基于回溯算法解决电子线路板布线问题

摘要

本文针对矩形电子线路板的布线问题，建立了无向图的顶点染色模型。并进一步通过图的邻接矩阵转换为一系列邻接矩阵的主子阵问题。随后采用回溯算法搜索得到符合不同题目要求的解。

题目中不同的导线可视为无向图模型中不同的顶点，两个顶点间的边数是两个顶点代表的导线分布在同一层电子线路板时的交点数。这样的一个图可以刻画题目条件中导线的位置关系。从而题目中一系列导线分层问题可以转为求图的一系列满足要求的诱导子图的问题，即求图邻接矩阵的一系列满足要求的主子阵问题。需要输出的答案为图顶点集的满足题目要求的最小划分数与划分情况。

对于问题一，导线的轨迹是确定的直线段，因此每两条导线至多有一个交点，从而上述无向图是无向简单图。利用回溯算法的基本思想，我们将需要输出的划分情况表示为一些满足约束条件的数组，对数组的构成进行深度优先的搜索并在出现问题时回溯，直至搜索得到一个好的解。最终得到问题一的答案为3。

对于问题二，导线的轨迹与问题一中相同，因此图模型与图邻接矩阵也是相同的。仍然利用回溯算法的基本思想，将需要输出的划分情况表示为一些满足约束条件的数组，只需在深度优先搜索时将问题一中需要满足的条件更改为问题二中需要满足的条件即可。最终得到问题二的答案为2。

对于问题三，导线的轨迹不再是确定的，因此需要首先对导线的轨迹进行优化。找到使得相交情况全局最优的导线排布情况，从而确定此时对应的图模型与邻接矩阵。随后注意到导线的转弯数也是确定的，因此约束条件仍然可以转化为仅对交叉数的约束条件，并采用与前两问相同的算法进行搜索。最终得到问题三的答案为3。

对于问题四，上述三个问题依次的模型建立与算法仍然是使用的，只需要调整计算过程中输入的参数，最终得到问题四对应的三个问题答案分别为6、4、4。

最后，本文给出了建模过程与算法设计的优缺点分析，并分析了算法时间复杂度上的优化空间，在一些特殊情况下提出了算法的优化方式。同时注意到了一些问题与本文中所解决问题的相关性，给出了模型与算法可能的推广使用场景。

本文结论正确，建模与算法逻辑严密，均具有一定的应用价值和普适性。

一、问题重述

1.1 问题的背景

布线是电路设计的核心环节。经典的电路布线问题模型中，在一块电路板的上下两端分别有 n 个端口，用若干条导线将某些端口两两相连，并且可将这些导线置于若干层电子线路板上，使得每一层上的导线互不相交。因为不同层的电子线路板上的相同位置的端口是彼此有导线连接的，因此两个不同位置的端口只要在某一层上有导线连接，便可以认为这两个端口连接在了一起。

但在实际生产中，端口不一定只出现在电路板的上下两端，导线排布也不一定严格地要求“互不相交”，导线也不一定要求是直线，不过导线排布的方式可能还有另外某些限制。

1.2 问题的条件

本题中，某种电子零部件的矩形电子线路板在四条长度均为12cm的边上都均匀地排有端口，它们分别命名为“A1 ～ A5”，“B1 ～ B5”，“C1 ～ C5”，“D1 ～ D5”，具体分布情况如图1所示。

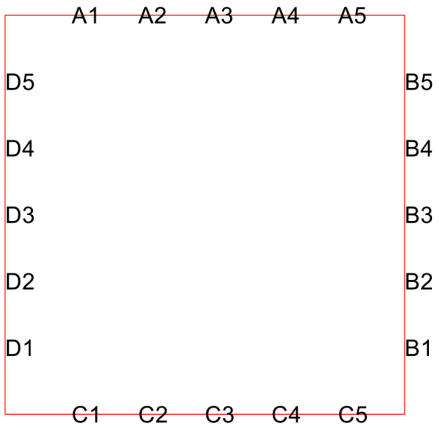

图 1

表1给出了要求用导线连接起来的端口对，其中每根导线都粘贴在电子线路板上。

表 1

起始端口	A1	B2	A2	D1	B4	A4	C2	D3	C4	C5
终止端口	B1	C1	A5	B3	A3	B5	D2	C3	D5	D4

1.3 问题的提出

（1）如果导线将端口直线相连，且完全类似于经典的布线问题，即不允许它们相交，那么至少分多少层就能完成布线？

（2）导线仍然将端口直线相连，若此时允许任意一根导线与其他所有导线相交至多一次，那么至少分多少层就能完成布线？

（3）如果导线从端口出发以及进入端口时的方向都必须与端口所在的边沿垂直，但允许导线转弯。而导线转弯时，只能转过90度的整数倍，且转弯半径至少是1厘米。若此时要求每根导线的转弯数与所在层内其它所有导线与其的交点数之和不超过2，至少分多少层就能完成布线？

（4）图2和表2 给出了一个四条边长度均为18cm的矩形电路板上均匀排布端口的分布与连接方式，在此条件下重新求解上述问题。

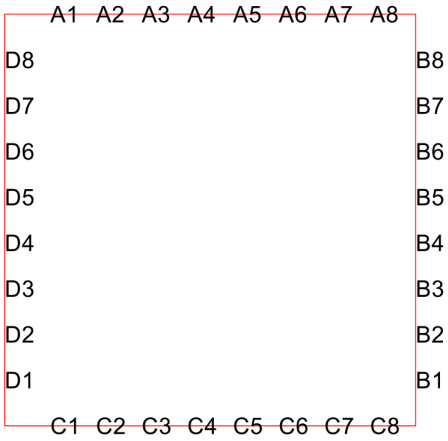

图 2

表 2

起始端口	A1	B1	B2	A2	B4	A3	B5	B6	A4	A5	B7	A6	A7	B8	D4	C8
终止端口	B3	C2	C3	D2	C1	D1	C4	C5	A8	C6	D5	D3	D6	C7	D8	D7

二、问题分析

2.1 问题一

问题一要求找到层数最少的保证每一层上直线导线不相交的布线方法。由于一共只有10根导线，显然电路板的层数不需要超过10层。为了确保满足题目要求，并且找到的层数是“最少”的，只需要依次验证1层、2层、3层、……电子线路板时符合题目要求的排布是否存在，而使得符合题目要求的排布存在的最小层数就是题目所求。

2.2 问题二

对于问题二，允许每块电路板上的每条直线导线与其他所有导线相交至多一次，在此条件下寻找符合题目要求的排布存在的最小层数。类似前一问题的思路，同样知

电路板的层数一定不超过10层，仍然依次验证1层、2层、3层、……电子线路板时符合题目要求的排布是否存在即可。

2.3 问题三

在问题三中，导线转弯的次数是容易确定的，从而可以计算出在同一层电路板上可以排布的每条导线上交点的数量限制。但导线的轨迹不再确定，因此两条导线是否交错是不确定的。

为此，我们先试图对每条导线的轨迹进行优化，使得该导线与其它所有导线的交点总数取到所有可能值中的最小，且尽可能做到全局最优。如果对每条导线都能如此优化，那么任意两条导线之间的交点数一定都是所有情况下的最小值，于是此时满足题目要求的最小层数一定不大于导线轨迹其它情形中满足题目要求的最小层数，从而它就是最终所求。

而进行了如此的优化之后，导线交错情况是确定的，同样可以通过类似前两问的做法求解。

2.4 问题四

只需采用新给出的参数条件，对前三个问题依次进行求解即可。

三、模型假设

（1）电子线路板边沿上的端口视为没有面积的点，且它们均匀分布，即每两个端口之间的距离以及首、末位置的端口距所在边沿端点的距离均相等。

（2）导线都是足够细的，即导线本身没有宽度。

（3）每层电子线路板都是完全相同的，即不考虑电子线路板的顺序差异。

（4）直导线是绝对笔直的；转弯部分的导线是光滑的圆弧。

（5）题目中所称的相交当且仅当导线在同一层上产生交点。

四、符号说明与名词解释

符号/名词	说明
交错的	若将两条导线分布在同一层时，它们会相交，则称这两条导线是“交错的”
G	无向图
V	顶点集
E	边集
A	邻接矩阵
V'	顶点集 V 的非空子集
G'	诱导子图
E'	诱导子图 G' 的边集
A'	诱导子图 G' 的邻接矩阵，即 A 的一个主子阵

五、模型建立与求解

为了方便问题的解决与叙述，首先将题目中的导线与顶点按照表3与表4进行编号：

表 3

起始端口	A1	B2	A2	D1	B4	A4	C2	D3	C4	C5
终止端口	B1	C1	A5	B3	A3	B5	D2	C3	D5	D4
编号	1	2	3	4	5	6	7	8	9	10

表 4

端口名称	A1～A5	B5～B1	C5～C1	D1～D5
编号（依次）	1～5	6～10	11～15	16～20

5.1 模型的建立

将所有的导线看作顶点集 $V = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ ，两个顶点间的边数取为它们所对应导线分布在同一层电子线路板时的交点数，建立无向图模型 $G = (V, E)$ 。

考虑分布同一层电子线路板上的所有边以同样方式生成的图 $G' = (V', E')$ ，注意到两根确定的导线排布在同一层电子线路板上时交点数保持不变，从而 G' 是 G 的诱导子图。此时，导线分布在不同层的电子线路板上可转化为对上述无向图的顶点进行染色，其中分布在相同一层电子线路板上的导线所对应的顶点染相同的颜色，分布在

不同层电子线路板上的导线所对应的顶点染不同的颜色，且该对顶点的染色符合不同题目的要求。

进一步考虑图 G 的邻接矩阵 A ，知诱导子图 G' 的邻接矩阵 A' 为 A 的一个主子阵。因此原问题就转化为求图邻接矩阵满足某些条件的主子阵的一系列问题。

5.2 问题一：导线均为直线且排线不允许相交时模型的建立与求解

导线均为直线时，两根导线是否交错是确定的，可以通过对端口编号进行判断与计算得到所有会交错的导线对，从而可以生成对应的无向简单图的邻接矩阵。

进一步，题目要求将每条导线排布在若干层电子线路板上，且同一层电子线路板上的导线两两是不相交的。即排布在同一层的导线对应的顶点构成上述无向简单图G的一个独立集，其生成的诱导子图 G' 的邻接矩阵 A'为零矩阵。需要将图的顶点划分成若干个独立集，并求划分数的最小值。换言之，就是将对应的无向简单图中的每个顶点染色，保证每个顶点与它的邻居是不同色的（由此同色顶点正好构成一个独立集），求染色数的最小值。这正是优化版本下的图着色问题（Graph Coloring Problem）。

为解决这一问题，我们首先生成问题所对应的无向简单图G的邻接矩阵 A ，随后采用回溯算法进行搜索。

5.2.1 算法准备

先将端口按照表4的顺序进行编号，然后用数对 (a_i, b_i) 表示第i条导线的起始端口与终止端口的编号，并要求 $a_i < b_i$ 。

例： $(a_1, b_1) = (1, 10)$ ，这表示编号为1的导线连接的两个端口分别是1号端口与10号端口。

邻接矩阵A中的 (i, j) 位置阵元可以按照如下方式给出：

（1） $i = j$ 时，阵元位置处于邻接矩阵的主对角线上，由于 G 是简单图，因此 $A(i, i) = 0, \forall\, i$ 。

（2） $i \neq j$ 时，注意到第i条导线和第j条导线是交错的当且仅当 $a_i < a_j < b_i < b_j$或 $a_j < a_i < b_j < b_i$ ，因此若 $a_i < a_j < b_i < b_j$ 或 $a_j < a_i < b_j < b_i$ ，则 $A(i, j) = 1$ ；否则， $A(i, j) = 0$ 。

至此，得到了问题一所对应的简单无向图的邻接矩阵 A 。

得到邻接矩阵 A 后，原问题转化为：将顶点集 $V = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ 划分为 m 个非空集合，使得以每个集合中的数为行、列生成的 A 的主子阵为零矩阵。求能够达成上述要求的最小的 m 。为此，由问题分析2.1，只需要依次检测 $m = 1, 2, \ldots$ 时是否存在可行解，并输出存在可行解时最小的 m 及其对应的划分方式。

引入表示划分的划分矩阵 P ，其每一行构成的行向量中的非零元表示顶点集 V 划分所得的一个集合。 P 的第 i 行上的非零元素就表示位于第 i 块电路板上的导线全体。

为叙述简洁，若划分矩阵 P 的任意一行中非零元素构成的行向量 p_i 均满足：以 p_i 中元素为行、列生成的邻接矩阵 A 的主子阵 A_i 中每一行和的最大值不大于 0 ，则称划分矩阵 P 满足“主子阵0判据”。

若划分矩阵 P 满足“主子阵0判据”，则由于邻接矩阵 A 中每个元素都是非负整数，知它的主子阵中每个元素都是非负整数，再由“主子阵0判据”中以 p_i 元素为行、列生成的主子阵 A_i 满足每一行行和的最大值都不大于 0 ，知 A_i 中只含有阵元 0 。即对任意的i， p_i 所表示的这些导线是两两不交的。因此划分矩阵 P 所表示的划分方式是一个满足题目一要求的划分方式。

又因为划分矩阵 P 所表示的划分方式满足题目一要求时，其以 p_i 中元素为行、列生成的邻接矩阵 A 的主子阵 A_i 均为零矩阵，满足“主子阵0判据”，因此划分矩阵 P 满足“主子阵0判据”与它表示的划分是一个满足问题一要求的解是等价的。

