个可能前进方向，增加这个方向的评价分

$$Score = Score + c$$

   如果$k \geq \sigma$，对每一个可能前进的方向使用问题三中的方法评价：

$$Score = Score + c_{new}$$

   其中

$$c_{new} = \begin{cases} c, & \text{下一方向为没有巡逻到的区域} \\ 0, & \text{其它} \end{cases}$$

在这里，σ可以进行调整，σ越大，隐蔽性越强，但巡逻效果显著性会下降。

4. 选择$Score$最高的方向作为前进方向。回到步骤 2，不断循环。

我们将从结果差异的角度来评价隐蔽性，我们根据上述算法产生多套巡逻方案，如果这些方案差异越大，表明隐蔽性越高。

下面是根据上述算法产生的三套巡逻方案，这里σ=0.2：

方案	车辆数	满足 D1	接警三分钟赶到现场比例	巡逻到达区域覆盖率	与方案 1 差异率
1	18	是	0.9332	0.6504	-
2	18	是	0.9303	0.7556	78.67%
3	18	是	0.9409	0.6681	81.4%

我们可以看到，方案 2 和 3 与方案 1 之间的差异率均很高，说明这些巡逻方案有较强隐蔽性。

4.5 问题五的模型建立与求解

如果只有 10 辆警车，则 D1 条件很难完全满足，所以需要对第三问中的方法进行一些修改。修改后算法如下：

1. 由第一问中所述方法，得到 10 辆警车的初始最优位置。
2. 警车巡逻时，对下一时刻所有可能前进方向进行下述评价：

$$Score = rate_{normal} + 3 * rate_{important} + c_{new}$$

其中：

$rate_{normal}$：警车在接警后三分钟赶到现场的比例；

$rate_{important}$：重点部位在两分钟之内到达的比例；

$$c_{new} = \begin{cases} c, & \text{下一方向为没有巡逻到的区域} \\ 0, & \text{其它} \end{cases};$$

3. 选择$Score$最高的方向作为前进方向。回到步骤 2，不断循环。

这里$rate_{important}$前面有个权重 3 表明巡逻方案的重点是先保证重点部位在两分钟之内警车可以到达，然后是警车在三分钟内能赶到现场的区域尽可能大，最后是提高巡逻效果显著程度。

我们针对不同的c值进行了计算，最后得到结果如下（部分较优结果）：

标号	车辆数	是否满足重点部位在两分钟之内到达	接警三分钟赶到现场比例	巡逻到达区域覆盖率	c 值范围
1	10	是	0.6599	0.8319	(0.00114,0.00120]
2	10	是	0.6667	0.7393	(0.00120,0.00126]
3	10	是	0.6892	0.8207	(0.00126,0.00134]
4	10	是	0.6525	0.8044	(0.00134,0.00142]

按照第二问中提出的指标评价方法，我们对上述结果进行了评价，由于车辆数都一样，所以车辆数目的影响在这里不考虑，得到结果如下：

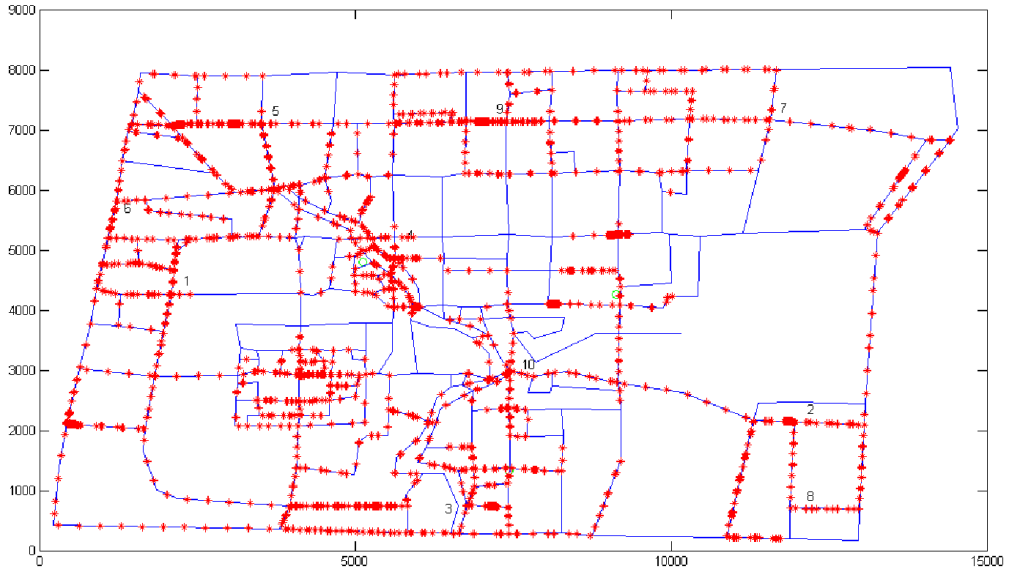
标号	Car_num	rate	Cover_rate	indicator
1	10	0.6599	0.8319	1.4918
2	10	0.6667	0.7393	1.406
3	10	0.6892	0.8207	1.5099
4	10	0.6525	0.8044	1.4569

