交巡警服务平台的设置与调度

摘要：

本文中主要探讨了交巡警平台的分配问题，提出了交巡警平台设置和管辖分配模式的评价标准，改进了目前存在的明显不合理的平台设置方案。同时给出了围堵逃犯的最佳方案。

对于第一问的第一小题，我们采用了 Floyd 算法确定了任意两点间的最短路径，根据就近原则确定了每个交巡警平台的管辖范围。

对于第一问的第二小题，我们的目标是：封锁全部路口的时间最短，并且尽量缩短到达每一个封锁路口的时间。 我们采取了分枝限界的搜索算法确定了最优解。

对于第一问的第三小题，我们提出了衡量管辖分配方式的标准，即一个既考察出警时间又考察工作量均衡的评价函数。然后通过最小化评价函数的方式确定了最佳管辖分配方式。然后通过搜索所有可能增加的警局，利用评价函数确定警局增加的最佳方式。在求解过程中我们使用了带参数的贪心算法，获得一个较优的警局增加方式。

对于第二问的第一小题，我们提出了两个模型。模型一利用单个路口的综合安全评价值，确定了交巡警平台的设置合理程度，并提出了改善交巡警平台设置的方案；模型二利用衡量一个区域的整体评价函数，比较了各区的平台设置合理程度，提出利用跨区联动管理的解决方案。

对于第二问的第二小题，我们认为要用尽量短的时间把逃犯控制在最小区域内。通过比较警察和逃犯分别可以达到的区域，给出并证明了最佳围堵方案。

关键词：最短路径 分枝限界法 贪心算法 评价函数模型

目录

1 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平台。每个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（以下简称平台）、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。

我们将根据某事设置平台的相关情况进行数学建模。下面是五个具体的问题：

问题一：

（1）某市中心城区 A 有 20 个平台，现要为这些平台分别分配管辖范围，使得在发生突然事件时，能尽快地从交巡警平台赶到事发地。(警车的时速为 60km/h，时间尽量控制在 3 分钟之内）

(2)对于重大突发事件,现需要尽快调度 20 个平台封锁出入该区的 13 条交通要到，实际上一个平台的警力最多封锁一个路口。

（3）对于现有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，现需要确定增加平台的具体个数和位置。

问题二：

（1）针对全市的具体情况，按照设置平台的原则和任务，分析研究该市现有平台设置方案的合理性。如果有明显不合理，请给出解决方案。

(2)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，请给出调度全市平台警力资源的最佳围堵方案。

2 模型假设

1. 汽车（包括警察与犯罪嫌疑人的车）在公路的节点间沿直线运行，不考虑道路宽度，并且假设每一条道路均双向通行。没有红绿灯与交通堵塞的情况发生。
2. 汽车只能沿着马路行驶，不能行驶到马路以外的地方或者抄近路。不考虑汽车损耗或故障问题。
3. 汽车不会行驶到该城以外的地方,或者借道别的城市的道路往来于该城市的两点间。
4. 警察的时速为 60Km/h，并且接到命令后立刻出警。
5. 该城任一平台的警力最多封锁一个路口。
6. 事发案件均发生在路口节点，不考虑除此以外的案件。(注意到数据中给出的案发次数均为路口节点处的数据）
7. 逃犯速度和警车速度一样为 60Km/h，这样警车无法追击。
8. 每个区内的人口都均匀分布。

3 模型符号说明

1. V_p :警察的速度，$V_p = 60\text{km/h}$。
2. T_a:警察到达事故现场的最晚时间
3. T_{max}:警察封锁所有 13 个路口的最晚时间。

4. d_a:警察到达事故现场的最长距离。
5. d_{max}:警察封锁所有 13 个路口时，最晚达到的平台达到封锁路口的距离。
6. F: 衡量区域分配平衡度和出警时间的评价函数；
7. G_i：衡量第 i 个路口节点综合安全程度的评价函数。
8. H：衡量各区平台的综合评价函数
9. W_i：第 i 个平台的期望工作量。
10. U_i：第 i 个路口的发案率。
11. σ: 每个平台的期望出警量的标准差。
12. α，β：待定参数。
13. pi：指第 i 个路口节点。
14. dis(p_i, p_j)：指第 i 个路口节点和第 j 和路口节点最短路的距离
15. d_i：平台 i 到事发地的距离。
16. dis(i)：路口 i 到所对应的平台的距离。
17. ρ_i：路口 i 所在的区的人口密度。
18. area：每个区的面积。
19. population：每个区的人口。
20. avecapacity：每个平台接警量的平均值。

4 模型的建立与求解

首先我们建立给定两点之间最短路的距离：我们采用 Floyd 算法，是一种使用动态规划思想，用于寻找给定的加权图中顶点间最短路径的算法。其状态转移方程如下：

$$dis(i,j) = \min(dis(i,k) + dis(k,j), dis(i,j)\} \quad k \neq i,j$$

其中 dis(i,j)表示 i 到 j 的最短距离，k 不等于 i 或 j，时间复杂度为 $O(n^3)$。

通过运行Floyd算法,我们得到了A城区的任意两个节点间的最短路距离(Matlab代码见**附录一**)。

4.1 问题一（1）的建模

4.1.1 问题一（1）的分析

问题一（1）要求事发之后，警察要在最短时间内到达案发现场，即目标函数为：

$$\min\ T_a, \text{ 这就相当于 } \min\ d_a$$

约束条件为

$$V_p = 60\text{km/h}..$$

事实上,对于每一次的案发,派出最近的交巡警平台出警可以保证目标函数最优。于是我们检查已经得到的 A 区的最短路径，就能立刻判断出最快到达现场时间以及能不能在 3 分钟内到达现场：

$$d_a = \min\{dis(pc, pi)\}$$

其中 pc 为案发路口，$i \in [1,20]$, i 为整数（在 1-20 路口分布了 A 区的警察平台）

同时，由于$T_a = d_a / V_p$，这样可以直接计算得出到达现场的最短时间。

4.1.2问题一（1）的结论

问题的结论在下表中显示，其中最短距离的单位是100m,最短时间的单位是 s.

表格 1 每个路口对应的出警平台表

路口	出警平台编号	最短距离	最短时间	路口	出警平台编号	最短距离	最短时间
1	1	0	0	47	7	12.80625	76.83749
2	2	0	0	48	7	12.90202	77.41212
3	3	0	0	49	5	5	30
4	4	0	0	50	5	8.485281	50.91169
5	5	0	0	51	5	12.29317	73.75901
6	6	0	0	52	5	16.59433	99.56598
7	7	0	0	53	5	11.7082	70.24922
8	8	0	0	54	3	22.70886	136.2532
9	9	0	0	55	3	12.65899	75.95393
10	10	0	0	56	5	20.83697	125.0218
11	11	0	0	57	4	18.68154	112.0893
12	12	0	0	58	5	23.01889	138.1134
13	13	0	0	59	5	15.20864	91.25186
14	14	0	0	60	4	17.39244	104.3547
15	15	0	0	61	7	41.90202	251.4121
16	16	0	0	62	4	3.5	21
17	17	0	0	63	4	10.30776	61.84658
18	18	0	0	64	4	19.36315	116.1789
19	19	0	0	65	3	15.23975	91.4385
20	20	0	0	66	3	18.40203	110.4122
21	13	27.08314	162.4988	67	1	16.19417	97.16504
22	13	9.055385	54.33231	68	1	12.07107	72.42641
23	13	5	30	69	1	5	30
24	13	23.85372	143.1223	70	2	8.602325	51.61395
25	12	17.88854	107.3313	71	1	11.40312	68.41875
26	11	9	54	72	2	16.06226	96.37355
27	11	16.43303	98.59821	73	1	10.29611	61.77667
28	15	47.51842	285.1105	74	1	6.264982	37.58989
29	15	57.00525	342.0315	75	1	9.300538	55.80323
30	7	5.830952	34.98571	76	1	12.83607	77.01643
31	9	20.55716	123.343	77	19	9.848858	59.09315
32	7	11.40175	68.41053	78	1	6.403124	38.41875
33	8	8.276473	49.65884	79	19	4.472136	26.83282
34	9	5.024938	30.14963	80	18	8.062258	48.37355
35	9	4.242641	25.45584	81	18	6.708204	40.24922
36	16	6.082763	36.49658	82	18	10.79349	64.76095
37	16	11.18178	67.09069	83	18	5.385165	32.31099

38	16	34.05877	204.3526	84	20	11.75225	70.51348
39	2	36.82186	220.9312	85	20	4.472136	26.83282
40	2	19.14419	114.8651	86	20	3.605551	21.63331
41	17	8.5	51	87	20	14.65091	87.90547
42	17	9.848858	59.09315	88	20	12.94632	77.67793
43	2	8	48	89	20	9.486833	56.921
44	2	9.486833	56.921	90	20	13.02237	78.1342
45	9	10.95084	65.70507	91	20	15.9877	95.92622
46	8	9.300538	55.80323	92	20	36.01269	216.0761

我们可以发现，有如下几个路口节点是警察在 3 分钟内无法到达的（也就是上表中出警时间大于 180 秒的）：28,29,38,39,61,92。最长到达距离为 29 号路口节点的 5.7km,到警时间 342s.