5.2.2 问题一的回溯算法

首先，生成一棵 $l+1$ 层树 T ， l 为导线总数，除根节点外每层分别代表一条导线的分层方式。其所有非叶子节点均恰有 m 个子节点，则其共有 m^l 个叶节点，表示全部可能的分层方案。其次，回溯法或深度优先搜索法的具体操作为：从根节点出发，优先访问其所有子节点中最左侧的一个，判断此时已生成的子树是否满足主子阵判据。若满足，则继续访问下一层的节点；若不满足，则忽略以当前节点为根节点的子树，退回到上一层的父节点并重新进行访问。最后，此搜索要么在达到叶节点时停止，要么此时所有方案均不满足条件（停在根节点）。并且，此算法可以找到所有可行解。

具体来说，首先令 P 为 m 行 10 列的零矩阵。

随后将编号为 1 的导线置于P第1行中第1个0阵元所在的位置即 $P(1,1)$ 。

假设 $n-1$ 号导线位置已经确定，$n \geq 2$ 。

再顺次决定n号导线的位置，将它依次置于 P 第 i 行 $(1 \leq i \leq m)$ 中第1个0阵元所在的位置，并检查此时的划分矩阵 P 是否满足“主子阵0判据”。若满足，再考虑 $n+1$ 号导线的位置；若不满足，表明此时已有导线相交，舍弃此情况下所有分布方案并调整n号导线的位置并重新进行判断。若n号导线不存在满足判据的分层方案，则回溯至 $n-1$ 导线并调整其位置。以此类推。

算法的流程图如图3所示：

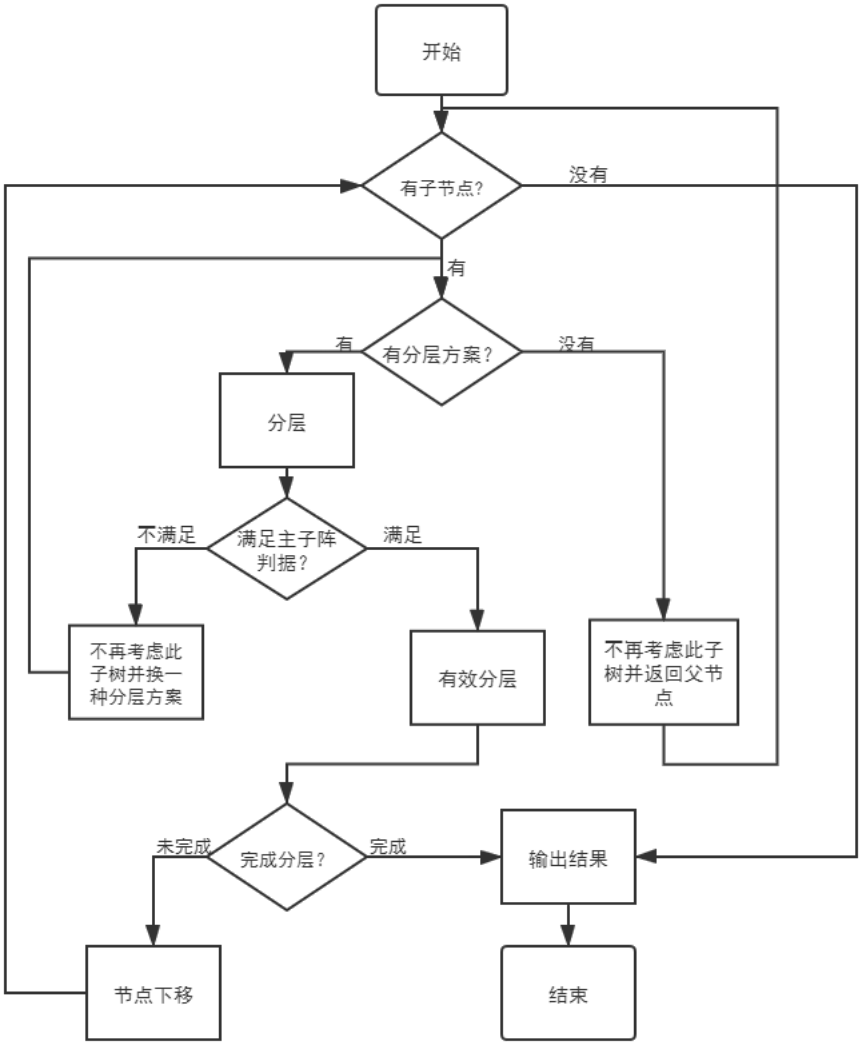

图 3

注1：为程序的效率考量，我们将与其余导线的相交数从大到小，对1～10号导线重新排序，以期尽可能减少不必要的运算（即对邻接矩阵 A 进行置换合同变换，使得它的行和递减）。输出答案时再将导线的编号还原。

注2：按照如上方法可获得 m 取最小值时的全部解，但每一个解会出现 $m_{\min}!$ 次。可在 $2,3,\ldots,m$ 放入 P 中时要求它禁止出现在顺序靠后的非零行，这样就能排除所有的重复解。如：

$$P=\begin{pmatrix}1&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\end{pmatrix}$$

当将 2 放入此矩阵时，只能放在第一或第二行，而禁止第三行（因为它是顺序靠后的全零行）。

5.2.3 问题一的答案展示

经过参数输入与程序计算，最终得到 m 最小值取 3 ，当 $m=3$ 时，共有 96 个解，我们将其储存在文件“solutions of 1.xls”中。

其中的第一个解如图4所示，其中不同颜色表示导线置于不同层的电子线路板上：

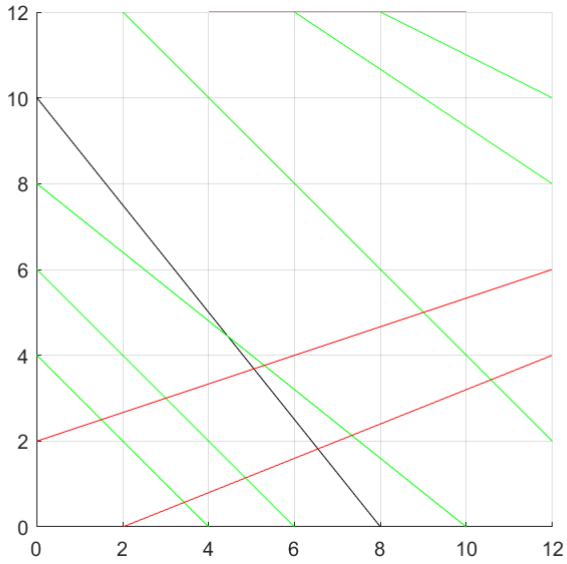

图 4

5.3 问题二：导线均为直线且至多相交一次时模型的建立与求解

5.3.1 算法准备

此时的邻接矩阵 A 及其生成方式与5.2.1中完全相同，但判断划分矩阵 P 是否符合题目条件的判据产生了变化。

同样为叙述简洁，若划分矩阵 P 的任意一行中非零元素构成的行向量 p_i 均满足：以 p_i 中元素为行、列生成的邻接矩阵 A 的主子阵 A_i 满足每一行和的最大值不大于 1 ，则称划分矩阵 P 满足“主子阵1判据”。

若划分矩阵 P 满足“主子阵1判据”，则由于邻接矩阵 A 中每个元素都是非负整数，知它的主子阵中每个元素都是非负整数，再由“主子阵1判据”中以 p_i 元素为行、列生成的主子阵 A_i 满足每一行行和的最大值都不大于 1 ，知 A_i 中每行至多只含有一个非零阵元 1 。即对任意的i， p_i 所表示的这些导线每根上至多只有一个交点。因此划分矩阵 P 所表示的划分方式是一个满足题目二要求的划分方式。

又因为划分矩阵 P 所表示的划分方式满足题目二要求时，其以 p_i 中元素为行、列生成的邻接矩阵 A 的主子阵 A_i 中每行至多只有一个非零阵元 1 ，满足“主子阵1判据”，因此划分矩阵 P 满足“主子阵1判据”与它表示的划分是一个满足问题二要求的解是等价的。

5.3.2 问题二的回溯算法

思路与5.2.2大致相同，只需将算法中用于判断的“主子阵0判据”修改为“主子阵1判据”即可。

5.3.3 问题二的答案展示

经过参数输入与程序计算，最终得到m最小值取2，当m=2时，共有6个解，我们将其储存在文件“solutions of 2.xls”中。

其中的第一个解如图5所示，其中不同颜色表示导线置于不同层的电子线路板上：

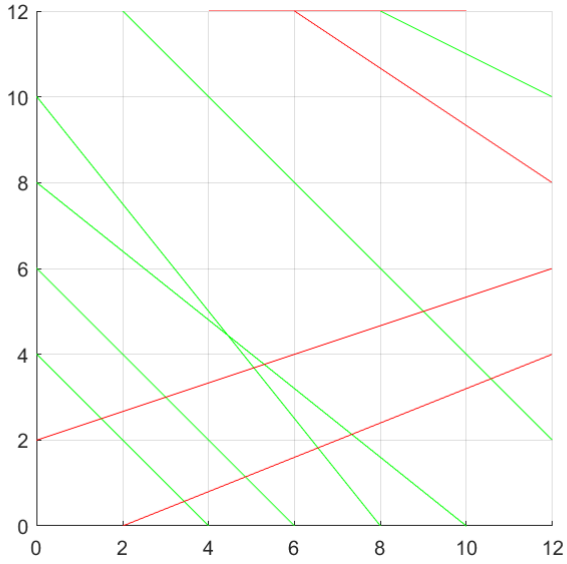

图 5

5.4 问题三：导线垂直于端口边沿且转弯半径不小于1cm时模型的建立与求解

5.4.1 算法准备

由于此时导线的交错情况比较复杂，邻接矩阵也并不确定，为此，我们先定义邻接矩阵的“最优情形”。

最短导线对于端点不在对边上的一条导线，存在唯一一条使得该导线长度最短的可能导线，称为最短导线。

最优布线方案对于一族给定的端点对，如果在平面上存在一种不分层的布线方案，使得每一条导线都有取到所有可能布线方案中数量最少的交点个数，就称此布线方案为最优布线方案。

易见，若最优布线方案存在，则只需在最优布线方案下的邻接矩阵中考虑分层问题即可，换言之我们就将问题划归为先前已经处理过的情况。在导线不太多的情况下，

可以不考虑同一边上有多条端点均在此边上的导线，这在题目涉及到的两种导线状态下成立。断言，对所有导线均存在最短导线的情形，最优布线方案必存在。

证明：由于最短导线均存在，所有导线的端点或在相邻边，或为同一边。我们首先考虑端点在相邻边上的导线l,和另一条导线t。如果l与t满足相嵌交错判据，则其必定相交错，对应邻接矩阵元素必为1.若否，分类讨论：