在这里，由于题目是要求尽量满足 D1 和 D2，所以我们认为 D1 和 D2 一样重要，取$k_1 = 1$，而车辆情况不考虑，$k_2 = 0$。

由上表可以看出，综合评价各项指标，第三种情况（即$c \in (0.00126, 0.00134]$时）巡逻方案较好，所以，我们的巡逻方案及其评价指标值如下：

车辆数目	10
巡逻方案	见上述算法
接警三分钟赶到现场比例	68.92%
重点部位两分钟赶到比例	100%
巡逻到达区域覆盖率	82.07%
综合指标评价	1.5099

巡逻的效果如下图所示：

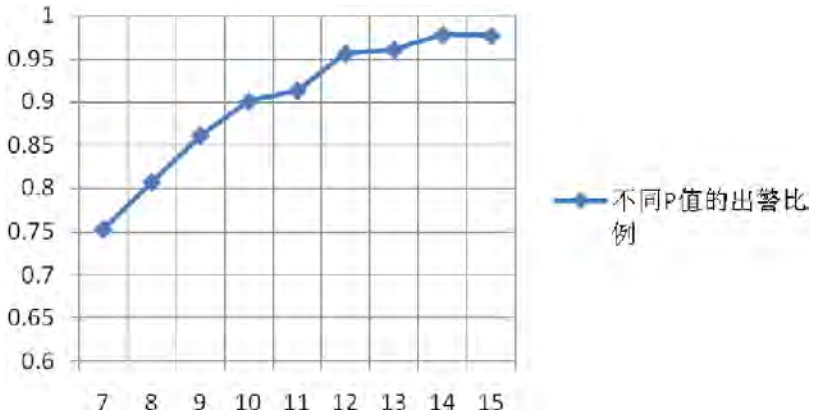


图（4）：问题五的巡逻方案效果图

4.6 问题六的模型建立与求解

若警车接警后的平均行驶速度提高到 50km/h，我们需要重新考虑警车的配置问题，即在平均行驶速度 50km/h 的情况下不同 P 值的最优解。

我们采用同问题一相同的方法，得到结果如下（具体数据见附录）：



图（5）：平均行驶速度 50km/h 情况下不同 P 值情况下警车在接警后三分钟赶到现场的比例

与问题三中的方法相同，我们针对不同的车辆数目和不同的 c 值进行了计算，最后得到最优结果如下：

车辆数	是否满足 D1	接警三分钟赶到现场比例	巡逻到达区域覆盖率	最优 c 值范围
10	否	-	-	-
11	是	0.9219	0.2956	[0.0022,0.0024]
12	是	0.9401	0.6615	[0.0032,0.0038]
13	是	0.9326	0.8519	[0.0032,0.0038]
14	是	0.9355	0.9311	[0.0048,0.0052]
15	是	0.9337	0.9615	[0.0062,0.0068]

按照第二问中提出的指标评价方法，我们对上述进行对了评价，得到结果如下：

Car_num	rate	Cover_rate	indicator
10	-	-	-
11	0.9219	0.2956	3.7631
12	0.9401	0.6615	4.2631
13	0.9326	0.8519	4.3864
14	0.9355	0.9311	4.2977
15	0.9337	0.9615	4.1067