我们可以清楚的从下表中看出每一个交巡警平台分管的路口：

表格 2　每个交巡警平台分管的路口表

路口	出警平台编号	路口	出警平台编号	路口	出警平台编号	路口	出警平台编号
1	1	57	4	46	8	37	16
67	1	60	4	9	9	38	16
68	1	62	4	31	9	17	17
69	1	63	4	34	9	41	17
71	1	64	4	35	9	42	17
73	1	5	5	45	9	18	18
74	1	49	5	10	10	80	18
75	1	50	5	11	11	81	18
76	1	51	5	26	11	82	18
78	1	52	5	27	11	83	18
2	2	53	5	12	12	19	19
39	2	56	5	25	12	77	19
40	2	58	5	13	13	79	19
43	2	59	5	21	13	20	20
44	2	6	6	22	13	84	20
70	2	7	7	23	13	85	20
72	2	30	7	24	13	86	20
3	3	32	7	14	14	87	20
54	3	47	7	15	15	88	20
55	3	48	7	28	15	89	20
65	3	61	7	29	15	90	20
66	3	8	8	16	16	91	20
4	4	33	8	36	16	92	20

4.2 问题一（2）的建模

4.2.1问题一（2）的分析

为最快可能封锁所有的出口，最基本的要求便是所有路口全部被封锁的时间尽可能的短，即 $\min T_{max}$，亦即 $\min\ d_{max}$（为了讨论的方便，我们以距离进行建模）。但是经过进一步的分析，我们提出了要求更加高的目标函数：

假设 13 个交巡警平台的警力到达各自指定封锁路口的距离为$d_i(i = 1,2,\ldots,13)$. 不妨设 $d_1 \geq d_2 \geq \cdots \geq d_{13}$.显然有$d_{max} = d_1$，我们的目标函数是:

$\min\ d_1$

$\min\ d_2$，约束条件：d_1已经达到最小

$\min\ d_3$，约束条件： d_1，d_2已经达到最小

…….

$\min\ d_i$，约束条件： d_1，d_2，…，d_{i-1}已经达到最小

上述目标函数的解释：在保证所有路口全部被封锁的时间最短的前提下（即要求最慢达到封锁出口的时间最短），保证第二慢到达封锁出口的时间最短，在这个基础上保证第三满到达封锁出口的时间最短，以此类推，要保证第 K 慢的到达时间最快（建立在第 1 到第 K-1 慢时间已经最优的情况下）。如此可以确定出 13 个出口对应的平台的安排最优。

4.2.2问题一（2）的算法

为保证上述目标的实现，我们使用采取搜索的思想，对每一种出警方式进行搜索。但是A_{20}^{13}的计算量在有限的时间内难以得出结论。我们对于 A 城区进行了分块，以 x 坐标 350 为界，左侧为 A 左区，右侧为 A 右区。以下我们采取分支限界法减小搜索空间。

通过对照最短路表发现，A 右区的平台不可能比 A 左区的平台更快的对 A 左区的出口进行封锁。于是我们对于 A 左区进行搜索，得出 A 左区的d_{max}，此时最慢的封锁点与之对应的平台被确定。

为了保证我们达成设定的目标，我们需要去掉这个封锁出口与对应的平台，再次进行搜索，得出的结论保证了第二慢的时间最优；然后再一次去掉第二慢的封锁出口与对应的平台，进行再一次的搜索……如此进行十三次的搜索与去点之后，再对 A 右区进行类似搜索，就能保证目标的达成。具体的 C++程序代码见附录二.

4.2.3问题一（2）的结论

问题的结论在下表中显示，其中最短距离的单位是 100 米，最短时间的单位是秒。

表格 3　封堵任务安排表

出入口	出警平台编号	距离	时间
29	7	80.1546	480.9276
12	10	75.8659	455.1954
14	16	67.4166	404.4996
28	15	47.5184	285.1104
38	2	39.8219	238.9314
24	12	35.9163	215.4978

22	11	32.6956	196.1736
21	14	32.6497	195.8982
30	8	30.6082	183.6492
48	5	24.7583	148.5498
16	9	15.3254	91.9524
23	13	5	30
62	4	3.5	21

实现封锁的时间为 480.93 秒

4.3 问题一（3）的建模

4.3.1问题一（3）的分析

在 4.1 中我们采取的办法是离事发地最近的交巡警平台前往事发地。虽然这样能够减少最长出警时间，但却会导致各个平台的期望工作时间过于不均衡；反之，一个过于平衡的工作量会导致出警时间的增加。也就是说，我们需要在工作量平衡与最长出警时间进行一个平衡，使得工作量越平衡，最长出警时间越短，这个管辖分配越是合理。为此我们引入评价函数 F。

首先，一个平台的期望工作量W_i：

$$W_i = \sum_{j=1}^{k} U_j ,$$

即期望工作量等于该平台管辖所有路口的发案率之和。其中：

σ: 每个平台的期望工作量的标准差，是衡量工作量均衡与否的标准。

d_a:警察到达事故现场的最长距离。（由于警察的速度恒定，为了方便起见，我们采用距离进行评价）

下面来确定警局分配评价函数$F(d_1, d_2 \ldots, d_{92}, \sigma)$：由于市民对于出警时间大于 3 分钟（即出警距离大于 3000 米）的容忍度较小，而对于工作量不均衡的的容忍度较大。有理由认为出警时间三分钟是一个重要的分界线，评价函数应对于两种情况进行不同的评判。

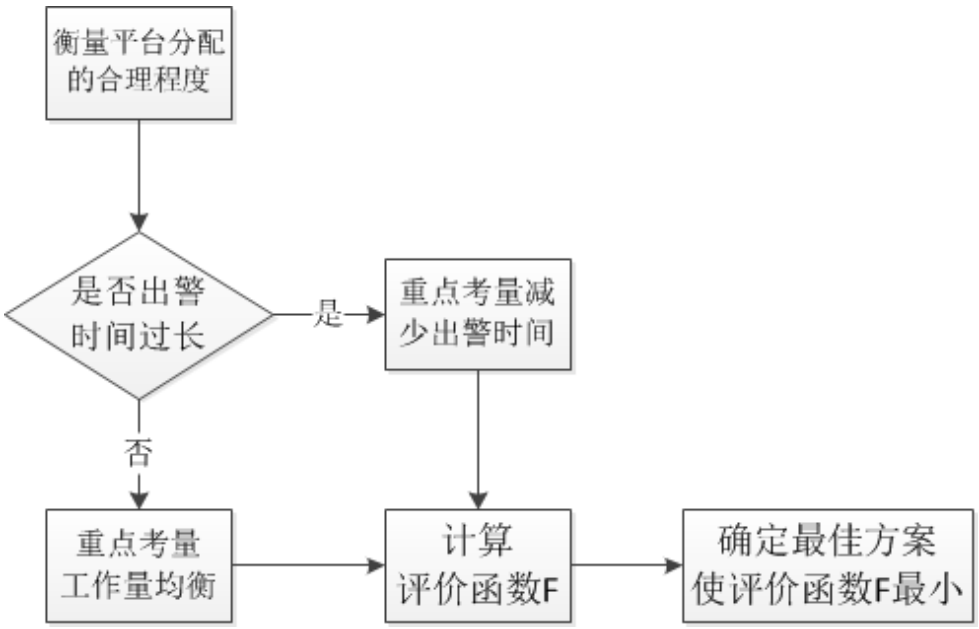

图 1　确定最佳平台分配方案方法流程图

交巡警平台分配标准评价函数

$$F=\begin{cases}\sum_{di>30}(d_i-30)*\sigma^{\alpha}, \text{当} d_a>30 \text{时（*）}\\ \left(1-\frac{100}{\sum_{i=1}^{92}d_i}\right)*\sigma^{\alpha}, \text{当} d_a\le 30 \text{时（**）}\end{cases}$$，其中α为参数。