(1)t的端点也在一对邻边上，且均与l不同，如图6易见，l为最短导线时才有可能令其与t交点数最少；

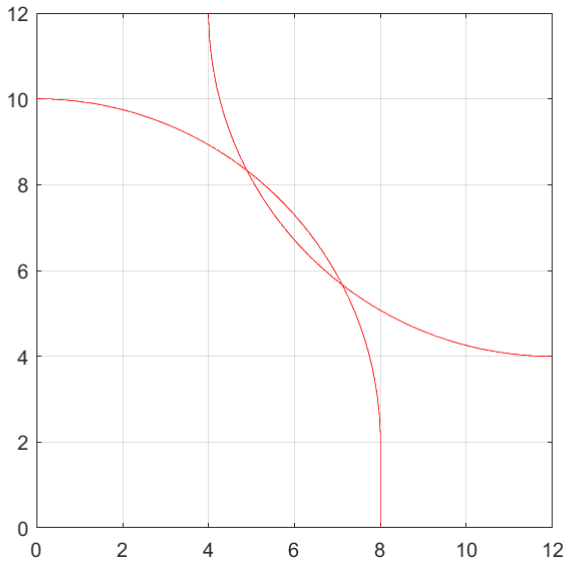

图 6

(2)t的端点也在一对邻边上，且与l有一条相同边，显然其无交点，对应邻接矩阵元素必为 0 （如图7所示的情形）；

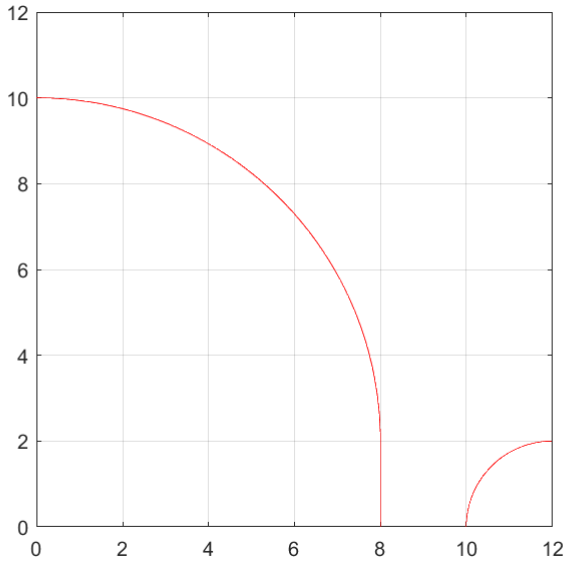

图 7

(3)t的端点也在一对邻边上，且与l完全相同，则当它们都为最短导线时，如图8可见它们不会交错，即对应邻接矩阵元素为 0 ；

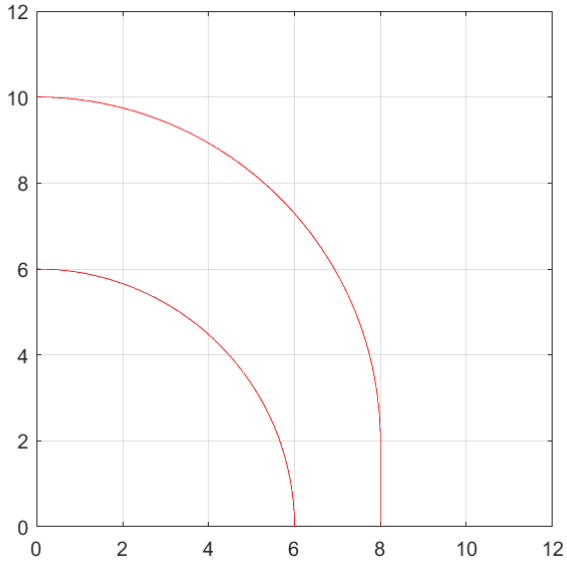

图 8

(4)t的端点在同一条边上，此时可将其视为两段相连的端点在邻边上的最短导线，由之前讨论即证（如图9所示的情形）。

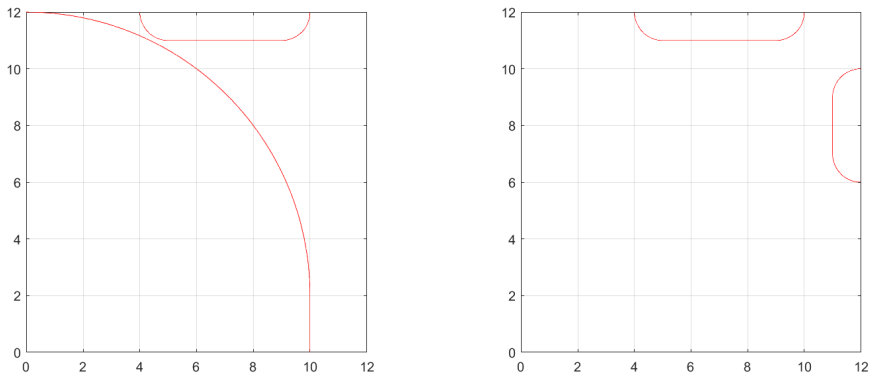

图 9

如果 l 的端点在同一边上，只需考虑 t 也在同一边上的情形，如果在对边或邻边上则与之前讨论类似，可知其均为最短导线时交点数最少。如果为同一边，则其或必定相交，或可分别取两个充分接近的转弯半径（如1厘米和1.0001厘米），使得其不相交，同时不影响与其他导线的相交情况。证毕。

于是，在本题的具体问题中，我们可以首先确定存在最优导线的全体导线的布线方案，随后调整剩余导线(即端点在对边上的导线)的排布方式，使得其对应邻接矩阵边上的全体元素都尽可能小。事实上，某些导线一定会与我们调整的目标导线相交，且交点个数确定，另一些导线则一定不会与之相交。容易验证，随着我们调整的目标导线的形状变化，与其交点数会产生变化的导线只有2号导线。也就是说，我们只需针对个别导线调整目标导线(这在本题两种情形中均容易实现，借助判断导线相交的算法，具体方案如图)。至此我们便确定了最佳的邻接矩阵，此后的操作便可仿照先前的方案进行。

接下来我们来解释一下我们判断导线相交的的算法，对任意两条已经确定具体位置的导线，我们首先将其分别参数化。随后，分别在其上以充分小步长取两组点，计算它们之间距离的最小值。近似地，我们可以认为此最小值大于0.5时就可以认为它们不交。

针对端点在对边的导线2，其恰有两个转弯处且转弯方向也唯一确定，我们只需针对导线4对其进行调整。由于其端点固定，我们可以用三个变量唯一确定其位置。它们分别是初始端口所在直线段长度t_1，第一个转弯半径大小t_2以及第二个转弯半径大小t_3。计算得，当$t_1 = 0, t_2 = 1, t_3 = 3$时（如图10所示），目标导线2与导线4无交点，此时便得到了全局最优布线方案，也就可以确定相应的邻接矩阵。

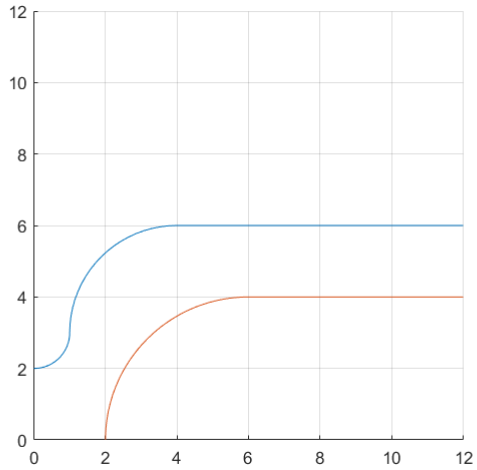

图 10

于是我们可以生成一个邻接矩阵 A ，但判断划分矩阵 P 是否符合题目条件的判据产生了变化。我们引入广义邻接矩阵 A_0 的概念， A_0 非对角线上元素与A完全相同，

同样为叙述简洁，若划分矩阵 P 的任意一行中非零元素构成的行向量 p_i 均满足：以 p_i 中元素为行、列生成的广义邻接矩阵 A_0 的主子阵 A_i 满足每一行和的最大值不大于 2 ，则称划分矩阵 P 满足“主子阵2判据”。

若划分矩阵 P 满足“主子阵2判据”，则由于广义邻接矩阵 A_0 中每个元素都是非负整数，知它的主子阵中每个元素都是非负整数，再由“主子阵2判据”中以 p_i 元素为行、列生成的主子阵 A_i 满足每一行行和的最大值都不大于 2 ，于是对任意的i，第i条导线转弯数和与替他所有导线的交点数之和不超过 2 。因此划分矩阵 P 所表示的划分方式是一个满足题目三要求的划分方式。

反过来，因为划分矩阵 P 所表示的划分方式满足题目三要求时，其以 p_i 中元素为行、列生成的邻接矩阵 A 的主子阵 A_i 中每行非0元素（转弯数与交点数）之和不大于2，满足“主子阵2判据”，因此划分矩阵 P 满足“主子阵2判据”与它表示的划分是一个满足问题二要求的解是等价的。

5.4.2 问题三的回溯算法

思路与5.2.2完全相同，在具体算法实现上，只需修改主子阵判据为“主子阵2判据”即可。

5.4.3 问题三的答案展示

经过参数输入与程序计算，最终得到m最小值取3，当m=3时，共有60个解，我们将其储存在文件“solutions of 3.xls”中。

其中的第一个解如图11所示，其中不同颜色表示导线置于不同层的电子线路板上：

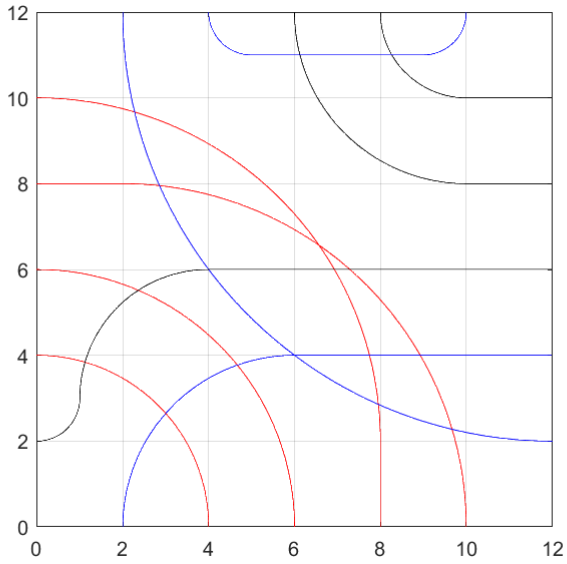

图 11

5.5 问题四：新参数下上述问题的模型建立与求解

对于问题四，可以完全参照问题一、二、三的思路进行。

具体地，对于问题四的第一、二小问，只需改变初始输入的导线对即可。

而对于第三小问，还需如同5.4.1判断邻接矩阵。针对端点在对边的导线11，其恰有两个转弯处且转弯方向也唯一确定，我们只需针对导线8与13对其进行调整。由于其端点固定，我们可以用三个变量唯一确定其位置。它们分别是初始端口所在直线段长度t_1，第一个转弯半径大小t_2以及第二个转弯半径大小t_3。计算得，当$t_1 = 10, t_2 = 2.2, t_3 = 1.8$时，目标导线11与导线8和13都无交点（如图12所示），此时便得到了全局最优布线方案，也就可以确定相应的邻接矩阵。

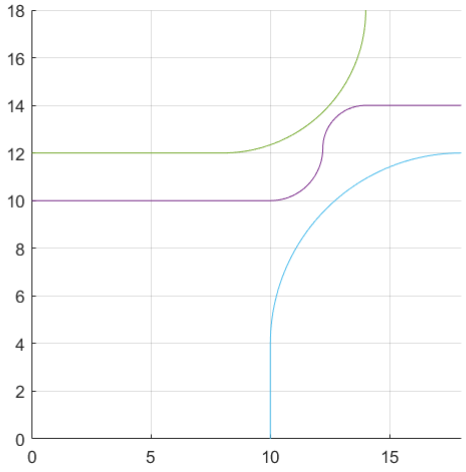

图 12

5.5.1 问题四的答案展示

对第一小问，最终得到 m 最小值取 6 ，当 $m = 6$ 时，共有 5004 个解，我们将其储存在文件“solutions of 4.1.xls”中。其中的第一个解如图13所示，其中不同颜色表示导线置于不同层的电子线路板上：

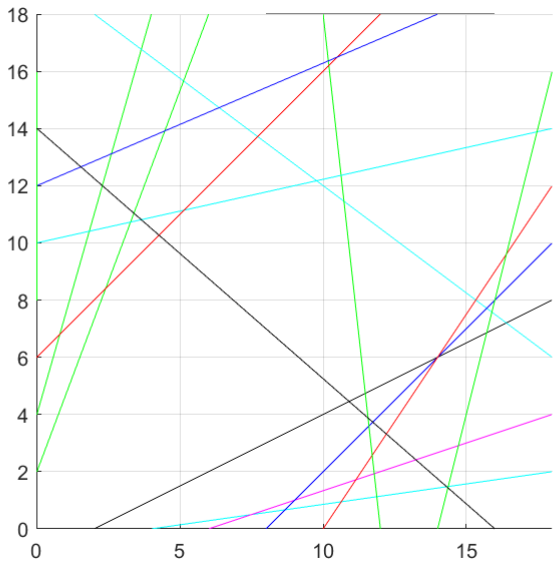

图 13

对第二小问，最终得到 m 最小值取 4 ，当 $m = 4$ 时，共有 1795 个解，我们将其储存在文件“solutions of 4.2.xls”中。其中的第一个解如图14所示，其中不同颜色表示导线置于不同层的电子线路板上：

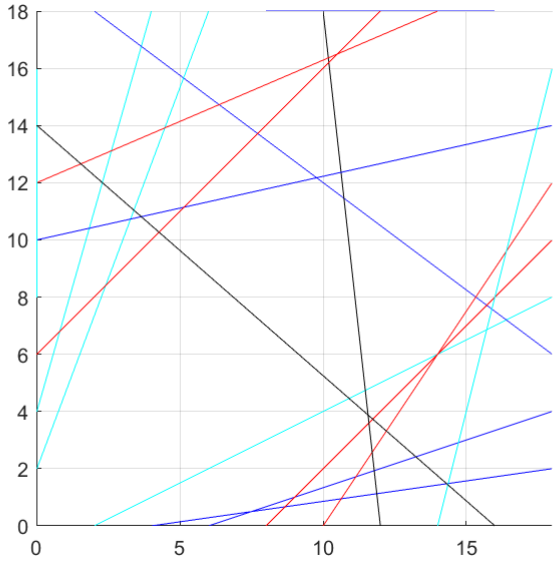

图 14

对第三小问，最终得到 m 最小值取 6 ，当 $m=4$ 时，共有 8 个解，我们将其储存在文件“solutions of 4.3.xls”中。其中的第一个解如图15所示，其中不同颜色表示导线置于不同层的电子线路板上：