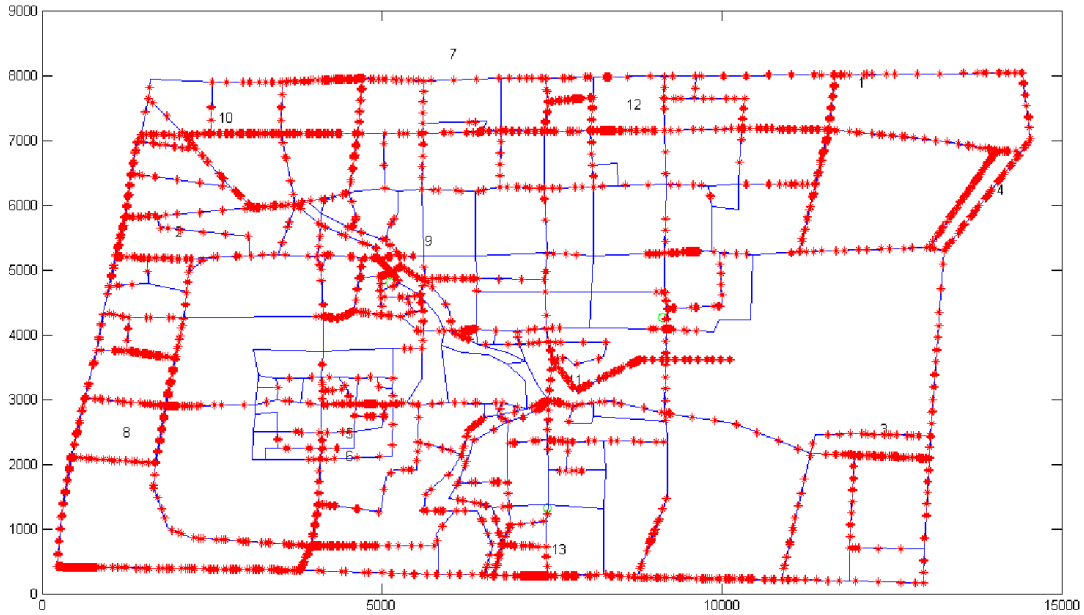
在这里，警车数目为 10 辆时由于巡逻时不能满足 D1，所以不予考虑；从剩下几种情况的计算情况来看，警车数目为 13 辆时最优。（注：这里警车数目虽然不再问题三提出的指标模型里警车数目的范围内，但不影响计算。）

综上，问题六的巡逻方案及其评价指标值如下：

车辆数目	13
巡逻方案	见问题三描述算法
接警三分钟赶到现场比例	93.26%

重点部位两分钟赶到比例	100%
巡逻到达区域覆盖率	85.19%
综合指标评价	4.3864

巡逻的效果如下图所示：



图（6）：问题六的巡逻方案效果图

4.7 问题七的分析与解答

对不同城市警车实际配置及巡逻方案，我们认为还需考虑现实生活中当地的治安状况是否良好，以及当地的经济社会发展程度。因为治安状况不同，市民对城市的安全感不同，事件发生的频率也不同，因而对治安维护及巡逻密度的需求将会不同。配备先进设施的警车及其它相关设备是需要考虑购置成本及维护费用的，因此警车的配备数量还要跟当地的经济发展相适应。

另外，城市中不同的设施及道路场所，所需的安全保卫级别也不会完全相同的，需要有不同的侧重。具体的方案就是对城市设施划定不同的安全级别，并设定相应巡逻方案。

此外，不同区域的人口密度也是设定巡逻方案时该考虑的因素，在人口稀少、分布广泛的地方,警察巡逻只具有较低的重要性。但随着小镇变为大城市，城市随着人口的密集不断扩张，警察巡逻的理由及现实需要越来越明显。人口密度会直接影响治安事件及刑事案件发生的几率。因此，对于人口密度高的地方，我们需要加大警力配备，增强巡逻密度。