我们的目标是$\min F$。上述评价模型的含义是在不能保证 3 分钟到达现场的情况下，将所有到场时间过长的道路节口进行超额时间的累加，这能刻画出出警时间不足的问题；而在能保证所有情况下均能 3 分钟到场的情况下，重点考量的是工作量的均衡程度。

下面我们来确定参数α：由于平台的最大作用是响应市区内的突发事件，所以评估函数 F 应该更加偏重于d_a而不是σ。往往市民对于出警时间的容忍度大于工作量是否均衡。所以我们认为$\alpha<1$ 。同时，我们的计算便利与评估函数量纲的协调性，我们选择$\alpha=1/2$ 。

4.3.2问题一（3）的求解

在新的评价标准 F 下，我们需要重新划分每一个平台的管辖范围。如果进行搜索，运算量应该在P_{72}^{20}左右，软件无法对于每一种情况进行计算 F 值后比较。于是我们采取了贪心的近似算法：

首先在 A 区中寻找距离平台最近的路口，将此路口分配给该平台。这样增加了一个警察的工作程度的不均衡程度，于是我们对于该平台进行调整。将所有路口和该平台的距离进行调整，乘以折算系数 K（K>1）。然后将原始最短路表中的距离调整完毕后，继续按照贪心算法将最近的路口分配给一个平台，然后继续调整该平台和所有路口的距离参数……以此类推，可以得出所有的分配方案。

折算系数 K 需要具有良好的性质，我们考虑 2 个警察站的工作量之比 Wi/Wj，我们认为在上述贪心算法中，工作量比相同的 2 个警察站的折算系数相同，于是考虑指数折算因子 K=β^{Wi}，$\beta>1$，Wi 是为 i 的工作站已经分配的工作时间。

我们要用一些数据来确定参数β的最优解。

下面我们使用了 21 个参考β值（1 到 2，每隔 0.05 测一个值；再 1.1 到 1.2，每隔 0.01 测一个值），然后按照上述贪心算法进行程序运算，对于每一个 β 都确定了一个分配方案和对应的 F 值，最后得出 21 个对应的 F 值。我们选取了 F 最小的值（对应的分配方案最佳）对应的参数β值。具体模型参数的选择结果见下表：

表格 4　参数 β 对应的 F 值表

β	F 值	β	F 值	β	F 值
1	124.9373	1.55	225.7745	1.1	115.9734
1.05	123.2174	1.6	225.7745	1.11	115.9734
1.1	115.9734	1.65	300.7952	1.12	115.9734
1.15	114.7863	1.7	297.0513	1.13	115.9734
1.2	147.9003	1.75	305.0059	1.14	114.7863
1.25	217.1618	1.8	399.6434	1.15	114.7863
1.3	217.1618	1.85	399.6434	1.16	114.7863
1.35	212.4195	1.9	399.6434	1.17	137.7909
1.4	272.7056	1.95	399.6434	1.18	147.9003
1.45	261.9748	2	399.6434	1.19	147.9003

1.5	261.9748			1.2	147.9003

最优的$\beta = 1.15$

下面我们使用贪心算法逐个增加平台：在每一次的添加平台过程中，我们尽可能的减少评价函数 F 的值，与此同时，如果最优情况的 F 存在多种添加平台方案，我们就计算（**）函数值，选择使得（**）值最小的方案（这意味着总路程较短）。

4.3.3问题一（3）的结论

表格 5　新增平台评价函数表

第几次添加	添加路口标号	评价函数 F 值	（**）函数值	选择
0		114.7863		
1	28	56.80797534	1.27694363	
	29	56.80797534	1.27694363	√
2	87	34.3720158	1.192869107	
	88	34.3720158	1.19052896	√
	89	34.3720158	1.191189947	
	91	34.3720158	1.191063881	
3	48	17.91100956		
4	38	3.38806664	1.121065431	
	39	3.38806664	1.12048041	√
	40	3.38806664	1.123866417	
5	21	1.178164461	1.178164461	
	22	1.133203388	1.133203388	√

于是我们得到最佳添加平台方案是添加 5 个平台，分别是：28,88,48,39,22 或者 29,88,48,39,22（实际上这两种方案本质上是相同的）

下面我们给出添加 29,88,48,39,22 号时平台任务分配的情况：

表格 6　添加后的出警平台任务表

路口	出警平台	出警距离	出警时间	路口	出警平台	出警距离	出警时间
1	1	0	0	47	48	10.19804	61.18823
2	2	0	0	48	48	0	0
3	3	0	0	49	5	5	30
4	4	0	0	50	5	8.485281	50.91169
5	5	0	0	51	5	12.29317	73.75901
6	6	0	0	52	6	23.24786	139.4871
7	7	0	0	53	5	11.7082	70.24922
8	8	0	0	54	3	22.70886	136.2532
9	9	0	0	55	3	12.65899	75.95393
10	10	0	0	56	5	20.83697	125.0218
11	11	0	0	57	4	18.68154	112.0893
12	12	0	0	58	6	23.84147	143.0488

13	13	0	0	59	6	16.03122	96.18732
14	14	0	0	60	4	17.39244	104.3547
15	15	0	0	61	48	29	174
16	16	0	0	62	4	3.5	21
17	17	0	0	63	4	10.30776	61.84658
18	18	0	0	64	4	19.36315	116.1789
19	19	0	0	65	3	15.23975	91.4385
20	20	0	0	66	3	18.40203	110.4122
21	22	18.02776	108.1665	67	3	22.64467	135.868
22	22	0	0	68	1	12.07107	72.42641
23	13	5	30	69	1	5	30
24	13	23.85372	143.1223	70	2	8.602325	51.61395
25	12	17.88854	107.3313	71	1	11.40312	68.41875
26	11	9	54	72	2	16.06226	96.37355
27	11	16.43303	98.59821	73	1	10.29611	61.77667
28	29	9.486833	56.921	74	1	6.264982	37.58989
29	29	0	0	75	1	9.300538	55.80323
30	7	5.830952	34.98571	76	19	14.32099	85.92596
31	15	29.68164	178.0899	77	19	9.848858	59.09315
32	7	11.40175	68.41053	78	1	6.403124	38.41875
33	8	8.276473	49.65884	79	19	4.472136	26.83282
34	9	5.024938	30.14963	80	19	8.944272	53.66563
35	9	4.242641	25.45584	81	18	6.708204	40.24922
36	16	6.082763	36.49658	82	18	10.79349	64.76095
37	16	11.18178	67.09069	83	18	5.385165	32.31099
38	39	3	18	84	88	7.031129	42.18677
39	39	0	0	85	20	4.472136	26.83282
40	39	17.67767	106.066	86	20	3.605551	21.63331
41	17	8.5	51	87	88	4.031129	24.18677
42	17	9.848858	59.09315	88	88	0	0
43	2	8	48	89	88	4.031129	24.18677
44	3	11.6297	69.77822	90	88	7.566663	45.39998
45	9	10.95084	65.70507	91	88	3.041381	18.24829
46	8	9.300538	55.80323	92	88	23.06637	138.3982

