

北 京 林 业 大 学

2019 学年—2020 学年第 2 学期计算机算法设计与实践实验报告书

专业： 计算机科学与技术(创新实验班) 班级： 计创 18

姓 名： 连月菡 学 号： 181002222

实验地点： 家 任课教师： 王春玲

实验题目： 实验 5 电路布线问题

实验环境： Visual Studio 2019 Community

一、 实验目的

- (1) 进一步掌握分支限界法的设计思想，掌握限界函数的设计技巧；
- (2) 考察分支限界法求解问题的有效程度，并与回溯法进行对比；
- (3) 理解这样一个观点：好的限界函数不仅计算简单，还要保证最优解在搜索空间中，更重要的是能在搜索的早期对超出目标函数界的结点进行丢弃，减少搜索空间，从而尽快找到问题的最优解。

二、 实验内容

印刷电路板将布线区域划分成 $n \times n$ 个方格。精确的电路布线问题要求确定连接方格 a 到方格 b 的最短布线方案。在布线时，电路只能沿着直线或直角布线，也就是不允许线路交叉。

三、 实验步骤

- (1) 对电路布线问题建立合理的模型，通过实验确定一个合理的限界函数；

图 1 所示是一块准备布线的电路板。在布线时，电路只能沿直线或直角布线。为了避免线路相交，已布线的方格做了封锁标记（图中用阴影表示），其他线路不允许穿过被封锁的方格。

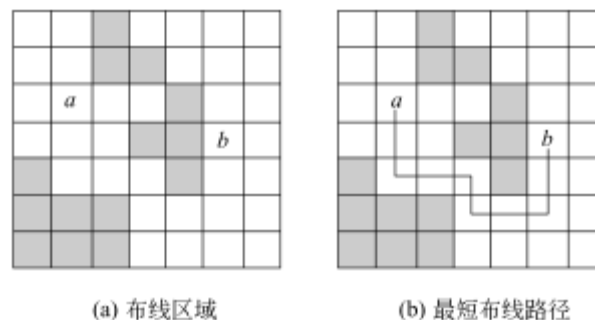
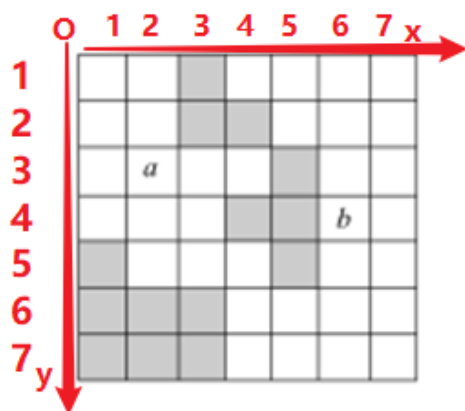
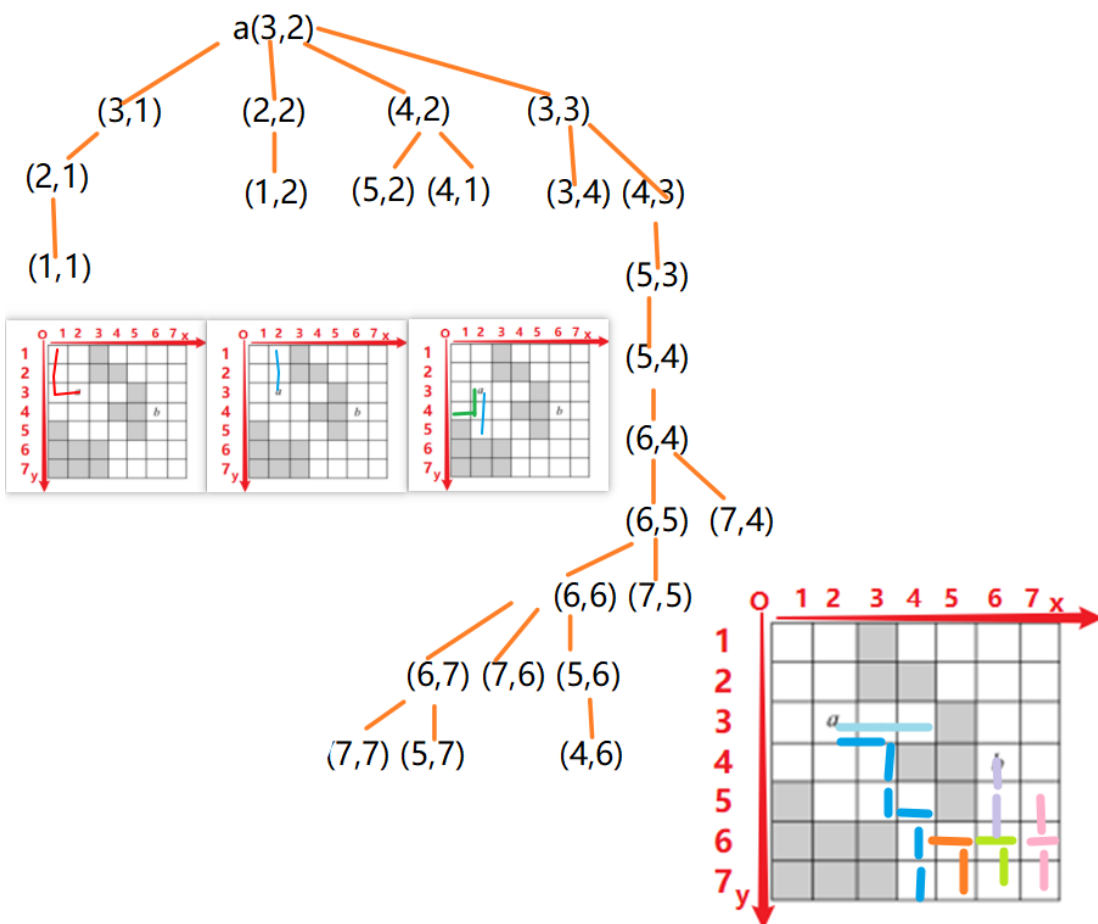