五、总结和改进

警察巡逻是一个既新又老,既稚嫩又成熟的研究领域。随着社会治安形势的日益动态化、复杂化，以往那种在计划经济体制下形成的，以静态管理为主的治安管理方式、方法和手段，已越来越不适应新形势的发展和治安状况的需求。本文就“110警车配置及巡逻方案”提出了自己的解决方案，合理的解决了警车的巡逻问题，得到了较优的可行解，有一定的现实意义。但由于遗传算法本身限制，很难得到精确最优解，以及我们在确定巡逻路线时使用的贪婪策略，这些都值得改进。如果我们能更详细的定义各种评价指标以及有足够的时间对各项指标之间的权重进行调整，相信能得到更优的结果。

六、参考文献

[1] 谢盈，当前巡逻勤务存在问题及设计构想，金卡工程：经济与法，02 期：89，2009 年.

[2] 王均平.警察组织机构设置的科学化[J].河南高等专科学校学报，06 期:25. 2000 年.

[3] 冯景超、师亚莉、宋建锋，基于 GPS 和 GIS 技术的警车实时监测系统设计，现代电子技术，第 18 期：21-22，2005 年。

[4] 王均平.警察组织机构设置的科学化[J].河南高等专科学校学报，(6):25,2000.

[5] Scaparra M P.Facilities,locations,customers: building blocks of locat ionmodels:a survey[R].U niversity of Pisa,2001. http://www.di.unipi.it/scaparra/resume html.

[6] Hakimi S L.Optimal locations of switching centers and the absolute centers and medians of agraph [J]. OperRes, 4:B-31,1970.

[7] HARKIMIL. Optimal locations of switching centers and the absolute centers and medians of a graph [J].Operations Research, 12(3):450-459, 1964.

[8] 蔡延光，钱积新，孙优贤. 受限p-中心的并行迭代算法[J]. 系统工程理论与实践, 20(7): 1-6, 2000.

[9] 阎新芳,胡华东等. 受限 p-中心的遗传算法及其应用. 计算机程,32(4):33-35 2006 年

[10] 陈国良，王熙法，庄镇泉等. 遗传算法及其应用[M].人民邮电出版社, 1996.

[11] 王小平, 曹立明，遗传算法：理论、应用与软件实现，西安：西安交通大学出版社，2002 年。

[12] 雷英杰,、张善文,、李续武，MATLAB 遗传算法工具箱及应用，西安：西安电子科技大学出版社，2005 年。

[13] 宋兆基、徐流美等，MATLAB6.5 在科学计算中的应用，北京：清华大学出版社，2005 年。

[14] 杨丰梅等.选址问题研究的若干进展[J].运筹与管理, (14):1-7, 2005.

七、附录 A：

1. 问题一中不同 P 值情况下警车在接警后三分钟赶到现场的比例

车辆数目	三分钟赶到现场比率
10	0.755555556
11	0.788148148
12	0.823703704
13	0.842222222
14	0.862222222
15	0.874814815
16	0.913333333
17	0.928148148
18	0.957037037
19	0.961481481
20	0.962222222

2. 问题六中不同 P 值情况下警车在接警后三分钟赶到现场的比例

车辆数目	三分钟赶到现场比率
7	0.752593
8	0.807407
9	0.861481
10	0.900741
11	0.913333
12	0.956296
13	0.960741
14	0.977778
15	0.977037

附录 B （部分关键代码）：

1. 求解问题一使用的弗洛伊德算法代码（Matlab）：

```
for v=1:307
    for w=1:307
        for u=1:307
            lagmap(v,w,u)=-1;
            if lagroad(v,w)<2900000
                lagmap(v,w,v)=w;
            end
        end
    end
end

for u=1:307
    for v=1:307
        for w=1:307
            if (lagroad(v,u)+lagroad(u,w))<lagroad(v,w)
                lagroad(v,w)=lagroad(v,u)+lagroad(u,w);
                for i=1:307
                    if lagmap(v,u,i)~=-1
                        lagmap(v,w,i)=lagmap(v,u,i);
                    else
                        lagmap(v,w,i)=lagmap(u,w,i);
                    end
                end
            end
        end
    end
    u
end
```