下图给出了新添平台的分布，其中已经存在的警局用圆圈标记，新添的警局用三角标记。

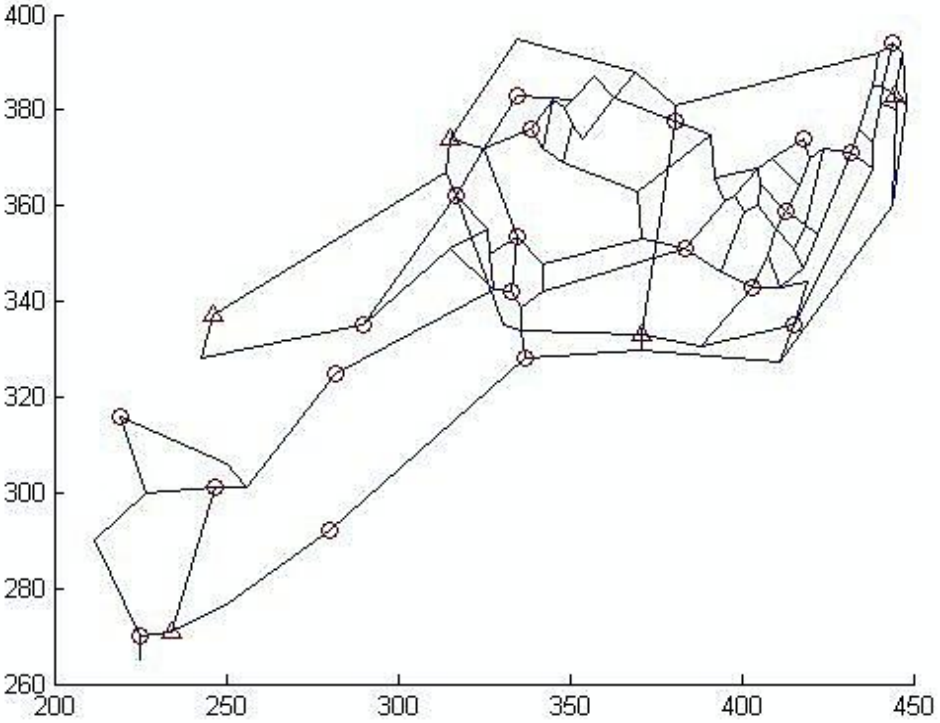

图 2　新增警局布局图

4.3.4问题一（3）的结论分析

增加了五个交巡警平台之后，满足了所有路口三分钟内出警的要求，同时调整了一系列平台工作量不均衡的问题：

其中新增的 29 号平台解决了 28,29 号路口出警时间过长的问题；88 号平台既解决了 92 号路口出警时间过长的问题，又分担了 20 号平台过大的工作量；48 号平台既解决了 61 号路口出警时间过长的问题，又分担了 5,6,7 号平台过大的工作量；39 号平台既解决了 38,39,40 号路口出警时间过长的问题;22 号平台既解决了 21,22 号路口出警时间过长的问题，又分担了 13 号平台过大的工作量。

4.4 问题二（1）的建模

4.4.1问题二（1）的分析

对于分配全市范围内的平台管辖范围，我们继续运用在 4.3 中建立的评价函数算法进行管辖区域的分配。

对于这样的一种分配方式，显然存在着一定的问题，我们需要对每一个路口进行评测，来衡量在每一个路口处的综合安全程度。然后我们对所有路口的综合安全程度进行比较，指出警局设立的不合理之处，调整现有的平台的设置结构，优化该市的平台分布的构局。

其次，我们将对每一个区的整体警局构造进行评价，得出该市各区的整体评价和改进意见。

4.4.2问题二（1）的建模及结论

4.4.2.1　问题二（1）的第一个模型

首先我们要建立每一点的综合安全程度评价函数G_i。我们有：

$$Gi = Ui * dis(i) * \sqrt{\rho_i} \quad ,$$

其中G_i表示第i号路口的综合安全程度评价函数值，U_i表示路口的案发率，dis(i)表示路口i离它对应的平台的距离，ρ_i 代表路口i所在区的人口密度。

评价函数Gi的解释：某个路口综合安全程度评价函数正相关于报案率，出警速度和人口密度的平方根。我们的目标是$\min G_i$，也就是说，G_i的函数值越低，越是说明该点的综合治安程度越好。引入人口密度是因为人口密度的增加会恶化一个地方的综合治安程度，但是由于该市的人口密度分布相对不均，直接在函数中引入人口密度不妥，由于高密度的人口并不是成比例需要交巡警平台，所以做开平方处理。

首先我们得到的是原始状态下每个路口的综合安全评价函数值，综合治安函数值前20的显示在下表中。全市的评价函数值如下图，高度为该点的函数值:

表格 7　路口评价函数值表（改进前）

路口	原始函数值	路口	原始函数值
29	131.7975019	92	58.51932954
387	131.174987	370	57.91528247
28	102.0163089	390	57.53424882
39	85.68076851	418	53.98412358
31	78.4281953	40	53.74650094
21	75.48678391	301	50.09826047
239	68.95862274	371	48.20654569
420	68.67683276	25	47.26713254
38	67.49545439	24	43.33243589
419	60.8092426	58	43.31017963

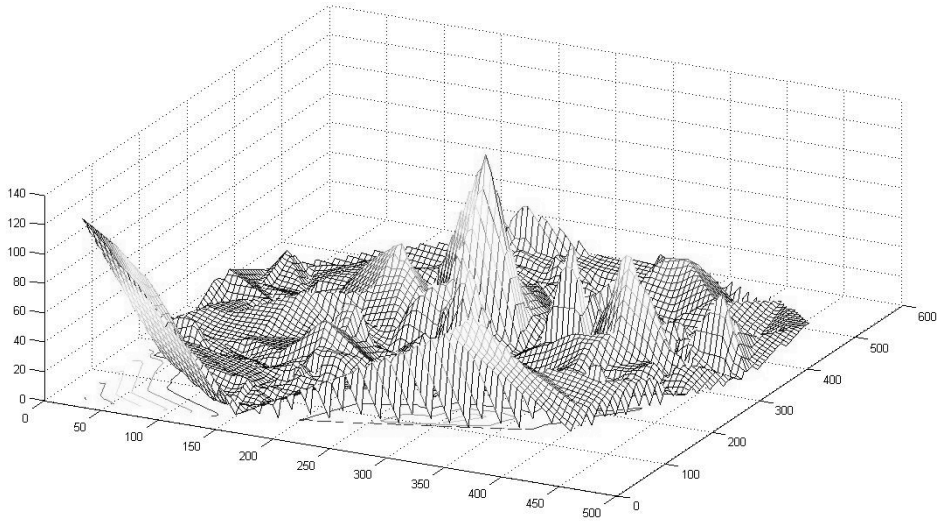

图 3　全市路口评价函数得分图（改进前）

为了改进某些评价函数得分过高的路口，我们需要对某些特别不合理的平台设计进行改进。由于拆掉原始的平台显得不太现实，我们需要做的是增加几个平台来降低某些峰值特别高的点，使得某些区域内特别不合理的平台构局得到改善。

首先我们的目标是消除G_i>100 的点，为此我们发现只需要在 29 号路口和 387 号路口增加平台即可。以下我们根据 4.3 中的算法，对增加这两个平台后的情况重新进行区域分配，所得的结果见附录。

下面是改进后的排名前 20 的评价函数值和全市的评价函数图：

表格 8　路口评价函数值表（改进后）

路口	原始函数值	改进后的函数值	路口	原始函数值	改进后的函数值
390	57.53424882	90.01701897	418	53.98412358	53.98412358
39	85.68076851	85.68076851	40	53.74650094	53.74650094
31	78.4281953	78.4281953	301	50.09826047	50.09826047
21	75.48678391	75.48678391	371	48.20654569	48.20654569
239	68.95862274	68.95862274	25	47.26713254	47.26713254
420	68.67683276	68.67683276	24	43.33243589	43.33243589
38	67.49545439	67.49545439	58	43.31017963	43.31017963
419	60.8092426	60.8092426	61	41.51934484	41.51934484
92	58.51932954	58.51932954	315	41.498383	41.498383
370	57.91528247	57.91528247	240	39.82121553	39.82121553