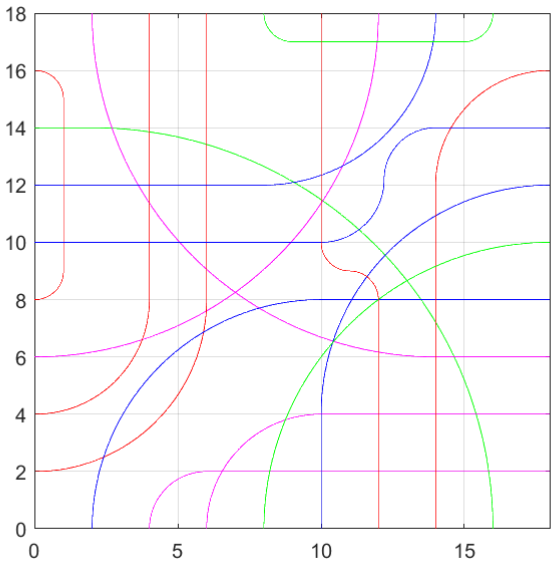

图 15

六、模型评价

6.1 模型优点

（1）由于回溯算法本质上是一个类似枚举的搜索尝试过程，因此能够保证答案的严格最优性，同时能够计算出所有该答案所对应的导线排布情况。

（2）在本题中，由于数据量较小，算法的计算速度较快，各个问题均在 1 秒以内。

6.2 模型缺点

（1）起始端口和终止端口都落在同一条边沿上，或起始端口和终止端口落在相对的边沿上的导线较多时，这种导线的走向与邻接矩阵的生成需要较多人工处理。

（2）算法的时间复杂度是指数级别的，在导线数量 n 较大时算法的效率不足。且当邻接矩阵比较稀疏时，给出解的效率一般情形下也不如穷举算法。

七、模型改进与推广

7.1 模型改进

经典的图着色问题（Graph Coloring Problem）是一个著名的NP-完全问题，即它不仅是一个多项式复杂程度的非确定性问题（即NP问题），并且如果它存在一个多项式时间的算法，则所有的NP问题均存在多项式时间的算法。而这一算法的存在与否被列为千禧年七大数学难题的第一位。因此目前还无法找到一个严格的优化算法，使得它的算法效率达到多项式级别，即目前最好的严格算法时间复杂度也只能达到指数级别，所以在保持解严格性的前提下，算法的时间复杂度优化空间似乎不大。

然而，在以下两种特殊情况下，上述模型与算法仍然存在不错或实用的改进：

（1）当导线的交点相对于导线的数量非常少时，生成的矩阵会比较稀疏，比较容易得到一个较好的解。此时若采用回溯算法，由于生成一个解至少需要 n 次判断，因此算法效率不如直接进行穷举。

（2）当导线数量 n 非常大时，由于算法的时间复杂度是指数级别的，效率有所不足，因此可以牺牲算法的严格性，采用Welch-Powell算法、贪心算法或遗传算法寻找一个较优的解。虽然该解可能不是全局最优的解，但可以将算法的时间复杂度降低至多项式级别，且在导线数量非常大时与全局最优解的偏离相对较小甚至可以忽略，仍具有不错的实用价值。

7.2 模型推广

（1）优化版本的图着色问题模型以及本题中对经典图着色问题模型的推广可以应用于安全装箱问题、电视频道分配问题、课程表问题、化学药剂放置问题及座位安排问题等求最小划分数使得某些元素不能放在同一集合中或限制放在同一集合中的某些元素数量的问题情形。

（2）回溯算法可以应用于八皇后问题、骑士游历问题、最大k乘积问题及购票排队问题等问题的解可以表示为有穷数组，且每个元组需要满足一些约束条件的问题情形。

参考文献

[1] 王晓东，计算机算法设计与分析（第四版），电子工业出版社，2012

附录

附录零：压缩包中各文件的含义

压缩文件中各文件的含义
solve1.m 第一题的主程序
solve2.m 第二题的主程序
solve3.m 第三题的主程序
solve4.1.m 第四题第一问的主程序
solve4.2.m 第四题第二问的主程序
solve4.3.m 第四题第三问的主程序
A3.m 第三题判断曲线相交情况的程序
A43.m 第四题第三问判断曲线相交情况的程序
draw3.m 第三题辅助画图
draw4.3.m 第四题第三问辅助画图
solution of 1.xlsx 第一题m取最小值的全部解
solution of 2.xlsx 第二题m取最小值的全部解
solution of 3.xlsx 第三题m取最小值的全部解
solution of 4.1.xlsx 第四题第一问m取最小值的全部解
solution of 4.2.xlsx 第四题第二问m取最小值的全部解
solution of 4.3.xlsx 第四题第三问m取最小值的全部解
1.xlsx 第一题m取最小值的全部解（未去0）
2.xlsx 第二题m取最小值的全部解（未去0）
3.xlsx 第三题m取最小值的全部解（未去0）
4.1.xlsx 第四题第一问m取最小值的全部解（未去0）
4.2.xlsx 第四题第二问m取最小值的全部解（未去0）
4.3.xlsx 第四题第三问m取最小值的全部解（未去0）

附录一：问题一的MATLAB程序代码（储存在solve1文件中）

```
% 第一题

% 导入题目中导线连接的数据
% 下面表示第条曲线的首末端点ici
c1 = [1, 10]; c2 = [9, 15]; c3 = [2, 5]; c4 = [8, 16]; c5 = [3, 7];
c6 = [4, 6]; c7 = [14, 17]; c8 = [13, 18]; c9 = [12, 20]; c10 = [11, 19];
% 将这些向量拼成一个矩阵，进而计算邻接矩阵
C = [c1; c2; c3; c4; c5; c6; c7; c8; c9; c10];

% 计算邻接矩阵
A = zeros(10, 10);

for i = 1:10

for j = 1:10
a_i = C(i, 1);
b_i = C(i, 2);
a_j = C(j, 1);
b_j = C(j, 2);
% 以下判断导线对(i,j)是否两两相交，是就把A(i,j)改为1
if a_j < min(a_i, b_i) && b_j > min(a_i, b_i) && b_j < max(a_i, b_i)
A(i, j) = 1;
elseif a_j > min(a_i, b_i) && a_j < max(a_i, b_i) && b_j > max(a_i, b_i)
A(i, j) = 1;
end

end

end

% 接下来，将条线按照与其它导线的交点数从大到小排列10
% 生成的向量的第位表示，重排后的第个导线原来是哪一位的Mii
MAX = sum(A);
M = [];

for i = 0:max(MAX)

for j = 1:10

if MAX(j) == max(MAX) - i
M = [M j];
end

end

```

```
47 end
48
49 % 再将矩阵的行和列按照中的顺序重排（进行合同变换）AM
50 % 先换行，为辅助换行的矩阵B
51 B = A;
52
53 for i = 1:10
54 MM = M(i);
55 A(i, :) = B(MM, :);
56 end
57
58 % 再换列，为辅助换列的矩阵B
59 B = A;
60
61 for i = 1:10
62 MM = M(i);
63 A(:, i) = B(:, MM);
64 end
65
66 x = 0; % 用来计数，最终输出的表示，在可以取到的最小的时的全体解的个数xxm
67 k = 0; % 在这一问中k相当于判断=0是否满足主子阵判据（之后Vk分别可以用来判断次、次次主子阵判据）
       =1,2
68 m_min = 0; % 表示最小可行的值，一开始赋m0
69 All_solution_of_m_min = zeros(1, 10); % 用来储存当取最小值时所有的可行数组mV
70
71 for m = 1:10
72 P = zeros(m, 16); % 为划分矩阵P
73 PT = zeros(m, 16); % 为可行的解密PTv取逆置换）后的数组(
74 u = zeros(1, m); % 为辅助判断某个是否满足主子阵判据的辅助变量uV
75 count = zeros(1, m); % 的第个元素表示的第行的非零元素有多少个ctivi
76 % 用来帮助计算下一个导线应该排在数组的哪个位置上ct
77 v0 = P;
78
79 % 以下就是回溯算法
80 % 先把第条导线分数组的某一行中iV
81 i1 = 1; %由对称性可知，将第一条导线置于数组的每一行都是等价的，不妨就放在第一行。
82 P = v0;
83 n = count(i1) + 1; % count(i1)表示此时第行非零元素的长度i1
84 P(i1, n) = 1; % 这表示将赋在第1行非零元素的末尾i1
85 v1 = P; % 用来储存此时的数组，以便之后可能的回溯v1
86 % 这部分是判断此时的是否满足主子阵判据V
87 for j = 1:m
88 w = P(j, :);
89 w(w == 0) = [];
90 u(j) = max(sum(A(w, w)));
91 maxu = max(u);
92 end
93
94 if maxu ≤ k
95 % 这一问中k，该条件满足代表此时的=0满足主子阵判据，才有进行下一步的必要V
```

```
96  count(i1) = count(i1) + 1;
97
98  for i2 = 1:m
99  P = v1;
100 n = count(i2) + 1;
101 P(i2, n) = 2;
102 % 紧接着的这个语句是为了避免最终输出结果的重复。if
103 if i2 > 1 && P(i2 - 1, 1) == 0
104 break
105 end
106
107 v2 = P;
108
109 for j = 1:m
110 w = P(j, :);
111 w(w == 0) = [];
112 u(j) = max(sum(A(w, w)));
113 maxu = max(u);
114 end
115
116 if maxu ≤ k
117 count(i2) = count(i2) + 1;
118
119 for i3 = 1:m
120 P = v2;
121 n = count(i3) + 1;
122 P(i3, n) = 3;
123
124 if i3 > 1 && P(i3 - 1, 1) == 0
125 break
126 end
127
128 v3 = P;
129
130 for j = 1:m
131 w = P(j, :);
132 w(w == 0) = [];
133 u(j) = max(sum(A(w, w)));
134 maxu = max(u);
135 end
136
137 if maxu ≤ k
138 count(i3) = count(i3) + 1;
139
140 for i4 = 1:m
141 P = v3;
142 n = count(i4) + 1;
143 P(i4, n) = 4;
144
145 if i4 > 1 && P(i4 - 1, 1) == 0
```

```
146 break
147 end
148
149 v4 = P;
150
151 for j = 1:m
152 w = P(j, :);
153 w(w == 0) = [];
154 u(j) = max(sum(A(w, w)));
155 maxu = max(u);
156 end
157
158 if maxu ≤ k
159 count(i4) = count(i4) + 1;
160
161 for i5 = 1:m
162 P = v4;
163 n = count(i5) + 1;
164 P(i5, n) = 5;
165
166 if i5 > 1 && P(i5 - 1, 1) == 0
167 break
168 end
169
170 v5 = P;
171
172 for j = 1:m
173 w = P(j, :);
174 w(w == 0) = [];
175 u(j) = max(sum(A(w, w)));
176 maxu = max(u);
177 end
178
179 if maxu ≤ k
180 count(i5) = count(i5) + 1;
181
182 for i6 = 1:m
183 P = v5;
184 n = count(i6) + 1;
185 P(i6, n) = 6;
186
187 if i6 > 1 && P(i6 - 1, 1) == 0
188 break
189 end
190
191 v6 = P;
192
193 for j = 1:m
194 w = P(j, :);
195 w(w == 0) = [];
```

```
196 u(j) = max(sum(A(w, w)));
197 maxu = max(u);
198 end
199
200 if maxu ≤ k
201 count(i6) = count(i6) + 1;
202
203 for i7 = 1:m
204 P = v6;
205 n = count(i7) + 1;
206 P(i7, n) = 7;
207
208 if i7 > 1 && P(i7 - 1, 1) == 0
209 break
210 end
211
212 v7 = P;
213
214 for j = 1:m
215 w = P(j, :);
216 w(w == 0) = [];
217 u(j) = max(sum(A(w, w)));
218 maxu = max(u);
219 end
220
221 if maxu ≤ k
222 count(i7) = count(i7) + 1;
223
224 for i8 = 1:m
225 P = v7;
226 n = count(i8) + 1;
227 P(i8, n) = 8;
228 v8 = P;
229
230 if i8 > 1 && P(i8 - 1, 1) == 0
231 break
232 end
233
234 for j = 1:m
235 w = P(j, :);
236 w(w == 0) = [];
237 u(j) = max(sum(A(w, w)));
238 maxu = max(u);
239 end
240
241 if maxu ≤ k
242 count(i8) = count(i8) + 1;
243
244 for i9 = 1:m
245 P = v8;
```

```
n = count(i9) + 1;
P(i9, n) = 9;

if i9 > 1 && P(i9 - 1, 1) == 0
break
end

v9 = P;

for j = 1:m
w = P(j, :);
w(w == 0) = [];
u(j) = max(sum(A(w, w)));
maxu = max(u);
end

if maxu ≤ k
count(i9) = count(i9) + 1;

for i10 = 1:m
P = v9;
n = count(i10) + 1;
P(i10, n) = 10;
v10 = P;

for j = 1:m
w = P(j, :);
w(w == 0) = [];
u(j) = max(sum(A(w, w)));
maxu = max(u);
end

if maxu ≤ k
% 一开始为m_min0
% 从小到大一一遍历，当的取值存在可行解之后，就将的值赋予mmmm_min，当,的值改变之后就停止循环。
    m_min
if m > m_min && m_min ≠ 0
All_solution_of_m_min(:, all(All_solution_of_m_min == 0)) = [] %把全的列去掉0
xlswrite('1.xlsx', All_solution_of_m_min)
% 在最终的前，在表格中输出所有当取最小值的满足要求的数组returnexcelmV
m_min
return
end

m_min = m;
x = x + 1;
% 以下这部分将可行的还原成重排前数组v
PT = zeros(m, 10);

for i = 1:m
```

```
295
296 for j = 1:10
297
298 if P(i, j) > 0
299 PT(i, j) = M(P(i, j));
300 end
301
302 end
303
304 end
305
306 % 把算出的解记录下来
307 line1 = (m + 1) * x - m;
308 line2 = (m + 1) * x - 1;
309 All_solution_of_m_min(line1:line2, :) = PT;
310
311 end
312
313 end
314
315 count(i9) = count(i9) - 1;
316 end
317
318 end
319
320 count(i8) = count(i8) - 1;
321 end
322
323 end
324
325 count(i7) = count(i7) - 1;
326 end
327
328 end
329
330 count(i6) = count(i6) - 1;
331 end
332
333 end
334
335 count(i5) = count(i5) - 1;
336 end
337
338 end
339
340 count(i4) = count(i4) - 1;
341 end
342
343 end
344
```

```
345  count(i3) = count(i3) - 1;
346  end
347
348  end
349
350  count(i2) = count(i2) - 1;
351  end
352
353  end
354
355  count(i1) = count(i1) - 1;
356  end
357
358  end
```