2. 求解问题一使用的遗传算法代码（C#）：

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Data;
using System.IO;

namespace GAD2
{
```

```
9)      class Program
10)     {
11)         static int generation = 3000;
12)         static int population = 120;
13)         static double Pc = 0.5;
14)         static double Pm = 0.01;
15)
16)         static int pointNum = 1350;
17)         static int len;
18)         StreamReader reader;
19)         double[,] lagroad;
20)
21)
22)         static int[,] indiv; //个体
23)         static int[,] new_indiv; //下一代个体
24)         static double[] score; //适应度
25)         //static int[] score_indiv; //每个适应度对应的个体
26)         Random rand;
27)
28)         public void readMap()
29)         {
30)             lagroad = new double[pointNum, pointNum];
31)             reader = File.OpenText("C:\\book3.txt");
32)             string thisLine = null;
33)             int i = 0;
34)             while ((thisLine = reader.ReadLine()) != null)
35)             {
36)                 string[] line = thisLine.Trim().Split('\t');
37)                 for (int j = 0; j < pointNum; j++)
38)                 {
39)                     lagroad[i, j] = Double.Parse(line[j]);
40)                 }
41)                 i++;
42)             }
43)             reader.Close();
44)         }
45)
46)         public void initial()
47)         {
48)             indiv = new int[population, len];
49)             new_indiv = new int[population, len];
50)             score = new double[population];
51)             rand = new Random();
52)             for (int i = 0; i < population; i++)
53)             {
54)                 for (int j = 0; j < len; j++)
55)                 {
56)                     indiv[i, j] = rand.Next(0, 1349);
57)                 }
```

```
58)             }
59)         }
60)
61)         public void select()
62)         {
63)             //评分
64)             for (int i = 0; i < population; i++)
65)             {
66)                 score[i] = 0.00;
67)
68)                 int goodpoint = 0; //三分钟能赶到的点数
69)                 int[] goodimportant = { 0, 0, 0 }; //二分钟能赶到重点部位
    的点数
70)
71)                 for (int p = 0; p < pointNum; p++)
72)                 {
73)                     for (int p2 = 0; p2 < len; p2++)
74)                     {
75)                         //计算每个点和每个中心的距离,取最小距离
76)                         int temp = indiv[i, p2]; // 第p2个点
77)
78)                         if (lagroad[p, temp] < 2000.00)
79)                         {
80)                             goodpoint++;
81)                             break;
82)                         }
83)
84)                     }
85)
86)                 }
87)
88)                 score[i] += (double)goodpoint / (double)pointNum;
89)
90)                 for (int p2 = 0; p2 < len; p2++)
91)                 {
92)                     //计算每个点和每个中心的距离,取最小距离
93)                     int temp = indiv[i, p2]; // 第p2个点
94)                     if (lagroad[100, temp] < 1170.3364 || lagroad[102, temp]
    < 1178.4915 || lagroad[109, temp] < 1170.3364 || lagroad[111, temp] <
    1141.9907)
95)                     {
96)                         goodimportant[0] = 1;
97)                     }
98)                     if (lagroad[122, temp] < 1188.2127 || lagroad[140, temp]
    < 1145.4078)
99)                     {
100)                        goodimportant[1] = 1;
101)                    }
102)                    if (lagroad[276, temp] < 1333.3333)
```

```
103)                     {
104)                         goodimportant[2] = 1;
105)                     }
106)
107)                 }
108)
109)                 if (0 == goodimportant[0] || 0 == goodimportant[1] || 0
    == goodimportant[2]) score[i] -= 0.5;
110)
111)
112)             }
113)
114)
115)
116)             for (int i = 0; i < population; i++)
117)             {
118)                 int maxIndex = i;
119)                 for (int j = i + 1; j < population; j++)
120)                 {
121)                     if (score[maxIndex] < score[j]) maxIndex = j;
122)                 }
123)                 double temp = score[maxIndex];
124)                 score[maxIndex] = score[i];
125)                 score[i] = temp;
126)
127)                 int temp2 = 0;
128)                 for (int j = 0; j < len; j++)
129)                 {
130)                     temp2 = indiv[maxIndex, j];
131)                     indiv[maxIndex, j] = indiv[i, j];
132)                     indiv[i, j] = temp2;
133)                 }
134)             }
135)
136)             for (int i = 0; i < population / 3; i++)
137)             {
138)                 for (int j = 0; j < len; j++)
139)                 {
140)                     new_indiv[i, j] = indiv[i, j];
141)                 }
142)             }
143)         }
144)
145)         public void crossover()
146)         {
147)             for (int i = population / 3; i < 2 * population / 3; i++)
148)             {
149)                 int x1 = rand.Next(0, 39);
150)                 int x2 = rand.Next(0, 39);
```