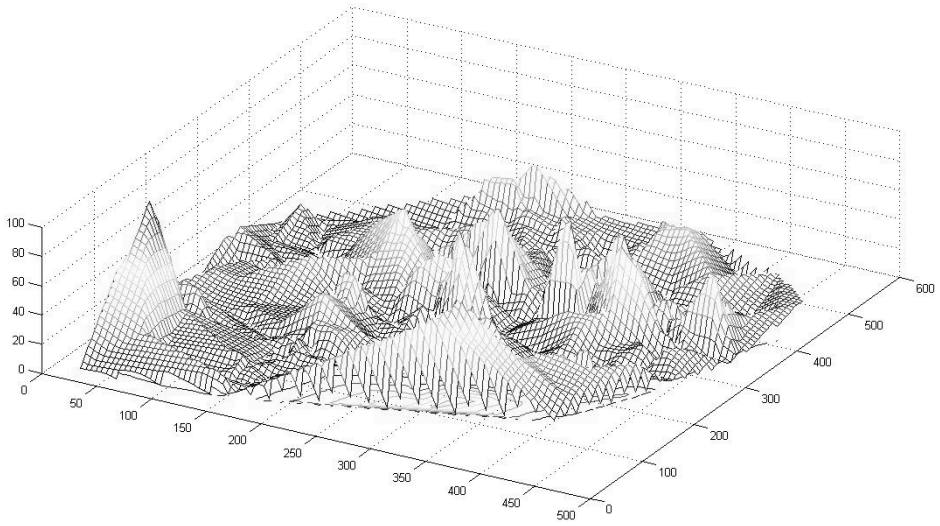

图 4　全市路口评价函数得分图（改进后）

如果需要进一步的增加平台，则可以考虑增加如下平台：

5 个：390, 39, 31, 31, 239

10 个：390, 39, 31, 31, 239, 420, 92, 370, 301, 25

4.4.2.2　问题二（2）的第二个模型

我们将另外建立一个模型，来衡量每个区之间的警局分布合理程度。

首先我们根据 4.3 的算法，计算了每个区的平台分配（见附录）和 F 值，下面建立一个新的评价函数 H：

$$H = \frac{F * population}{\sqrt{area * avecapacity}}$$

其中 F 是该区的 F 评价函数值，population 是该区的人口，area 是该区的面积，avecapacity 是该区的每个警局的平均接警量。

模型的解释：由于在 F 中存在$\sqrt{\sigma}$这个变量，除以$\sqrt{\text{avecapacity}}$，刻画的是一个区内警局工作量的变异系数；而除以$\sqrt{\text{area}}$，是因为 F 中存在距离和这一变量，为了量纲的统一，我们选择了如此处理 H。显然，G 的值越小，该地区的警局构建格局越是合理。

下面我们得到的是每个区的 H 值：

表格 9　各区 H 值表

区域	F 值	H 值
A	114.7863	588.5186
B	75.2205	54.02538
C	1.95E+03	1937.69
D	1.00E+03	1361.207
E	1.77E+03	2296.695
F	1.54E+03	1564.971

从上表看出：警局分布合理程度依次为：E,C,F,D,A,B。对于警局分布不太合理的 E,C 区，可以通过增加警局或者改变原始警局分布来进行平衡。

最后，我们给出每个区的警局接警量的变异系数：

表格 10　各区变异系数表

区号	变异系数
A	0.296525
B	0.183626
C	0.266949
D	0.321543
E	0.303029
F	0.277777

可以看出，除了 B 区以外，其他五个区的接警量的变异系数都显著偏大。这可以通过设置每个区偏好在区的出入口设置大量的平台，这会导致资源的浪费。究其原因，还是因为行政设置导致不能跨区管辖，这样导致在区与区的交界处设置冗余的平台。

解决方案：设置跨区联动接警机制，使得区与区的交界处能够得到最有效地管理，这样可以减少在边界出冗余的平台设置。

4.5 问题二（2）的建模

4.5.1 问题二（2）的分析

首先我们假设逃犯的速度和警察一样，为 60km/h。为了简化问题，我们假设警察只在路口节点设岗拦截逃犯，不在路上飞车拦截。事实上如果在路上拦截逃犯，即使迎面相逢，逃犯可以选择在交汇前掉头，警察无法有效的追击上。其次我们认为警察只要在逃犯逃出一片区域前，控制住这片区域的所有出入口，就算在这片区域内成功围堵逃犯。

4.5.2问题二（2）的建模

我们的目标是尽可能在最短的时间成功围堵逃犯，同时要求逃犯在被围堵前能逃亡的区域尽可能的小。相比较之下，后者更为重要：因为如果只追求围堵时间最短，只要将出入该城市的所有出入口都封堵就可以了，但显然这样允许逃犯自由窜逃的范围过大。于是我们的目标函数是 max dis(P32,Pi)，(Pi 为警察封堵的路口节点)，目的是：

min{max dis(P32,pi)}；　　(Pi 为警察封堵的路口节点)

min{2^{nd} max dis(p32,pi)}，约束条件 min{max dis(p32,pi)}已经达到；

min{3^{rd} max dis(p32,pi)}，约束条件 min{max dis(p32,pi)}，min{2^{nd}max dis(p32,pi)}已经达到；

……；

min{n^{st} max dis(p32,pi)}，约束条件 min{max dis(p32,pi)}，…，min{$(n-1)^{st}$ max dis(p32,pi)}已经达到。

同时满足：警察能成功围堵逃犯。

其中 n^{st} max dis(p32,pi)指离路口 32 第 n 远的封堵节点。

4.5.3问题二（2）的求解

以下是三分钟时逃犯所能逃到的路口（加黑圈标记）

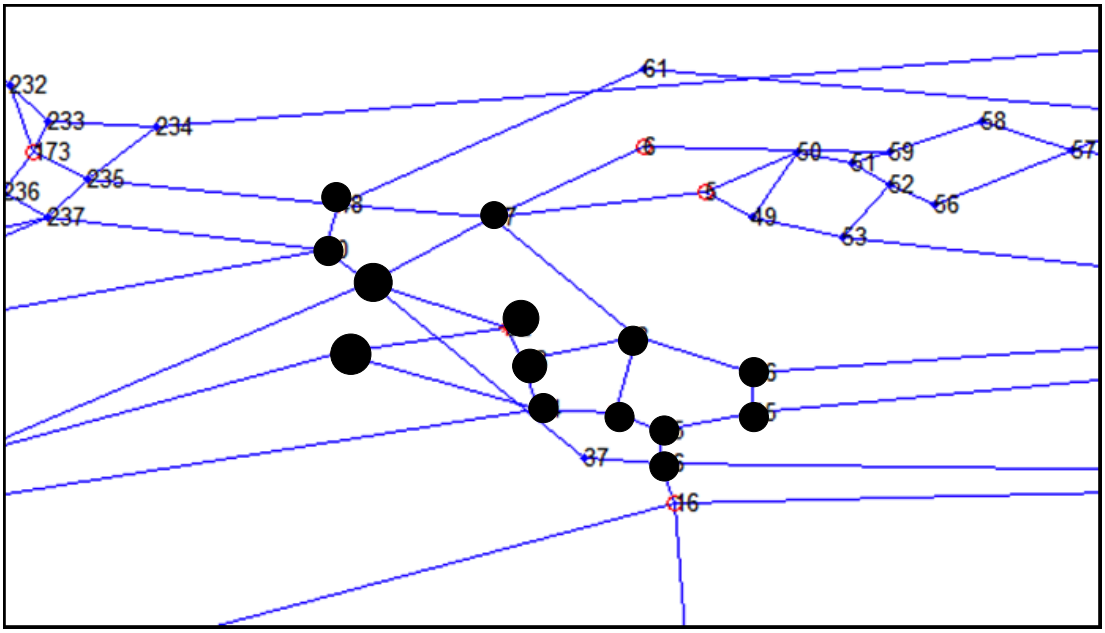

图 5　逃犯在三分钟可能逃窜的区域图

我们根据如下 3 个图（甲图，乙图，丙图）进行分析与求解，方法是为了达成最优目标，确定一些平台必须封锁的路口，从而限定了一些平台的封堵方式，然后分析得出了最优解，最后我们证明了该封堵方法的最优性：