附录二：问题二的MATLAB程序代码（储存在solve2文件中）

```
% 第二题

% 导入题目中导线连接的数据
% 下面表示第条曲线的首末端点ici
c1 = [1, 10]; c2 = [9, 15]; c3 = [2, 5]; c4 = [8, 16]; c5 = [3, 7];
c6 = [4, 6]; c7 = [14, 17]; c8 = [13, 18]; c9 = [12, 20]; c10 = [11, 19];
% 将这些向量拼成一个矩阵，进而计算邻接矩阵
C = [c1; c2; c3; c4; c5; c6; c7; c8; c9; c10];

% 计算邻接矩阵
A = zeros(10, 10);

for i = 1:10

for j = 1:10
a_i = C(i, 1);
b_i = C(i, 2);
a_j = C(j, 1);
b_j = C(j, 2);
% 以下判断导线对(i,j)是否两两相交，是就把A(i,j)改为1
if a_j < min(a_i, b_i) && b_j > min(a_i, b_i) && b_j < max(a_i, b_i)
A(i, j) = 1;
elseif a_j > min(a_i, b_i) && a_j < max(a_i, b_i) && b_j > max(a_i, b_i)
A(i, j) = 1;
end

end

end

% 接下来，将条线按照与其它导线的交点数从大到小排列10
% 生成的向量的第位表示，重排后的第个导线原来是哪一位的Mii
MAX = sum(A);
M = [];

for i = 0:max(MAX)

for j = 1:10

if MAX(j) == max(MAX) - i
M = [M j];
end

end

```

```
47 end
48
49 % 再将矩阵的行和列按照中的顺序重排（进行合同变换）AM
50 % 先换行，为辅助换行的矩阵B
51 B = A;
52
53 for i = 1:10
54 MM = M(i);
55 A(i, :) = B(MM, :);
56 end
57
58 % 再换列，为辅助换列的矩阵B
59 B = A;
60
61 for i = 1:10
62 MM = M(i);
63 A(:, i) = B(:, MM);
64 end
65
66 x = 0; % 用来计数，最终输出的表示，在可以取到的最小的时的全体解的个数xxm
67 k = 1; % 在这一问中k相当于判断=1是否满足次主子阵判据V
68 m_min = 0; % 表示最小可行的值，一开始赋m0
69 All_solution_of_m_min = zeros(1, 10); % 用来储存当取最小值时所有的可行数组mV
70
71 for m = 1:10
72 P = zeros(m, 16); % 为划分矩阵P
73 PT = zeros(m, 16); % 为可行的解密PTv取逆置换）后的数组(
74 u = zeros(1, m); % 为辅助判断某个是否满足主子阵判据的辅助变量uV
75 count = zeros(1, m); % 的第个元素表示的第行的非零元素有多少个ctivi
76 % 用来帮助计算下一个导线应该排在数组的哪个位置上ct
77 v0 = P;
78
79 % 以下就是回溯算法
80 % 这部分是把第条导线分数组的某一行中，下同iV
81 i1 = 1; %由对称性可知，将第一条导线置于数组的每一行都是等价的，不妨就放在第一行。
82 P = v0;
83 n = count(i1) + 1; % count(i1)表示此时第行非零元素的长度i1
84 P(i1, n) = 1; % 这表示将赋在第1行非零元素的末尾i1
85 v1 = P; % 用来储存此时的数组，以便之后可能的回溯v1
86 % 这部分是判断此时的是否满足主子阵判据V
87 for j = 1:m
88 w = P(j, :);
89 w(w == 0) = [];
90 u(j) = max(sum(A(w, w)));
91 maxu = max(u);
92 end
93
94 if maxu ≤ k
95 % 这一问中k，该条件满足代表此时的=0满足主子阵判据，才有进行下一步的必要V
96 count(i1) = count(i1) + 1;
```

```
97
98  for i2 = 1:m
99  P = v1;
100 n = count(i2) + 1;
101 P(i2, n) = 2;
102 % 紧接着的这个语句是为了避免最终输出结果的重复。if
103 if i2 > 1 && P(i2 - 1, 1) == 0
104 break
105 end
106
107 v2 = P;
108
109 for j = 1:m
110 w = P(j, :);
111 w(w == 0) = [];
112 u(j) = max(sum(A(w, w)));
113 maxu = max(u);
114 end
115
116 if maxu ≤ k
117 count(i2) = count(i2) + 1;
118
119 for i3 = 1:m
120 P = v2;
121 n = count(i3) + 1;
122 P(i3, n) = 3;
123
124 if i3 > 1 && P(i3 - 1, 1) == 0
125 break
126 end
127
128 v3 = P;
129
130 for j = 1:m
131 w = P(j, :);
132 w(w == 0) = [];
133 u(j) = max(sum(A(w, w)));
134 maxu = max(u);
135 end
136
137 if maxu ≤ k
138 count(i3) = count(i3) + 1;
139
140 for i4 = 1:m
141 P = v3;
142 n = count(i4) + 1;
143 P(i4, n) = 4;
144
145 if i4 > 1 && P(i4 - 1, 1) == 0
146 break
```

```
147 end
148
149 v4 = P;
150
151 for j = 1:m
152 w = P(j, :);
153 w(w == 0) = [];
154 u(j) = max(sum(A(w, w)));
155 maxu = max(u);
156 end
157
158 if maxu ≤ k
159 count(i4) = count(i4) + 1;
160
161 for i5 = 1:m
162 P = v4;
163 n = count(i5) + 1;
164 P(i5, n) = 5;
165
166 if i5 > 1 && P(i5 - 1, 1) == 0
167 break
168 end
169
170 v5 = P;
171
172 for j = 1:m
173 w = P(j, :);
174 w(w == 0) = [];
175 u(j) = max(sum(A(w, w)));
176 maxu = max(u);
177 end
178
179 if maxu ≤ k
180 count(i5) = count(i5) + 1;
181
182 for i6 = 1:m
183 P = v5;
184 n = count(i6) + 1;
185 P(i6, n) = 6;
186
187 if i6 > 1 && P(i6 - 1, 1) == 0
188 break
189 end
190
191 v6 = P;
192
193 for j = 1:m
194 w = P(j, :);
195 w(w == 0) = [];
196 u(j) = max(sum(A(w, w)));
```

```
maxu = max(u);
end

if maxu ≤ k
count(i6) = count(i6) + 1;

for i7 = 1:m
P = v6;
n = count(i7) + 1;
P(i7, n) = 7;

if i7 > 1 && P(i7 - 1, 1) == 0
break
end

v7 = P;

for j = 1:m
w = P(j, :);
w(w == 0) = [];
u(j) = max(sum(A(w, w)));
maxu = max(u);
end

if maxu ≤ k
count(i7) = count(i7) + 1;

for i8 = 1:m
P = v7;
n = count(i8) + 1;
P(i8, n) = 8;
v8 = P;

if i8 > 1 && P(i8 - 1, 1) == 0
break
end

for j = 1:m
w = P(j, :);
w(w == 0) = [];
u(j) = max(sum(A(w, w)));
maxu = max(u);
end

if maxu ≤ k
count(i8) = count(i8) + 1;

for i9 = 1:m
P = v8;
n = count(i9) + 1;
```

```
P(i9, n) = 9;

if i9 > 1 && P(i9 - 1, 1) == 0
break
end

v9 = P;

for j = 1:m
w = P(j, :);
w(w == 0) = [];
u(j) = max(sum(A(w, w)));
maxu = max(u);
end

if maxu ≤ k
count(i9) = count(i9) + 1;

for i10 = 1:m
P = v9;
n = count(i10) + 1;
P(i10, n) = 10;
v10 = P;

for j = 1:m
w = P(j, :);
w(w == 0) = [];
u(j) = max(sum(A(w, w)));
maxu = max(u);
end

if maxu ≤ k
% 一开始为m_min0
% 从小到大一一遍历，当的取值存在可行解之后，就将的值赋予mmmm_min，当,的值改变之后就停止循环。
    m_min
if m > m_min && m_min ≠ 0
All_solution_of_m_min(:, all(All_solution_of_m_min == 0)) = [] %把全的列去掉0
xlswrite('2.xlsx', All_solution_of_m_min)
% 在最终的前，在表格中输出所有当取最小值的满足要求的数组returnexcelmV
m_min % 输出满足条件的最小的m
x % 输出当取最小值时满足条件的解的个数m
return
end

m_min = m;
x = x + 1;
% 以下这部分将可行的还原成重排前数组v
PT = zeros(m, 10);

for i = 1:m
```

```
296
297 for j = 1:10
298
299 if P(i, j) > 0
300 PT(i, j) = M(P(i, j));
301 end
302
303 end
304
305 end
306
307 % 把算出的解记录下来
308 line1 = (m + 1) * x - m;
309 line2 = (m + 1) * x - 1;
310 All_solution_of_m_min(line1:line2, :) = PT;
311
312 end
313
314 end
315
316 count(i9) = count(i9) - 1;
317 end
318
319 end
320
321 count(i8) = count(i8) - 1;
322 end
323
324 end
325
326 count(i7) = count(i7) - 1;
327 end
328
329 end
330
331 count(i6) = count(i6) - 1;
332 end
333
334 end
335
336 count(i5) = count(i5) - 1;
337 end
338
339 end
340
341 count(i4) = count(i4) - 1;
342 end
343
344 end
345
```

```
346  count(i3) = count(i3) - 1;
347  end
348
349  end
350
351  count(i2) = count(i2) - 1;
352  end
353
354  end
355
356  count(i1) = count(i1) - 1;
357  end
358
359  end
```

附录三：问题三的MATLAB程序代码（储存在solve3文件中）

```
1
2  % 第三题
3
4  % 导入题目中导线连接的数据
5  % 下面表示第条曲线的首末端点ici
6  c1 = [1, 10]; c2 = [9, 15]; c3 = [2, 5]; c4 = [8, 16]; c5 = [3, 7];
7  c6 = [4, 6]; c7 = [14, 17]; c8 = [13, 18]; c9 = [12, 20]; c10 = [11, 19];
8  % 将这些向量拼成一个矩阵，进而计算邻接矩阵
9  C = [c1; c2; c3; c4; c5; c6; c7; c8; c9; c10];
10
11 % 计算广义邻接矩阵,A_0(i,i)元素代表第条导向的转弯数i
12 A_0 = eye(10, 10);
13 A_0(3, 3) = 2;
14 A_0(4, 4) = 2;
15
16 for i = 1:10
17
18 for j = 1:10
19 a_i = C(i, 1);
20 b_i = C(i, 2);
21 a_j = C(j, 1);
22 b_j = C(j, 2);
23 % 以下判断导线对(i,j)是否两两相交，是就把A(i,j)改为1
24 if a_j < min(a_i, b_i) && b_j > min(a_i, b_i) && b_j < max(a_i, b_i)
25 A_0(i, j) = 1;
26 elseif a_j > min(a_i, b_i) && a_j < max(a_i, b_i) && b_j > max(a_i, b_i)
27 A_0(i, j) = 1;
28 end
29
30 end
31
32 end
33
34 A_0(1, 9) = 2;
35 A_0(1, 10) = 2;
36 A_0(9, 1) = 2;
37 A_0(10, 1) = 2;
38
39 % 接下来，将条线按照与其它导线的交点数从大到小排列10
40 % 生成的向量的第位表示，重排后的第个导线原来是哪一位的Mii
41 MAX = sum(A_0);
42 M = [];
43
44 for i = 0:max(MAX)
45
46 for j = 1:10
```

```
47
48  if MAX(j) == max(MAX) - i
49  M = [M j];
50  end
51
52  end
53
54  end
55
56  % 再将矩阵的行和列按照中的顺序重排（进行合同变换）AM
57  % 先换行，为辅助换行的矩阵B
58  B = A_0;
59
60  for i = 1:10
61  MM = M(i);
62  A_0(i, :) = B(MM, :);
63  end
64
65  % 再换列，为辅助换列的矩阵B
66  B = A_0;
67
68  for i = 1:10
69  MM = M(i);
70  A_0(:, i) = B(:, MM);
71  end
72
73  x = 0; % 用来计数，最终输出的表示，在可以取到的最小的时的全体解的个数xxm
74  k = 2; % 在这一问中k相当于判断=2是否满足次次主子阵判据V
75  m_min = 0; % 表示最小可行的值，一开始赋m0
76  All_solution_of_m_min = zeros(1, 10); % 用来储存当取最小值时所有的可行数组mV
77
78  for m = 1:10
79  P = zeros(m, 16); % 为划分矩阵P
80  PT = zeros(m, 16); % 为可行的解密PTv取逆置换）后的数组(
81  u = zeros(1, m); % 为辅助判断某个是否满足主子阵判据的辅助变量uV
82  count = zeros(1, m); % 的第个元素表示的第行的非零元素有多少个ctivi
83  % 用来帮助计算下一个导线应该排在数组的哪个位置上ct
84  v0 = P;
85
86  % 以下就是回溯算法
87  % 这部分是把第条导线分数组的某一行中，下同iV
88  i1 = 1; %由对称性可知，将第一条导线置于数组的每一行都是等价的，不妨就放在第一行。
89  P = v0;
90  n = count(i1) + 1; % count(i1)表示此时第行非零元素的长度i1
91  P(i1, n) = 1; % 这表示将赋在第1行非零元素的末尾i1
92  v1 = P; % 用来储存此时的数组，以便之后可能的回溯v1
93  % 这部分是判断此时的是否满足主子阵判据V
94  for j = 1:m
95  w = P(j, :);
96  w(w == 0) = [];
```

```
u(j) = max(sum(A_0(w, w)));
maxu = max(u);
end

if maxu ≤ k
% 这一问中k，该条件满足代表此时的=0满足主子阵判据，才有进行下一步的必要V
count(i1) = count(i1) + 1;

for i2 = 1:m
P = v1;
n = count(i2) + 1;
P(i2, n) = 2;
% 紧接着的这个语句是为了避免最终输出结果的重复。if
if i2 > 1 && P(i2 - 1, 1) == 0
break
end

v2 = P;

for j = 1:m
w = P(j, :);
w(w == 0) = [];
u(j) = max(sum(A_0(w, w)));
maxu = max(u);
end

if maxu ≤ k
count(i2) = count(i2) + 1;

for i3 = 1:m
P = v2;
n = count(i3) + 1;
P(i3, n) = 3;

if i3 > 1 && P(i3 - 1, 1) == 0
break
end

v3 = P;

for j = 1:m
w = P(j, :);
w(w == 0) = [];
u(j) = max(sum(A_0(w, w)));
maxu = max(u);
end

if maxu ≤ k
count(i3) = count(i3) + 1;