```
151)
152)                 int[] gene1 = new int[len];
153)                 int[] gene2 = new int[len];
154)
155)                 for (int j = 0; j < len; j++)
156)                 {
157)                     if (rand.Next(0, 99) < Pc * 100)
158)                     {
159)                         gene1[j] = indiv[x2, j];
160)                         gene2[j] = indiv[x1, j];
161)                     }
162)                     else
163)                     {
164)                         gene1[j] = indiv[x1, j];
165)                         gene2[j] = indiv[x2, j];
166)                     }
167)                 }
168)
169)                 for (int j = 0; j < len; j++)
170)                 {
171)                     new_indiv[i, j] = gene1[j];
172)                 }
173)
174)                 if ((i + 1) >= 2 * population / 3) break;
175)
176)                 for (int j = 0; j < len; j++)
177)                 {
178)                     new_indiv[i + 1, j] = gene2[j];
179)                 }
180)                 i++;
181)
182)             }
183)         }
184)
185)         public void mutation()
186)         {
187)             int i = 2 * population / 3;
188)             for (int j = 0; j < population / 3; j++)
189)             {
190)                 for (int p = 0; p < len; p++)
191)                 {
192)                     if (rand.Next(0, 99) < Pm * 100)
193)                     {
194)                         new_indiv[i, p] = rand.Next(0, 1349);
195)                     }
196)                     else
197)                     {
198)                         new_indiv[i, p] = indiv[j, p];
199)                     }
```

```
200)                 }
201)                 i++;
202)                 if (i >= population) break;
203)             }
204)         }
205)
206)         public void reproduct()
207)         {
208)             for (int i = 0; i < population; i++)
209)             {
210)                 for (int j = 0; j < len; j++)
211)                 {
212)                     indiv[i, j] = new_indiv[i, j];
213)                 }
214)             }
215)         }
216)
217)         public void print()
218)         {
219)             for (int i = 0; i < len; i++)
220)             {
221)                 Console.Write(indiv[0, i] + " ");
222)             }
223)             Console.WriteLine();
224)             Console.WriteLine(score[0]);
225)         }
226)
227)         static void Main(string[] args)
228)         {
229)             len = Convert.ToInt32(args[0]);
230)             Program pro = new Program();
231)             pro.readMap();
232)             pro.initial();
233)
234)             for (int i = 0; i < generation; i++)
235)             {
236)                 //Pm = 0.01 * (i / 500);
237)                 pro.select();
238)                 Console.WriteLine("generation " + i + ": " + score[0]);
239)                 pro.crossover();
240)                 pro.mutation();
241)                 pro.reproduct();
242)             }
243)             pro.select();
244)             pro.print();
245)
246)             Console.ReadKey();
247)         }
248)     }
```

```
249)  }
```

3. 求解问题三使用的贪婪算法代码（Matlab）：

```
1) newp=0.0000;
2) totalnum=18;
3) rangeindex=1;
4) rangetable=zeros(100,5); %巡逻到的点数，巡逻点比率，平均覆盖率，覆
   盖到90%的数目，三点全覆盖数目
5) for rangeindex=0:50;
6)     rangeindex
7) c13;
8) carRecord = zeros(1000,20);
9) patrolRecord = zeros(1350,2);
10)
11) carindex=1;
12) while carindex<1501
13)
14)     if mod(carindex,50)==0
15)         for ci=1:1350
16)             patrolRecord(ci,1)=0;
17)         end
18)     end
19)
20)     maxLong=0;
21)
22) for i=1:totalnum
23)     index=1;
24)     allSitu=zeros(4,5);
25)     for j=1:1501
26)         if newroad(j,1)==current(i,1)
27)             allSitu(index,1)=newroad(j,2);
28)             index=index+1;
29)         end
30)         if newroad(j,2)==current(i,1)
31)             allSitu(index,1)=newroad(j,1);
32)             index=index+1;
33)         end
34)     end
35)
36)     cur=current(i,1);
37)     for j=1:4
38)         num=0;
39)         if allSitu(j,1)>0
40)             current(i,1)=allSitu(j,1);
41)
42)             imp1=0;imp2=0;imp3=0;
43)             for p=1:totalnum
44)                 if newlagroad2(101,current(p,1))<1170.3364 ||
   newlagroad2(103,current(p,1))<1178.4915 ||
```