图 1 印刷电路板及其最短布线路径

通过观察,当端点没有可以移动的空间,且不能到达 b 的情况下,说明到达边界了。



例如,在该电路板中,以左上角尖端为原点,每个格子依次编上坐标。
a(3,2) b(4,6)



- (2) 设计算法实现电路布线问题;
用分支限界法求解电路布线问题,从起始方格 **a** 开始作为根结点,与起始位置 **a** 相邻且可达的方格成为可行结点,连同从起始方格到这个方格的距离 **l** 加入待处理结点表 **PT** 中(可用队列存储表 **PT**)。然后,从表 **PT** 中取出队首结点成为下一个扩展结点,将与当前扩展结点相

邻且可达的方格连同从起始方格到这个方格的距离 2 加入表 PT 中。
重复上述过程，直到达到目标方格 b 或表 PT 为空。

算法步骤

1. 如果扩展队列非空:
2. 取出队列中的首元素用以扩展,若该点是终点,则返回其步长
3. 若不是终点,则分别依次向上右下左四个方向扩展
4. 如果扩展后的点的横纵坐标没有超过方格的边框, 并且该点没有被封锁,那么标记这个点被访问过了, 加入队列, 否则搜索下一个方向。
- 5.重复 2-4 步骤,一直到队列变空或返回步长

时间复杂度为 $O(n*n)$

伪代码:

```
1. int solve() { //队列式分支限界法进行搜索
2.     while !q.empty //如果队列非空
3.         u = q.front; //将队列中的首元素取出用以扩展
4.         q.pop; //弹出队列首元素
5.         if u.x == b && u.y == bb
6.             return u.step; //已经是终点了
7.         for i=1:4 //朝着 4 个方向进行扩展
8.             v.x = u.x + dir[i][0]; //dir 数组表示是上,右,下,左
9.             v.y = u.y + dir[i][1];
10.            if 超出 n*n 方格的边框
11.                continue; //不符合
12.            if 该方格被封锁 //如果这个方格已经被封锁了
13.                continue; //不符合
14.            v.step = u.step + 1; //符合,步长+1
15.            vis[v.x][v.y] = 1; //标记这个方格的位置被访问过了
16.            q.push(v); //放到队列里,下次用以扩展
17.        return -1; //无解
18. }
```