```

```
for i4 = 1:m
P = v3;
n = count(i4) + 1;
P(i4, n) = 4;

if i4 > 1 && P(i4 - 1, 1) == 0
break
end

v4 = P;

for j = 1:m
w = P(j, :);
w(w == 0) = [];
u(j) = max(sum(A_0(w, w)));
maxu = max(u);
end

if maxu ≤ k
count(i4) = count(i4) + 1;

for i5 = 1:m
P = v4;
n = count(i5) + 1;
P(i5, n) = 5;

if i5 > 1 && P(i5 - 1, 1) == 0
break
end

v5 = P;

for j = 1:m
w = P(j, :);
w(w == 0) = [];
u(j) = max(sum(A_0(w, w)));
maxu = max(u);
end

if maxu ≤ k
count(i5) = count(i5) + 1;

for i6 = 1:m
P = v5;
n = count(i6) + 1;
P(i6, n) = 6;

if i6 > 1 && P(i6 - 1, 1) == 0
break
end
```

```
197
198 v6 = P;
199
200 for j = 1:m
201 w = P(j, :);
202 w(w == 0) = [];
203 u(j) = max(sum(A_0(w, w)));
204 maxu = max(u);
205 end
206
207 if maxu ≤ k
208 count(i6) = count(i6) + 1;
209
210 for i7 = 1:m
211 P = v6;
212 n = count(i7) + 1;
213 P(i7, n) = 7;
214
215 if i7 > 1 && P(i7 - 1, 1) == 0
216 break
217 end
218
219 v7 = P;
220
221 for j = 1:m
222 w = P(j, :);
223 w(w == 0) = [];
224 u(j) = max(sum(A_0(w, w)));
225 maxu = max(u);
226 end
227
228 if maxu ≤ k
229 count(i7) = count(i7) + 1;
230
231 for i8 = 1:m
232 P = v7;
233 n = count(i8) + 1;
234 P(i8, n) = 8;
235 v8 = P;
236
237 if i8 > 1 && P(i8 - 1, 1) == 0
238 break
239 end
240
241 for j = 1:m
242 w = P(j, :);
243 w(w == 0) = [];
244 u(j) = max(sum(A_0(w, w)));
245 maxu = max(u);
246 end
```

```
247
248 if maxu ≤ k
249 count(i8) = count(i8) + 1;
250
251 for i9 = 1:m
252 P = v8;
253 n = count(i9) + 1;
254 P(i9, n) = 9;
255
256 if i9 > 1 && P(i9 - 1, 1) == 0
257 break
258 end
259
260 v9 = P;
261
262 for j = 1:m
263 w = P(j, :);
264 w(w == 0) = [];
265 u(j) = max(sum(A_0(w, w)));
266 maxu = max(u);
267 end
268
269 if maxu ≤ k
270 count(i9) = count(i9) + 1;
271
272 for i10 = 1:m
273 P = v9;
274 n = count(i10) + 1;
275 P(i10, n) = 10;
276 v10 = P;
277
278 for j = 1:m
279 w = P(j, :);
280 w(w == 0) = [];
281 u(j) = max(sum(A_0(w, w)));
282 maxu = max(u);
283 end
284
285 if maxu ≤ k
286 % 一开始为m_min0
287 % 从小到大一一遍历，当的取值存在可行解之后，就将的值赋予mmmm_min，当,的值改变之后就停止循环。
      m_min
288 if m > m_min && m_min ≠ 0
289 All_solution_of_m_min(:, all(All_solution_of_m_min == 0)) = [] %把全的列去掉0
290 xlswrite('3.xlsx', All_solution_of_m_min)
291 % 在最终的前，在表格中输出所有当取最小值的满足要求的数组returnexcelmV
292 m_min % 输出满足条件的最小的m
293 x % 输出当取最小值时满足条件的解的个数m
294 return
295 end
```

```
296
297 m_min = m;
298 x = x + 1;
299 % 以下这部分将可行的还原成重排前数组v
300 PT = zeros(m, 10);
301
302 for i = 1:m
303
304 for j = 1:10
305
306 if P(i, j) > 0
307 PT(i, j) = M(P(i, j));
308 end
309
310 end
311
312 end
313
314 % 把算出的解记录下来
315 line1 = (m + 1) * x - m;
316 line2 = (m + 1) * x - 1;
317 All_solution_of_m_min(line1:line2, :) = PT;
318
319 end
320
321 end
322
323 count(i9) = count(i9) - 1;
324 end
325
326 end
327
328 count(i8) = count(i8) - 1;
329 end
330
331 end
332
333 count(i7) = count(i7) - 1;
334 end
335
336 end
337
338 count(i6) = count(i6) - 1;
339 end
340
341 end
342
343 count(i5) = count(i5) - 1;
344 end
345
```

```
346  end
347
348  count(i4) = count(i4) - 1;
349  end
350
351  end
352
353  count(i3) = count(i3) - 1;
354  end
355
356  end
357
358  count(i2) = count(i2) - 1;
359  end
360
361  end
362
363  count(i1) = count(i1) - 1;
364  end
365
366  end
```

附录四：问题四的MATLAB程序代码

问题四第一问（储存在solve4.1文件中）

```
1
2  % 第四题第一小问
3
4  % 导入题目中导线连接的数据
5  % 下面表示第条曲线的首末端点ici
6  c1 = [1, 14]; c2 = [16, 23]; c3 = [15, 22]; c4 = [2, 26]; c5 = [13, 24]; c6 ...
       = [3, 25]; c7 = [12, 21]; c8 = [11, 20];
7  c9 = [4, 8]; c10 = [5, 19]; c11 = [10, 29]; c12 = [6, 27]; c13 = [7, 30]; ...
       c14 = [9, 18]; c15 = [28, 32]; c16 = [17, 31];
8  % 将这些向量拼成一个矩阵，进而计算邻接矩阵
9  C = [c1; c2; c3; c4; c5; c6; c7; c8; c9; c10; c11; c12; c13; c14; c15; c16];
10
11 % 计算邻接矩阵
12 A = zeros(16, 16);
13
14 for i = 1:16
15
16 for j = 1:16
17 a_i = C(i, 1);
18 b_i = C(i, 2);
19 a_j = C(j, 1);
20 b_j = C(j, 2);
21 % 以下判断导线对(i,j)是否两两相交，是就把A(i,j)改为1
22 if a_j < min(a_i, b_i) && b_j > min(a_i, b_i) && b_j < max(a_i, b_i)
23 A(i, j) = 1;
24 elseif a_j > min(a_i, b_i) && a_j < max(a_i, b_i) && b_j > max(a_i, b_i)
25 A(i, j) = 1;
26 end
27
28 end
29
30 end
31
32 % 接下来，将条线按照与其它导线的交点数从大到小排列16
33 % 生成的向量的第位表示，重排后的第个导线原来是哪一位的Mii
34 MAX = sum(A);
35 M = [];
36
37 for i = 0:max(MAX)
38
39 for j = 1:16
40
41 if MAX(j) == max(MAX) - i
42 M = [M j];
```

```
43 end
44
45 end
46
47 end
48
49 % 再将矩阵的行和列按照中的顺序重排（进行合同变换）AM
50 % 先换行，为辅助换行的矩阵B
51 B = A;
52
53 for i = 1:16
54 MM = M(i);
55 A(i, :) = B(MM, :);
56 end
57
58 % 再换列，为辅助换列的矩阵B
59 B = A;
60
61 for i = 1:16
62 MM = M(i);
63 A(:, i) = B(:, MM);
64 end
65
66 x = 0; % 用来计数，最终输出的表示，在可以取到的最小的时的全体解的个数xxm
67 k = 0; % 在这一问中k相当于判断=0是否满足主子阵判据（之后Vk分别可以用来判断次、次次主子阵判据）
       =1,2
68 m_min = 0; % 表示最小可行的值，一开始赋m0
69 All_solution_of_m_min = zeros(1, 16); % 用来储存当取最小值时所有的可行数组mV
70
71 for m = 1:16
72 P = zeros(m, 16); % 为划分矩阵P
73 PT = zeros(m, 16); % 为可行的解密PTv取逆置换）后的数组(
74 u = zeros(1, m); % 为辅助判断某个是否满足主子阵判据的辅助变量uV
75 count = zeros(1, m); % 的第个元素表示的第行的非零元素有多少个ctivi
76 % 用来帮助计算下一个导线应该排在数组的哪个位置上ct
77 v0 = P;
78
79 % 以下就是回溯算法
80 % 这部分是把第条导线分数组的某一行中，下同iV
81 i1 = 1; %由对称性可知，将第一条导线置于数组的每一行都是等价的，不妨就放在第一行。
82 P = v0;
83 n = count(i1) + 1; % count(i1)表示此时第行非零元素的长度i1
84 P(i1, n) = 1; % 这表示将赋在第1行非零元素的末尾i1
85 v1 = P; % 用来储存此时的数组，以便之后可能的回溯v1
86 % 这部分是判断此时的是否满足主子阵判据V
87 for j = 1:m
88 w = P(j, :);
89 w(w == 0) = [];
90 u(j) = max(sum(A(w, w)));
91 maxu = max(u);
```

```
92  end
93
94  if maxu ≤ k
95  % 这一问中k，该条件满足代表此时的=0满足主子阵判据，才有进行下一步的必要V
96  count(i1) = count(i1) + 1;
97
98  for i2 = 1:m
99  P = v1;
100 n = count(i2) + 1;
101 P(i2, n) = 2;
102 % 紧接着的这个语句是为了避免最终输出结果的重复。if
103 if i2 > 1 && P(i2 - 1, 1) == 0
104 break
105 end
106
107 v2 = P;
108
109 for j = 1:m
110 w = P(j, :);
111 w(w == 0) = [];
112 u(j) = max(sum(A(w, w)));
113 maxu = max(u);
114 end
115
116 if maxu ≤ k
117 count(i2) = count(i2) + 1;
118
119 for i3 = 1:m
120 P = v2;
121 n = count(i3) + 1;
122 P(i3, n) = 3;
123
124 if i3 > 1 && P(i3 - 1, 1) == 0
125 break
126 end
127
128 v3 = P;
129
130 for j = 1:m
131 w = P(j, :);
132 w(w == 0) = [];
133 u(j) = max(sum(A(w, w)));
134 maxu = max(u);
135 end
136
137 if maxu ≤ k
138 count(i3) = count(i3) + 1;
139
140 for i4 = 1:m
141 P = v3;
```

```
142 n = count(i4) + 1;
143 P(i4, n) = 4;
144
145 if i4 > 1 && P(i4 - 1, 1) == 0
146 break
147 end
148
149 v4 = P;
150
151 for j = 1:m
152 w = P(j, :);
153 w(w == 0) = [];
154 u(j) = max(sum(A(w, w)));
155 maxu = max(u);
156 end
157
158 if maxu ≤ k
159 count(i4) = count(i4) + 1;
160
161 for i5 = 1:m
162 P = v4;
163 n = count(i5) + 1;
164 P(i5, n) = 5;
165
166 if i5 > 1 && P(i5 - 1, 1) == 0
167 break
168 end
169
170 v5 = P;
171
172 for j = 1:m
173 w = P(j, :);
174 w(w == 0) = [];
175 u(j) = max(sum(A(w, w)));
176 maxu = max(u);
177 end
178
179 if maxu ≤ k
180 count(i5) = count(i5) + 1;
181
182 for i6 = 1:m
183 P = v5;
184 n = count(i6) + 1;
185 P(i6, n) = 6;
186
187 if i6 > 1 && P(i6 - 1, 1) == 0
188 break
189 end
190
191 v6 = P;
```

```
192
193 for j = 1:m
194 w = P(j, :);
195 w(w == 0) = [];
196 u(j) = max(sum(A(w, w)));
197 maxu = max(u);
198 end
199
200 if maxu ≤ k
201 count(i6) = count(i6) + 1;
202
203 for i7 = 1:m
204 P = v6;
205 n = count(i7) + 1;
206 P(i7, n) = 7;
207
208 if i7 > 1 && P(i7 - 1, 1) == 0
209 break
210 end
211
212 v7 = P;
213
214 for j = 1:m
215 w = P(j, :);
216 w(w == 0) = [];
217 u(j) = max(sum(A(w, w)));
218 maxu = max(u);
219 end
220
221 if maxu ≤ k
222 count(i7) = count(i7) + 1;
223
224 for i8 = 1:m
225 P = v7;
226 n = count(i8) + 1;
227 P(i8, n) = 8;
228 v8 = P;
229
230 if i8 > 1 && P(i8 - 1, 1) == 0
231 break
232 end
233
234 for j = 1:m
235 w = P(j, :);
236 w(w == 0) = [];
237 u(j) = max(sum(A(w, w)));
238 maxu = max(u);
239 end
240
241 if maxu ≤ k
```

```
count(i8) = count(i8) + 1;

for i9 = 1:m
P = v8;
n = count(i9) + 1;
P(i9, n) = 9;

if i9 > 1 && P(i9 - 1, 1) == 0
break
end

v9 = P;

for j = 1:m
w = P(j, :);
w(w == 0) = [];
u(j) = max(sum(A(w, w)));
maxu = max(u);
end

if maxu ≤ k
count(i9) = count(i9) + 1;

for i10 = 1:m
P = v9;
n = count(i10) + 1;
P(i10, n) = 10;
v10 = P;

if i10 > 1 && P(i10 - 1, 1) == 0
break
end

for j = 1:m
w = P(j, :);
w(w == 0) = [];
u(j) = max(sum(A(w, w)));
maxu = max(u);
end

if maxu ≤ k
count(i10) = count(i10) + 1;

for i11 = 1:m
P = v10;
n = count(i11) + 1;
P(i11, n) = 11;
v11 = P;

if i11 > 1 && P(i11 - 1, 1) == 0
```

```
292 break
293 end
294
295 for j = 1:m
296 w = P(j, :);
297 w(w == 0) = [];
298 u(j) = max(sum(A(w, w)));
299 maxu = max(u);
300 end
301
302 if maxu ≤ k
303 count(i11) = count(i11) + 1;
304
305 for i12 = 1:m
306 P = v11;
307 n = count(i12) + 1;
308 P(i12, n) = 12;
309 v12 = P;
310
311 if i12 > 1 && P(i12 - 1, 1) == 0
312 break
313 end
314
315 for j = 1:m
316 w = P(j, :);
317 w(w == 0) = [];
318 u(j) = max(sum(A(w, w)));
319 maxu = max(u);
320 end
321
322 if maxu ≤ k
323 count(i12) = count(i12) + 1;
324
325 for i13 = 1:m
326 P = v12;
327 n = count(i13) + 1;
328 P(i13, n) = 13;
329 v13 = P;
330
331 if i13 > 1 && P(i13 - 1, 1) == 0
332 break
333 end
334
335 for j = 1:m
336 w = P(j, :);
337 w(w == 0) = [];
338 u(j) = max(sum(A(w, w)));
339 maxu = max(u);
340 end
341
```

```
342 if maxu ≤ k
343 count(i13) = count(i13) + 1;
344
345 for i14 = 1:m
346 P = v13;
347 n = count(i14) + 1;
348 P(i14, n) = 14;
349 v14 = P;
350
351 if i14 > 1 && P(i14 - 1, 1) == 0
352 break
353 end
354
355 for j = 1:m
356 w = P(j, :);
357 w(w == 0) = [];
358 u(j) = max(sum(A(w, w)));
359 maxu = max(u);
360 end
361
362 if maxu ≤ k
363 count(i14) = count(i14) + 1;
364
365 for i15 = 1:m
366 P = v14;
367 n = count(i15) + 1;
368 P(i15, n) = 15;
369 v15 = P;
370
371 if i15 > 1 && P(i15 - 1, 1) == 0
372 break
373 end
374
375 for j = 1:m
376 w = P(j, :);
377 w(w == 0) = [];
378 u(j) = max(sum(A(w, w)));
379 maxu = max(u);
380 end
381
382 if maxu ≤ k
383 count(i15) = count(i15) + 1;
384
385 for i16 = 1:m
386 P = v15;
387 n = count(i16) + 1;
388 P(i16, n) = 16;
389 v16 = P;
390
391 for j = 1:m
```

```
w = P(j, :);
w(w == 0) = [];
u(j) = max(sum(A(w, w)));
maxu = max(u);
end

if maxu ≤ k
% 一开始为m_min0
% 从小到大一一遍历，当的取值存在可行解之后，就将的值赋予mmmm_min，当,的值改变之后就停止循环。
    m_min
if m > m_min && m_min ≠ 0
All_solution_of_m_min(:, all(All_solution_of_m_min == 0)) = []; %把全的列去掉0
xlswrite('41.xlsx', All_solution_of_m_min)
% 在最终的前，在表格中输出所有当取最小值的满足要求的数组returnexcelmV
m_min % 输出满足条件的最小的m
x % 输出当取最小值时满足条件的解的个数m
return
end

m_min = m;
x = x + 1;
% 以下这部分将可行的还原成重排前数组v
PT = zeros(m, 16);

for i = 1:m

for j = 1:16

if P(i, j) > 0
PT(i, j) = M(P(i, j));
end

end

end

% 把算出的解记录下来
line1 = (m + 1) * x - m;
line2 = (m + 1) * x - 1;
All_solution_of_m_min(line1:line2, :) = PT;

end

end

count(i15) = count(i15) - 1;
end

end