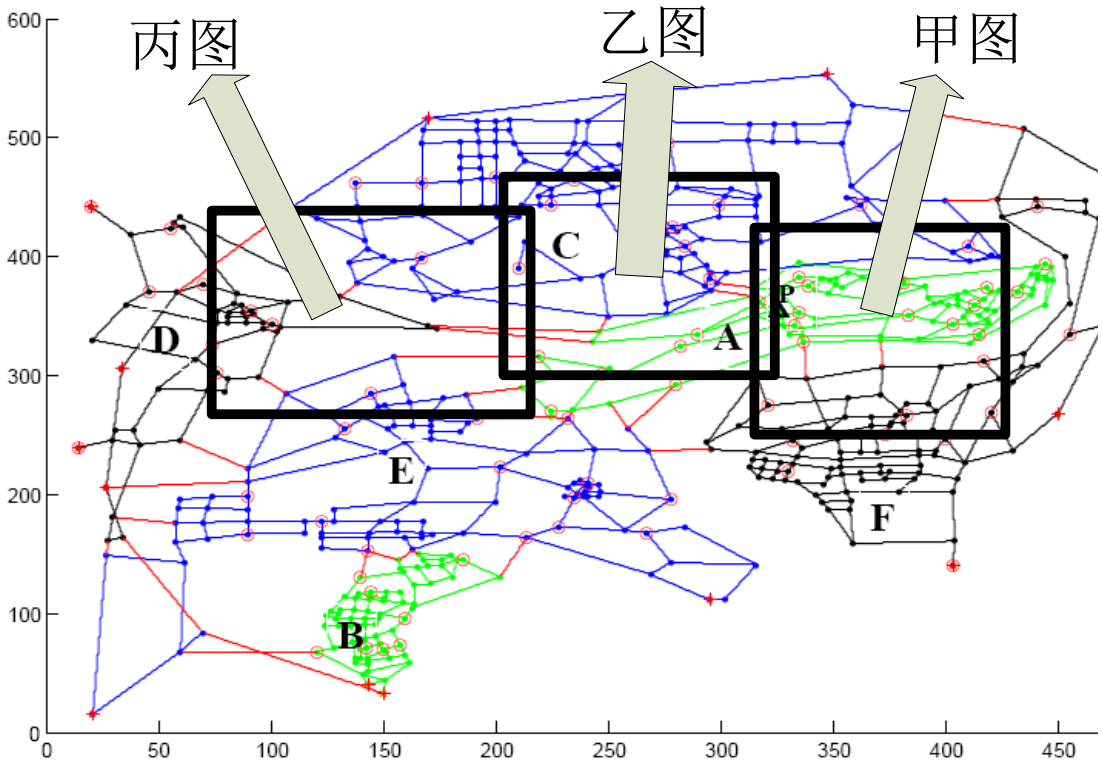

图 6　全市地图

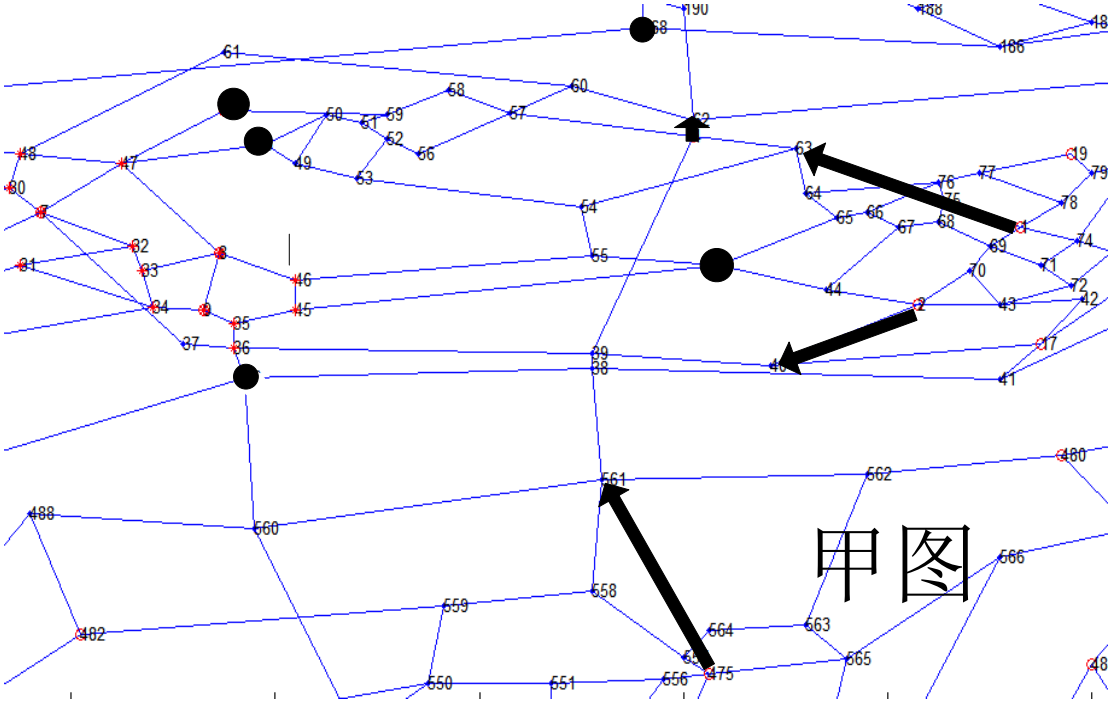

图 7　局部地图（甲图）

我们先来确定逃犯右侧周围的警局布防。首先必须要最快控制出 A 区的出入口，本着目标函数最优的原则，我们选择了就近不放的方式，选择 475→561（逃犯比所有的警察聚能先到 38 或 39，故只能选择 475→561），16→16，，5→5, 6→6, 4→62, 1→63, 3→3, 2→40, 17→41, 168→168. 此时警察出动的最长距离是

dis(475,561)=43.54

下面讨论左侧的围堵方案：

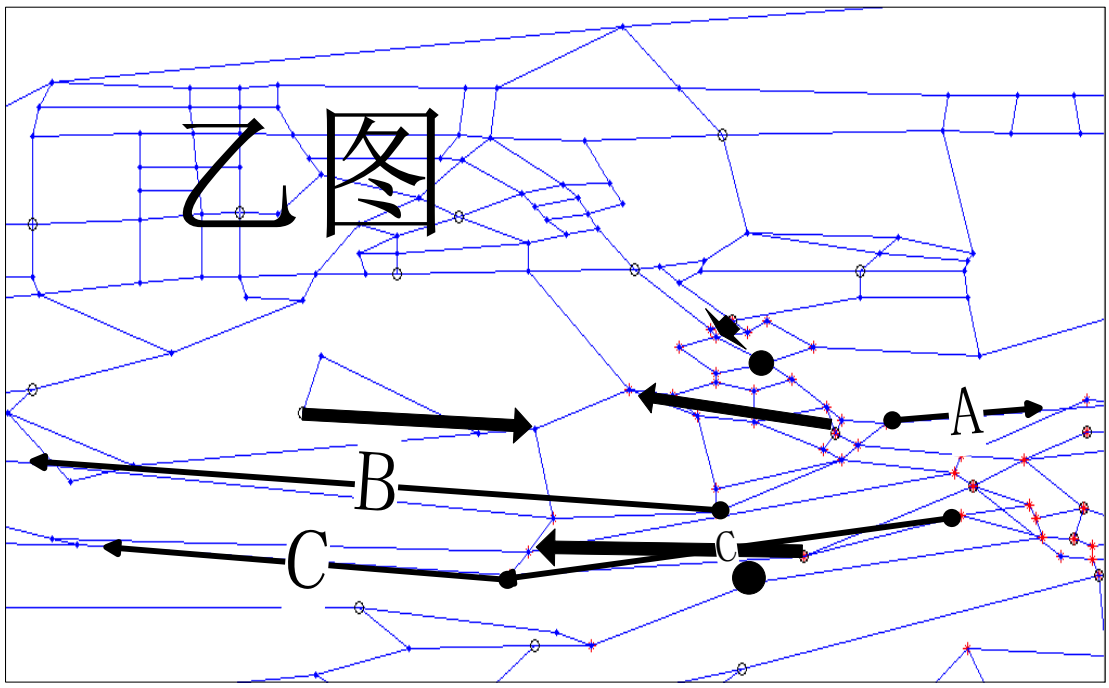

图 8 局部地图（乙图）

在左侧的布防中，我们选择 173→241, 169→240, 15→29, 172→228, 171→171, 10→10，此时左侧的最长出警距离是 dis(169,240)=70.47