```
   newlagroad2(110,current(p,1))<1170.3364 ||
   newlagroad2(112,current(p,1))<1141.9907
45)                 imp1=1;
46)             end
47)             if newlagroad2(123,current(p,1))<1188.2127 ||
   newlagroad2(141,current(p,1))<1145.4078
48)                 imp2=1;
49)             end
50)             if newlagroad2(277,current(p,1))<1333.3333
51)                 imp3=1;
52)             end
53)         end
54)
55)         if imp1==0 || imp2==0 || imp3==0
56)             allSitu(j,3)=0;
57)         else
58)             allSitu(j,3)=2;
59)         end
60)
61)         for i1=1:1350
62)             for j1=1:totalnum
63)                 dis = newlagroad2(i1,(current(j1,1)));
64)                 onmap=0;
65)                 if dis<2000
66)                     onmap=1;
67)                     break;
68)                 end
69)             end
70)             if onmap==1
71)                 num=num+1;
72)             end
73)         end
74)
75)         if (num/1350)>0.9
76)            allSitu(j,2)=1;
77)         else
78)             allSitu(j,2)=0;
79)         end
80)
81)         allSitu(j,4)=num/1350;
82)       end
83)   end
84)   best=1;
85)   for p2=1:4
86)
   allSitu(p2,5)=allSitu(p2,2)+allSitu(p2,3)+allSitu(p2,4);
87)       if allSitu(p2,1)>0 &&
   patrolRecord(allSitu(p2,1),1)==0
88)         allSitu(p2,5)=allSitu(p2,5)+0.0002*rangeindex;
89)       end
```

```
90)         if allSitu(p2,5)>allSitu(best,5)
91)             best=p2;
92)         end
93)     end
94)
95)
    current(i,2)=current(i,2)+newlagroad2(cur,allSitu(best,1));
96)     if current(i,2)>maxLong
97)         maxLong=current(i,2);
98)     end
99)
100)        current(i,1)=allSitu(best,1);
101)
102)        carRecord(carindex,i)=allSitu(best,1);
103)
    patrolRecord(allSitu(best,1),1)=patrolRecord(allSitu(best,1
    ),1)+1;
104)
    patrolRecord(allSitu(best,1),2)=patrolRecord(allSitu(best,1
    ),2)+1;
105)    end
106)        carRecord(carindex,19)=allSitu(best,4);
107)        carRecord(carindex,20)=allSitu(best,3);
108)        carindex=carindex+1;
109)        hasOne=0;
110)
111)    for i=1:totalnum
112)        if maxLong-current(i,2)>200
113)
114)        hasOne=1;
115)        index=1;
116)        allSitu=zeros(4,5);
117)        for j=1:1501
118)            if newroad(j,1)==current(i,1)
119)                allSitu(index,1)=newroad(j,2);
120)                index=index+1;
121)            end
122)            if newroad(j,2)==current(i,1)
123)                allSitu(index,1)=newroad(j,1);
124)                index=index+1;
125)            end
126)        end
127)
128)        cur=current(i,1);
129)        for j=1:4
130)            num=0;
131)            if allSitu(j,1)>0
132)                current(i,1)=allSitu(j,1);
133)
134)                imp1=0;imp2=0;imp3=0;
```