```

```
441  count(i14) = count(i14) - 1;
442  end
443
444  end
445
446  count(i13) = count(i13) - 1;
447  end
448
449  end
450
451  count(i12) = count(i12) - 1;
452  end
453
454  end
455
456  count(i11) = count(i11) - 1;
457  end
458
459  end
460
461  count(i10) = count(i10) - 1;
462  end
463
464  end
465
466  count(i9) = count(i9) - 1;
467  end
468
469  end
470
471  count(i8) = count(i8) - 1;
472  end
473
474  end
475
476  count(i7) = count(i7) - 1;
477  end
478
479  end
480
481  count(i6) = count(i6) - 1;
482  end
483
484  end
485
486  count(i5) = count(i5) - 1;
487  end
488
489  end
490
```

```
491  count(i4) = count(i4) - 1;
492  end
493
494  end
495
496  count(i3) = count(i3) - 1;
497  end
498
499  end
500
501  count(i2) = count(i2) - 1;
502  end
503
504  end
505
506  count(i1) = count(i1) - 1;
507  end
508
509  end
```

问题四第二问（储存在solve4.2文件中）

```
1
2  % 第四题第二小问
3
4  % 导入题目中导线连接的数据
5  % 下面表示第条曲线的首末端点ici
6  c1 = [1, 14]; c2 = [16, 23]; c3 = [15, 22]; c4 = [2, 26]; c5 = [13, 24]; c6 ...
       = [3, 25]; c7 = [12, 21]; c8 = [11, 20];
7  c9 = [4, 8]; c10 = [5, 19]; c11 = [10, 29]; c12 = [6, 27]; c13 = [7, 30]; ...
       c14 = [9, 18]; c15 = [28, 32]; c16 = [17, 31];
8  % 将这些向量拼成一个矩阵，进而计算邻接矩阵
9  C = [c1; c2; c3; c4; c5; c6; c7; c8; c9; c10; c11; c12; c13; c14; c15; c16];
10
11 % 计算邻接矩阵
12 A = zeros(16, 16);
13
14 for i = 1:16
15
16 for j = 1:16
17 a_i = C(i, 1);
18 b_i = C(i, 2);
19 a_j = C(j, 1);
20 b_j = C(j, 2);
21 % 以下判断导线对(i,j)是否两两相交，是就把A(i,j)改为1
22 if a_j < min(a_i, b_i) && b_j > min(a_i, b_i) && b_j < max(a_i, b_i)
23 A(i, j) = 1;
24 elseif a_j > min(a_i, b_i) && a_j < max(a_i, b_i) && b_j > max(a_i, b_i)
25 A(i, j) = 1;
26 end
27
28 end
29
30 end
31
32 % 接下来，将条线按照与其它导线的交点数从大到小排列16
33 % 生成的向量的第位表示，重排后的第个导线原来是哪一位的Mii
34 MAX = sum(A);
35 M = [];
36
37 for i = 0:max(MAX)
38
39 for j = 1:16
40
41 if MAX(j) == max(MAX) - i
42 M = [M j];
43 end
44
```

```
45 end
46
47 end
48
49 % 再将矩阵的行和列按照中的顺序重排（进行合同变换）AM
50 % 先换行，为辅助换行的矩阵B
51 B = A;
52
53 for i = 1:16
54 MM = M(i);
55 A(i, :) = B(MM, :);
56 end
57
58 % 再换列，为辅助换列的矩阵B
59 B = A;
60
61 for i = 1:16
62 MM = M(i);
63 A(:, i) = B(:, MM);
64 end
65
66 x = 0; % 用来计数，最终输出的表示，在可以取到的最小的时的全体解的个数xxm
67 k = 1; % 在这一问中k相当于判断=0是否满足次主子阵判据V
68 m_min = 0; % 表示最小可行的值，一开始赋m0
69 All_solution_of_m_min = zeros(1, 16); % 用来储存当取最小值时所有的可行数组mV
70
71 for m = 1:16
72 P = zeros(m, 16); % 为划分矩阵P
73 PT = zeros(m, 16); % 为可行的解密PTv取逆置换）后的数组(
74 u = zeros(1, m); % 为辅助判断某个是否满足主子阵判据的辅助变量uV
75 count = zeros(1, m); % 的第个元素表示的第行的非零元素有多少个ctivi
76 % 用来帮助计算下一个导线应该排在数组的哪个位置上ct
77 v0 = P;
78
79 % 以下就是回溯算法
80 % 这部分是把第条导线分数组的某一行中，下同iV
81 i1 = 1; %由对称性可知，将第一条导线置于数组的每一行都是等价的，不妨就放在第一行。
82 P = v0;
83 n = count(i1) + 1; % count(i1)表示此时第行非零元素的长度i1
84 P(i1, n) = 1; % 这表示将赋在第1行非零元素的末尾i1
85 v1 = P; % 用来储存此时的数组，以便之后可能的回溯v1
86 % 这部分是判断此时的是否满足主子阵判据V
87 for j = 1:m
88 w = P(j, :);
89 w(w == 0) = [];
90 u(j) = max(sum(A(w, w)));
91 maxu = max(u);
92 end
93
94 if maxu ≤ k
```

```
95  % 这一问中k，该条件满足代表此时的=0满足主子阵判据，才有进行下一步的必要V
96  count(i1) = count(i1) + 1;
97
98  for i2 = 1:m
99  P = v1;
100 n = count(i2) + 1;
101 P(i2, n) = 2;
102 % 紧接着的这个语句是为了避免最终输出结果的重复。if
103 if i2 > 1 && P(i2 - 1, 1) == 0
104 break
105 end
106
107 v2 = P;
108
109 for j = 1:m
110 w = P(j, :);
111 w(w == 0) = [];
112 u(j) = max(sum(A(w, w)));
113 maxu = max(u);
114 end
115
116 if maxu ≤ k
117 count(i2) = count(i2) + 1;
118
119 for i3 = 1:m
120 P = v2;
121 n = count(i3) + 1;
122 P(i3, n) = 3;
123
124 if i3 > 1 && P(i3 - 1, 1) == 0
125 break
126 end
127
128 v3 = P;
129
130 for j = 1:m
131 w = P(j, :);
132 w(w == 0) = [];
133 u(j) = max(sum(A(w, w)));
134 maxu = max(u);
135 end
136
137 if maxu ≤ k
138 count(i3) = count(i3) + 1;
139
140 for i4 = 1:m
141 P = v3;
142 n = count(i4) + 1;
143 P(i4, n) = 4;
144
```

```
145 if i4 > 1 && P(i4 - 1, 1) == 0
146 break
147 end
148
149 v4 = P;
150
151 for j = 1:m
152 w = P(j, :);
153 w(w == 0) = [];
154 u(j) = max(sum(A(w, w)));
155 maxu = max(u);
156 end
157
158 if maxu ≤ k
159 count(i4) = count(i4) + 1;
160
161 for i5 = 1:m
162 P = v4;
163 n = count(i5) + 1;
164 P(i5, n) = 5;
165
166 if i5 > 1 && P(i5 - 1, 1) == 0
167 break
168 end
169
170 v5 = P;
171
172 for j = 1:m
173 w = P(j, :);
174 w(w == 0) = [];
175 u(j) = max(sum(A(w, w)));
176 maxu = max(u);
177 end
178
179 if maxu ≤ k
180 count(i5) = count(i5) + 1;
181
182 for i6 = 1:m
183 P = v5;
184 n = count(i6) + 1;
185 P(i6, n) = 6;
186
187 if i6 > 1 && P(i6 - 1, 1) == 0
188 break
189 end
190
191 v6 = P;
192
193 for j = 1:m
194 w = P(j, :);
```

```
w(w == 0) = [];
u(j) = max(sum(A(w, w)));
maxu = max(u);
end

if maxu ≤ k
count(i6) = count(i6) + 1;

for i7 = 1:m
P = v6;
n = count(i7) + 1;
P(i7, n) = 7;

if i7 > 1 && P(i7 - 1, 1) == 0
break
end

v7 = P;

for j = 1:m
w = P(j, :);
w(w == 0) = [];
u(j) = max(sum(A(w, w)));
maxu = max(u);
end

if maxu ≤ k
count(i7) = count(i7) + 1;

for i8 = 1:m
P = v7;
n = count(i8) + 1;
P(i8, n) = 8;
v8 = P;

if i8 > 1 && P(i8 - 1, 1) == 0
break
end

for j = 1:m
w = P(j, :);
w(w == 0) = [];
u(j) = max(sum(A(w, w)));
maxu = max(u);
end

if maxu ≤ k
count(i8) = count(i8) + 1;

for i9 = 1:m
```

```
P = v8;
n = count(i9) + 1;
P(i9, n) = 9;

if i9 > 1 && P(i9 - 1, 1) == 0
break
end

v9 = P;

for j = 1:m
w = P(j, :);
w(w == 0) = [];
u(j) = max(sum(A(w, w)));
maxu = max(u);
end

if maxu ≤ k
count(i9) = count(i9) + 1;

for i10 = 1:m
P = v9;
n = count(i10) + 1;
P(i10, n) = 10;
v10 = P;

if i10 > 1 && P(i10 - 1, 1) == 0
break
end

for j = 1:m
w = P(j, :);
w(w == 0) = [];
u(j) = max(sum(A(w, w)));
maxu = max(u);
end

if maxu ≤ k
count(i10) = count(i10) + 1;

for i11 = 1:m
P = v10;
n = count(i11) + 1;
P(i11, n) = 11;
v11 = P;

if i11 > 1 && P(i11 - 1, 1) == 0
break
end