这样我们在左侧给逃犯留了 3 条路线：

A：向右侧的窜逃路线，已经由左侧的 168 路口的警察封堵

B，C：下方的向左测的窜逃路线，将由另外新增的 2 个警察封堵，见丙图。

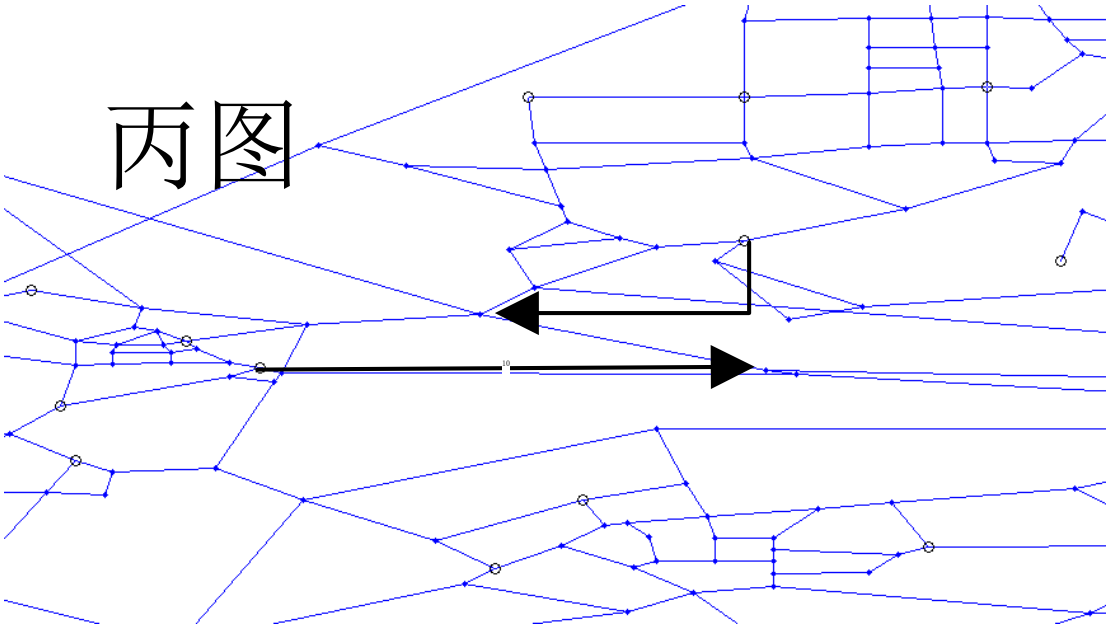

图 9 局部地图（丙图）

这两条路径分别为 320→370, 167→248，其中最长的警察出警路径是

dis(320,370)=78.08

下面我们来证明这样是最优的：

上述的方案使得 min{max dis(p32,pi)}=dis(p32,p248)= 205.2373， min{2nd max dis(p32,pi)}= 158.9405，….

假设存在方案使得 min{max dis(p32,pi)}< 205.2373

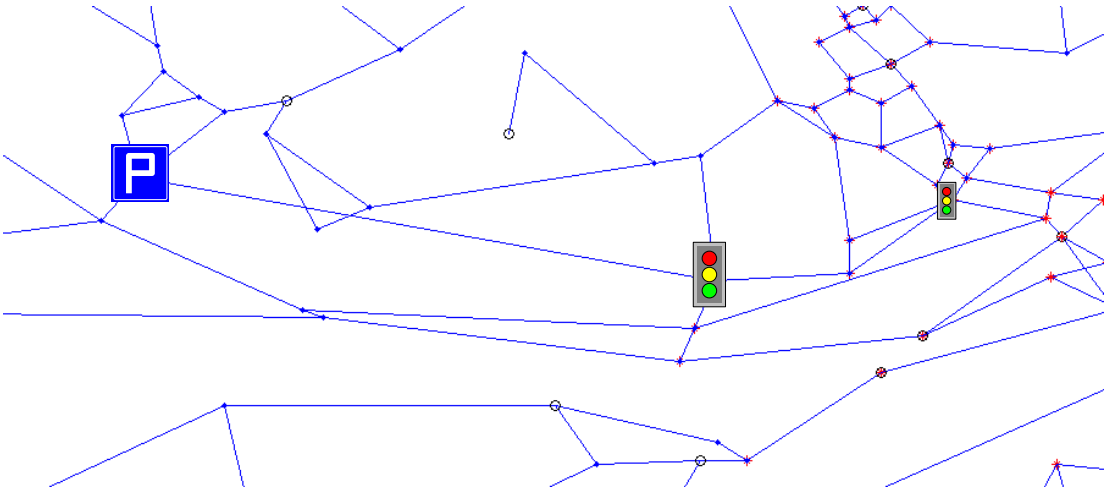

图 A

注意到上图中警察无法比逃犯 更早到达两处红绿灯位置（路口标号 237,239），于是警察必须在停车标志处拦截罪犯，不然会让罪犯逃的更远，此处即为 dis(p32,p248)= 205.2373。所以假设错误，min{max dis(p32,pi)}=dis(p32,p248)= 205.2373 是最优的；

再假设存在方案使得 min{2nd max dis(p32,pi)}<158.9405

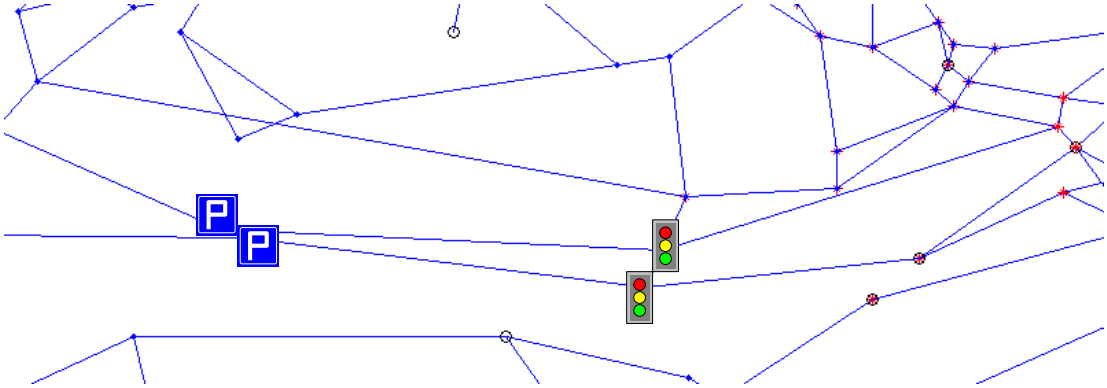

图 B

此时上图中警察无法同时封堵住两处红绿灯位置（路口标号 28,29），于是警察必须在停车标志处中的至少一处拦截罪犯，不然会让罪犯逃的更远，此二处标号即为：370,371，于是 min{2ndmax dis(p32,pi)}≥min{dis(32,370),dis(32,371)}= dis(32,371)= 158.9405。所以假设错误，min{2ndmax dis(p32,pi)}= 158.9405 是最优的。

…….

如此进行 18 次论证，证明了上述的围堵方案是使得目标函数最优的。

4.5.4问题二（2）的结论

这样，所有警察的封锁路径是：75→561,16→16，,5→5,6→6,4→62,1→63,3→3,2→40,17→41,168→168，173→241,169→240,15→29,172→228,171→171，10→10，320→370,167→248。

所有的警察到封锁口的最长路径是 320→370，满足 dis(320,370)=78.0848,即最长距离 7808.48 米，需要最长耗时 468.5088s

总体封锁区域在下图中显示

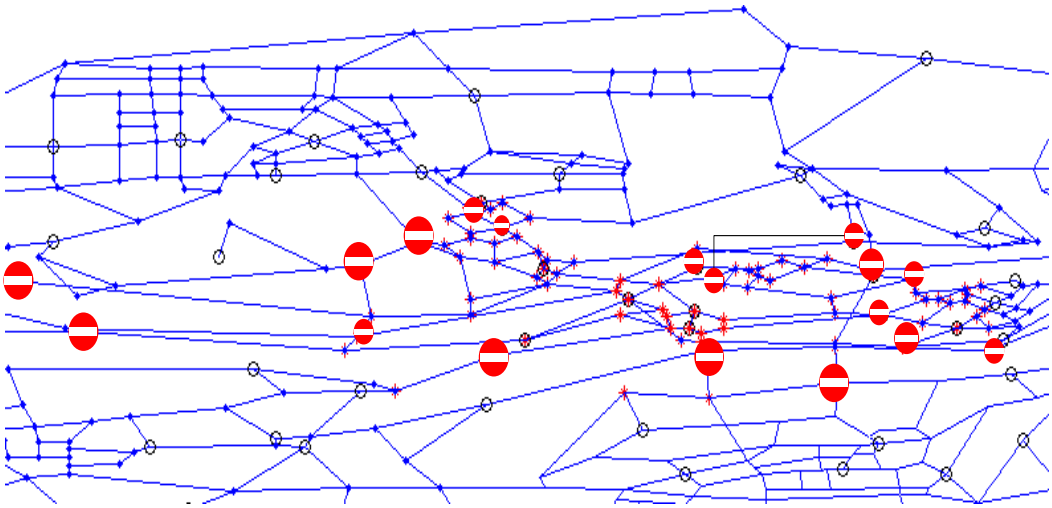

图 10　围堵方案图

5　模型分析

问题一（1）中我们采用了最短路的 Floyd 算法，得出了两点间最短距离，然后用路口找最近平台的办法得出了分管警局的分配方式。这样最大限度地保证了出警速度的最快，满足了问题中的要求。