```
135)            for p=1:totalnum
136)                if
   newlagroad2(101,current(p,1))<1170.3364 ||
   newlagroad2(103,current(p,1))<1178.4915 ||
   newlagroad2(110,current(p,1))<1170.3364 ||
   newlagroad2(112,current(p,1))<1141.9907
137)                    imp1=1;
138)                end
139)                if
   newlagroad2(123,current(p,1))<1188.2127 ||
   newlagroad2(141,current(p,1))<1145.4078
140)                    imp2=1;
141)                end
142)                if
   newlagroad2(277,current(p,1))<1333.3333
143)                    imp3=1;
144)                end
145)            end
146)
147)            if imp1==0 || imp2==0 || imp3==0
148)                allSitu(j,3)=0;
149)            else
150)                allSitu(j,3)=2;
151)            end
152)
153)            for i1=1:1350
154)                for j1=1:totalnum
155)                    dis = newlagroad2(i1,(current(j1,1)));
156)                    onmap=0;
157)                    if dis<2000
158)                        onmap=1;
159)                        break;
160)                    end
161)                end
162)                if onmap==1
163)                    num=num+1;
164)                end
165)            end
166)
167)            if (num/1350)>0.9
168)               allSitu(j,2)=1;
169)            else
170)                allSitu(j,2)=0;
171)            end
172)
173)            allSitu(j,4)=num/1350;
174)        end
175)    end
176)
177)    best=1;
```

```
178)      for p2=1:4
179)
   allSitu(p2,5)=allSitu(p2,2)+allSitu(p2,3)+allSitu(p2,4);
180)          if allSitu(p2,1)>0 &&
   patrolRecord(allSitu(p2,1),1)==0
181)
   allSitu(p2,5)=allSitu(p2,5)+0.0002*rangeindex;
182)          end
183)          if allSitu(p2,5)>allSitu(best,5)
184)             best=p2;
185)          end
186)      end
187)
188)
   current(i,2)=current(i,2)+newlagroad2(cur,allSitu(best,1));
189)      if current(i,2)>maxLong
190)         maxLong=current(i,2);
191)      end
192)
193)      current(i,1)=allSitu(best,1);
194)
195)      carRecord(carindex,i)=allSitu(best,1);
196)
   patrolRecord(allSitu(best,1),1)=patrolRecord(allSitu(best,1
   ),1)+1;
197)
   patrolRecord(allSitu(best,1),2)=patrolRecord(allSitu(best,1
   ),2)+1;
198)      end
199)   end
200)   if hasOne==1
201)      carRecord(carindex,19)=allSitu(best,4);
202)      carRecord(carindex,20)=allSitu(best,3);
203)      carindex=carindex+1;
204)   end
205)   end
206)
207)   range=0;
208)   for i=1:1350
209)      if patrolRecord(i,2)>0
210)         range=range+1;
211)      end
212)   end
213)
214)   rangetable(rangeindex+1,1)=range;
215)   rangetable(rangeindex+1,2)=range/1350;
216)
217)   reachrate=0;
218)   reach90=0;
219)   reachimport=0;
```

```
220)   for i=1:1000
221)      reachrate=reachrate+carRecord(i,19);
222)      if carRecord(i,19)>0.9
223)         reach90=reach90+1;
224)      end
225)      if carRecord(i,20)>0
226)         reachimport=reachimport+1;
227)      end
228)   end
229)   reachrate=reachrate/1000;
230)
231)   rangetable(rangeindex+1,3)=reachrate;
232)   rangetable(rangeindex+1,4)=reach90;
233)   rangetable(rangeindex+1,5)=reachimport;
234)
235)   end
```

4. 求解问题四使用的贪婪加随机算法代码关键部分（Matlab）：

```
1)
  allSitu(p2,5)=allSitu(p2,2)+allSitu(p2,3)+allSitu(p2,4);
2)       if allSitu(p2,1)>0 && rand(1)<threshold
3)          if rand(1)<(1/tn)
4)             allSitu(p2,5)=allSitu(p2,5)+0.0026;
5)          end
6)       else
7)       if allSitu(p2,1)>0 && patrolRecord(allSitu(p2,1),1)==0
8)          allSitu(p2,5)=allSitu(p2,5)+0.0026;
9)       end
10)       end
11)       if allSitu(p2,5)>allSitu(best,5)
12)          best=p2;
13)       end
14)    end
15)
```

5. 问题五和问题六使用代码与前面大同小异，不在附上，具体参加电子版。