C++代码

```

1  #include<stdio.h>
2  #include<string.h>
3  #include<algorithm>
4  #include<queue>
5  using namespace std;
6  struct Point { //结构体表示方格的三个参数
7      int x, y, step;//x, y轴坐标, 行经的步长
8  };
9  queue<Point> q; //用以存放扩展节点的队列
10 int vis[101][101]; //表示一个方格是否被走过
11 int dir[4][2] = { 0, 1, 1, 0, 0, -1, -1, 0 }; //表示行走方向
12 int n, k, a, aa, b, bb; //方格边长 封锁方格数 输入起点和终点x,y坐标
13
14 int solve() { //队列式分支限界法进行搜索
15     Point u, v; //当前用以扩展的点
16     while (!q.empty()) { //如果队列非空
17         u = q.front(); //将队列中的首元素取出用以扩展
18         q.pop(); //弹出队列首元素
19         if (u.x == b && u.y == bb)
20             return u.step; //已经是终点了
21         for (size_t i = 0; i < 4; i++) { //朝着4个方向进行扩展
22             int xx = u.x + dir[i][0]; //依次是上,右,下,左
23             int yy = u.y + dir[i][1];
24             if (xx <= 0 || xx > n || yy <= 0 || yy > n) //如果超出n*n方格的边框,则越界
25                 continue; //不符合
26             if (vis[xx][yy]) //如果这个方格已经被封锁了
27                 continue; //不符合
28             v.step = u.step + 1; //符合,步长+1
29             v.x = xx; //当前扩展到的坐标就是下一步的横纵坐标
30             v.y = yy;
31             vis[xx][yy] = 1; //标记这个方格的位置被访问过了
32             q.push(v); //放到队列里,下次用以扩展
33         }
34     }
35     return -1; //无解
36 }
37 int main() {
38     int x, y; //x y轴坐标
39     while (scanf("%d%d", &n, &k) != EOF) { //输入方格边长和封锁方格数
40         memset(vis, 0, sizeof(vis)); //访问数组重置
41         while (!q.empty()) //若队列非空
42             q.pop(); //则队列清空
43         while (k--) { //输入封锁的方格的横纵坐标
44             scanf("%d%d", &x, &y); //封锁的方格相当于访问过
45             vis[x][y] = 1; //初始化访问数组, 1表示已经访问过
46         }
47         scanf("%d%d%d%d", &a, &aa, &b, &bb); //输入起点和终点坐标
48         Point u; //创建起点
49         u.x = a, u.y = aa, u.step = 1; //初始化起点
50         q.push(u); //将起点放入队列中
51         int ans = solve(); //调用函数,寻找最少的方格数
52         printf("%d\n", ans); //输出答案
53     }
54 }

```

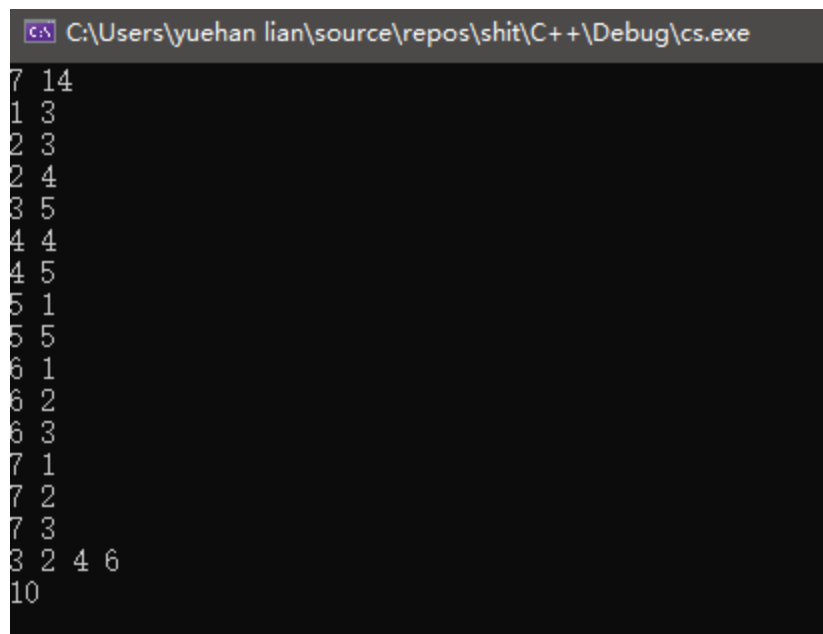
(3) 设计测试数据，统计搜索空间的结点数。

Input:

```
7 14
1 3
2 3
2 4
3 5
4 4
4 5
5 1
5 5
6 1
6 2
6 3
7 1
7 2
7 3
3 2 4 6
```

Output:

```
10
```

A screenshot of a Windows command prompt window. The title bar shows the file path: C:\Users\yuehan lian\source\repos\shit\C++\Debug\cs.exe. The window contains the same input and output as shown in the text blocks above. The input is a list of pairs and a quadruple, and the output is the number 10.

```
C:\Users\yuehan lian\source\repos\shit\C++\Debug\cs.exe
7 14
1 3
2 3
2 4
3 5
4 4
4 5
5 1
5 5
6 1
6 2
6 3
7 1
7 2
7 3
3 2 4 6
10
```

四、 实验心得

通过对搜寻可能路径的算法改进，实现能够同时输出多条可能路径的功能。而最优路径也有可能有多条，因此可以改进搜索最优路径的算法，使其能够输出全部的最优路径。可以考虑加入多重标记的法实现。