问题一（2）中我们提出了既保证全局最优，又考虑到全局最优的目标函数，在求解中采用了分枝限界的搜索算法，把搜索量降低到了软件可以接受的范围内，得到了最优解。

问题一（3）中我们提出了评价函数 F，既考虑到减低出警时间过长的问题，又考虑到平衡各个平台的工作量，用来确定最优的平台分配方案。然后我们对 A 区未设置平台的路口进行了逐个添加评测，得出了最优的添加平台方案。在求解的过程中我们使用了贪心算法，为了使得算法得到一个较优解，我们又设置了另外一个参数进行优化。总体来说这个模型和求解过程充分考虑到了各个平台的工作职能和实际工作量情况，给出了较优的结论。实际上，由于缺乏居民的反馈信息，从而缺乏对于平台分配的精确评价标准，使得很难获得一个最佳分配方案。

问题二（1）中我们提出了两个模型：在第一个模型中我们对于每一个路口都进行了综合安全评价，提出了改进方案。该模型的优点是充分考虑了平台的工作任务是维护社会治安，提出了增加 2 个，7 个或者 12 个平台的方案，不足之处在于忽视了建造平台的成本和居民的感受。在第二种模型中我们设计了评价各区相对优劣的指标，得到了那些区需要着重需要改进，同时指出了区域边界处警局设置的不合理性，给出了改进意见。该模型优点是给出了改建的方向性，给了相对优劣指标。但是模型参数的设置存在人为的成分，主观性较大，应该综合结合第一个模型分析考

虑。

问题二（2）中我们给出了类似问题一（2）的目标函数，保证在尽可能短的时间内把逃犯围堵在最小的区域内。我们采用道路分析的方法，排除了大量的可能，得出并证明了模型的最优解。这种算法保证了解的最优性，对于逃犯在指定点逃窜的问题具有不可比拟的优势。但是不足之处在于缺乏普适性，对于逃犯在任意点任意时间的逃窜问题不能通用。

6 参考文献

[1] Thomas H. Cormen 等著，《算法导论》，北京：高等教育出版社，2002 年。

[2]Stanley B. Lippman 等著，《C++ Primer 中文版》，北京：人民邮电出版社，2006 年。

[3]隋思涟 王岩 编著，《MATLAB 语言与工程数据分析》，北京：清华大学出版社，2009 年。

[4]乐经良 编著，《数学实验》，北京：高等教育出版社，1999 年。

[5]维基百科，变异系数，http://zh.wikipedia.org/wiki/变异系数 ，2011 年 9 月 10 日

7 附录

附录一：问题一（1）的 matlab 代码

```
shortestdist=distance;
n=92;
for k=1:n
    for i=1:n
        for j=1:n
            if((shortestdist(i,k)+shortestdist(j,k))<shortestdist(i,j))
                shortestdist(i,j)=shortestdist(j,k)+shortestdist(i,k);
                shortestdist(j,i)=shortestdist(i,j);
            end
        end
    end
end
```

附录二：问题一（2）的 C++代码：

```
#include <iostream>
#include <fstream>
using namespace std;

#define MAP_SIZE 92
#define POLICE 20
#define BLOCK 13

double map[MAP_SIZE][MAP_SIZE];
int police[POLICE];
int assignlist[POLICE];
int block[BLOCK];
bool policeassigned[POLICE];
double solutiondist = 999999999;
int solutionlist[13];
int resultlist[13];

void init(void)
{
   int i,j;
   ifstream fin("data.dat");
   for (i=0;i<MAP_SIZE;i++)
   {
      for (j=0;j<MAP_SIZE;j++)
      {
         fin >> map[i][j];
      }
```

```
    }
    for (i=0;i<POLICE;i++)
    {
        fin >> police[i];
        policeassigned[i] = false;
    }
    for (i=0;i<BLOCK;i++)
    {
        fin >> block[i];
        assignlist[i] = -1;
        solutionlist[i] = -1;
        resultlist[i] = -1;
    }
}

inline double min(double x, double y)
{
    return (x<y)?x:y;
}

inline double max(double x, double y)
{
    return (x>y)?x:y;
}

inline void swap(int &a, int &b)
{
    int temp = a;
    a = b;
    b = temp;
}

void findblock1(int size, int toblock, int assignlist[13], double
inpositiondist)
{
    int i;
    double temp;

    if (toblock == 0)
    {
        if (inpositiondist < solutiondist)
        {
            for (i = 0; i < size; i++)
```

```
                solutionlist[i] = assignlist[i];
            solutiondist = inpositiondist;
        }
        return;
    }

    for (i = 0; i< size+1; i++)
    {
        if (policeassigned[i] == true)
            continue;
        temp = max(inpositiondist,
map[police[i]-1][block[size-toblock]-1]);
        if (temp > solutiondist)
            continue;
        policeassigned[i] = true;
        assignlist[size-toblock] = i;
        findblock1(size, toblock-1, assignlist, temp);
        assignlist[size-toblock] = -1;
        policeassigned[i] = false;
    }
}

void postprocess(void )
{
    int i, mini =0;
    double min =99999999;
    for (i=12;i<20;i++)
    {
        if ((!policeassigned[i])&&(map[block[11]-1][police[i]-1] < min))
        {
            min = map[block[11]-1][police[i]-1];
            mini = i;
        }
    }
    policeassigned[mini] = true;
    solutionlist[11] = mini;
    swap(police[mini], police[12]);
    if (min > solutiondist)
    {
        solutiondist = min;
    }
    min = 999999999;
    mini = 0;
```

```
    for (i = 12; i < 20; i++)
    {
        if ((!policeassigned[i])&&(map[block[12]-1][police[i]-1] < min))
        {
            min = map[block[12]-1][police[i]-1];
            mini = i;
        }
    }
    policeassigned[mini] = true;
    solutionlist[12] = mini;
    swap(police[mini], police[13]);
    if (min > solutiondist)
    {
        solutiondist = min;
    }
}

void calc()
{
    int i, j, maxj;
    double max;
    for (i = 11; i > 0; i-- )
    {
        findblock1(i, i, assignlist, 0);
        max = 0.00000000;
        maxj = 0;
        for (j = 0; j < i; j++)
        {
            assignlist[j] = -1;
            policeassigned[i] = false;
            if (max < map[block[j]-1][police[solutionlist[j]]-1])
            {
                max = map[block[j]-1][police[solutionlist[j]]-1];
                maxj = j;
            }
        }
        swap(police[solutionlist[maxj]],police[i]);
        swap(block[maxj], block[i-1]);
    }
}

void output()
{
```

```
    ofstream fout("result.dat");
    int i;

    fout << "Time to get in position:\t" << solutiondist << endl;
    fout << "Block#,\tPolice#,\tDistance,\tTime" << endl;
    for (i = 0; i < 13; i++)
    {
        fout << block[i] << ",\t" << police[i+1] << ",\t" <<
map[block[i]-1][police[i+1]-1] << ",\t" <<
map[block[i]-1][police[i+1]-1]/60 <<endl;
    }
    fout.close();
}

int main()
{
    init();
    //findblock1(11, 11, assignlist, 0);
    calc();
    postprocess();
    output();
}
```