```

```
for j = 1:m
w = P(j, :);
w(w == 0) = [];
u(j) = max(sum(A(w, w)));
maxu = max(u);
end

if maxu ≤ k
count(i11) = count(i11) + 1;

for i12 = 1:m
P = v11;
n = count(i12) + 1;
P(i12, n) = 12;
v12 = P;

if i12 > 1 && P(i12 - 1, 1) == 0
break
end

for j = 1:m
w = P(j, :);
w(w == 0) = [];
u(j) = max(sum(A(w, w)));
maxu = max(u);
end

if maxu ≤ k
count(i12) = count(i12) + 1;

for i13 = 1:m
P = v12;
n = count(i13) + 1;
P(i13, n) = 13;
v13 = P;

if i13 > 1 && P(i13 - 1, 1) == 0
break
end

for j = 1:m
w = P(j, :);
w(w == 0) = [];
u(j) = max(sum(A(w, w)));
maxu = max(u);
end

if maxu ≤ k
count(i13) = count(i13) + 1;

```

```
for i14 = 1:m
P = v13;
n = count(i14) + 1;
P(i14, n) = 14;
v14 = P;

if i14 > 1 && P(i14 - 1, 1) == 0
break
end

for j = 1:m
w = P(j, :);
w(w == 0) = [];
u(j) = max(sum(A(w, w)));
maxu = max(u);
end

if maxu ≤ k
count(i14) = count(i14) + 1;

for i15 = 1:m
P = v14;
n = count(i15) + 1;
P(i15, n) = 15;
v15 = P;

if i15 > 1 && P(i15 - 1, 1) == 0
break
end

for j = 1:m
w = P(j, :);
w(w == 0) = [];
u(j) = max(sum(A(w, w)));
maxu = max(u);
end

if maxu ≤ k
count(i15) = count(i15) + 1;

for i16 = 1:m
P = v15;
n = count(i16) + 1;
P(i16, n) = 16;
v16 = P;

for j = 1:m
w = P(j, :);
w(w == 0) = [];
u(j) = max(sum(A(w, w)));
```

```
maxu = max(u);
end

if maxu ≤ k
% 一开始为m_min0
% 从小到大一一遍历，当的取值存在可行解之后，就将的值赋予mmmm_min，当,的值改变之后就停止循环。
    m_min
if m > m_min && m_min ≠ 0
All_solution_of_m_min(:, all(All_solution_of_m_min == 0)) = []; %把全的列去掉0
xlswrite('42.xlsx', All_solution_of_m_min)
% 在最终的前，在表格中输出所有当取最小值的满足要求的数组returnexcelmV
m_min % 输出满足条件的最小的m
x % 输出当取最小值时满足条件的解的个数m
return
end

m_min = m;
x = x + 1;
% 以下这部分将可行的还原成重排前数组v
PT = zeros(m, 16);

for i = 1:m

for j = 1:16

if P(i, j) > 0
PT(i, j) = M(P(i, j));
end

end

end

% 把算出的解记录下来
line1 = (m + 1) * x - m;
line2 = (m + 1) * x - 1;
All_solution_of_m_min(line1:line2, :) = PT;

end

end

count(i15) = count(i15) - 1;
end

end

count(i14) = count(i14) - 1;
end

```

```
444  end
445
446  count(i13) = count(i13) - 1;
447  end
448
449  end
450
451  count(i12) = count(i12) - 1;
452  end
453
454  end
455
456  count(i11) = count(i11) - 1;
457  end
458
459  end
460
461  count(i10) = count(i10) - 1;
462  end
463
464  end
465
466  count(i9) = count(i9) - 1;
467  end
468
469  end
470
471  count(i8) = count(i8) - 1;
472  end
473
474  end
475
476  count(i7) = count(i7) - 1;
477  end
478
479  end
480
481  count(i6) = count(i6) - 1;
482  end
483
484  end
485
486  count(i5) = count(i5) - 1;
487  end
488
489  end
490
491  count(i4) = count(i4) - 1;
492  end
493
```

```
494  end
495
496  count(i3) = count(i3) - 1;
497  end
498
499  end
500
501  count(i2) = count(i2) - 1;
502  end
503
504  end
505
506  count(i1) = count(i1) - 1;
507  end
508
509  end
```

问题四第三问（储存在solve4.3文件中）

```
1
2  % 第四题第三小问
3
4  % 导入题目中导线连接的数据
5  % 下面表示第条曲线的首末端点ici
6  c1 = [1, 14]; c2 = [16, 23]; c3 = [15, 22]; c4 = [2, 26]; c5 = [13, 24]; c6 ...
       = [3, 25]; c7 = [12, 21]; c8 = [11, 20];
7  c9 = [4, 8]; c10 = [5, 19]; c11 = [10, 29]; c12 = [6, 27]; c13 = [7, 30]; ...
       c14 = [9, 18]; c15 = [28, 32]; c16 = [17, 31];
8  % 将这些向量拼成一个矩阵，进而计算邻接矩阵
9  C = [c1; c2; c3; c4; c5; c6; c7; c8; c9; c10; c11; c12; c13; c14; c15; c16];
10
11 % 计算广义邻接矩阵,A_0(i,i)元素代表第条导向的转弯数i
12 A_0 = eye(16, 16);
13
14 A_0(9, 9) = 2;
15 A_0(10, 10) = 2;
16 A_0(11, 11) = 2;
17 A_0(15, 15) = 2;
18
19 for i = 1:16
20
21 for j = 1:16
22 a_i = C(i, 1);
23 b_i = C(i, 2);
24 a_j = C(j, 1);
25 b_j = C(j, 2);
26 % 以下判断导线对(i,j)是否两两相交，是就把A(i,j)改为1
27 if a_j < min(a_i, b_i) && b_j > min(a_i, b_i) && b_j < max(a_i, b_i)
28 A_0(i, j) = 1;
29 elseif a_j > min(a_i, b_i) && a_j < max(a_i, b_i) && b_j > max(a_i, b_i)
30 A_0(i, j) = 1;
31 end
32
33 end
34
35 end
36
37 A_0(16, 1) = 2;
38 A_0(1, 16) = 2;
39
40 % 接下来，将条线按照与其它导线的交点数从大到小排列16
41 % 生成的向量的第位表示，重排后的第个导线原来是哪一位的Mii
42 MAX = sum(A_0);
43 M = [];
44
```

```
for i = 0:max(MAX)

for j = 1:16

if MAX(j) == max(MAX) - i
M = [M j];
end

end

end

% 再将矩阵的行和列按照中的顺序重排（进行合同变换）AM
% 先换行，为辅助换行的矩阵B
B = A_0;

for i = 1:16
MM = M(i);
A_0(i, :) = B(MM, :);
end

% 再换列，为辅助换列的矩阵B
B = A_0;

for i = 1:16
MM = M(i);
A_0(:, i) = B(:, MM);
end

x = 0; % 用来计数，最终输出的表示，在可以取到的最小的时的全体解的个数xxm
k = 2; % 在这一问中k相当于判断=0是否满足次次主子阵判据V
m_min = 0; % 表示最小可行的值，一开始赋m0
All_solution_of_m_min = zeros(1, 16); % 用来储存当取最小值时所有的可行数组mV

for m = 1:16
P = zeros(m, 16); % 为划分矩阵P
PT = zeros(m, 16); % 为可行的解密PTv取逆置换）后的数组(
u = zeros(1, m); % 为辅助判断某个是否满足主子阵判据的辅助变量uV
count = zeros(1, m); % 的第个元素表示的第行的非零元素有多少个ctivi
% 用来帮助计算下一个导线应该排在数组的哪个位置上ct
v0 = P;

% 以下就是回溯算法
% 这部分是把第条导线分数组的某一行中，下同iV
i1 = 1; %由对称性可知，将第一条导线置于数组的每一行都是等价的，不妨就放在第一行。
P = v0;
n = count(i1) + 1; % count(i1)表示此时第行非零元素的长度i1
P(i1, n) = 1; % 这表示将赋在第1行非零元素的末尾i1
v1 = P; % 用来储存此时的数组，以便之后可能的回溯v1
% 这部分是判断此时的是否满足主子阵判据V
```

```
for j = 1:m
w = P(j, :);
w(w == 0) = [];
u(j) = max(sum(A_0(w, w)));
maxu = max(u);
end

if maxu ≤ k
% 这一问中k，该条件满足代表此时的=2满足次次主子阵判据，才有进行下一步的必要V
count(i1) = count(i1) + 1;

for i2 = 1:m
P = v1;
n = count(i2) + 1;
P(i2, n) = 2;
% 紧接着的这个语句是为了避免最终输出结果的重复。if
if i2 > 1 && P(i2 - 1, 1) == 0
break
end

v2 = P;

for j = 1:m
w = P(j, :);
w(w == 0) = [];
u(j) = max(sum(A_0(w, w)));
maxu = max(u);
end

if maxu ≤ k
count(i2) = count(i2) + 1;

for i3 = 1:m
P = v2;
n = count(i3) + 1;
P(i3, n) = 3;

if i3 > 1 && P(i3 - 1, 1) == 0
break
end

v3 = P;

for j = 1:m
w = P(j, :);
w(w == 0) = [];
u(j) = max(sum(A_0(w, w)));
maxu = max(u);
end

```

```
145 if maxu ≤ k
146 count(i3) = count(i3) + 1;
147
148 for i4 = 1:m
149 P = v3;
150 n = count(i4) + 1;
151 P(i4, n) = 4;
152
153 if i4 > 1 && P(i4 - 1, 1) == 0
154 break
155 end
156
157 v4 = P;
158
159 for j = 1:m
160 w = P(j, :);
161 w(w == 0) = [];
162 u(j) = max(sum(A_0(w, w)));
163 maxu = max(u);
164 end
165
166 if maxu ≤ k
167 count(i4) = count(i4) + 1;
168
169 for i5 = 1:m
170 P = v4;
171 n = count(i5) + 1;
172 P(i5, n) = 5;
173
174 if i5 > 1 && P(i5 - 1, 1) == 0
175 break
176 end
177
178 v5 = P;
179
180 for j = 1:m
181 w = P(j, :);
182 w(w == 0) = [];
183 u(j) = max(sum(A_0(w, w)));
184 maxu = max(u);
185 end
186
187 if maxu ≤ k
188 count(i5) = count(i5) + 1;
189
190 for i6 = 1:m
191 P = v5;
192 n = count(i6) + 1;
193 P(i6, n) = 6;
194
```

```
195 if i6 > 1 && P(i6 - 1, 1) == 0
196 break
197 end
198
199 v6 = P;
200
201 for j = 1:m
202 w = P(j, :);
203 w(w == 0) = [];
204 u(j) = max(sum(A_0(w, w)));
205 maxu = max(u);
206 end
207
208 if maxu ≤ k
209 count(i6) = count(i6) + 1;
210
211 for i7 = 1:m
212 P = v6;
213 n = count(i7) + 1;
214 P(i7, n) = 7;
215
216 if i7 > 1 && P(i7 - 1, 1) == 0
217 break
218 end
219
220 v7 = P;
221
222 for j = 1:m
223 w = P(j, :);
224 w(w == 0) = [];
225 u(j) = max(sum(A_0(w, w)));
226 maxu = max(u);
227 end
228
229 if maxu ≤ k
230 count(i7) = count(i7) + 1;
231
232 for i8 = 1:m
233 P = v7;
234 n = count(i8) + 1;
235 P(i8, n) = 8;
236 v8 = P;
237
238 if i8 > 1 && P(i8 - 1, 1) == 0
239 break
240 end
241
242 for j = 1:m
243 w = P(j, :);
244 w(w == 0) = [];
```

```
u(j) = max(sum(A_0(w, w)));
maxu = max(u);
end

if maxu ≤ k
count(i8) = count(i8) + 1;

for i9 = 1:m
P = v8;
n = count(i9) + 1;
P(i9, n) = 9;

if i9 > 1 && P(i9 - 1, 1) == 0
break
end

v9 = P;

for j = 1:m
w = P(j, :);
w(w == 0) = [];
u(j) = max(sum(A_0(w, w)));
maxu = max(u);
end

if maxu ≤ k
count(i9) = count(i9) + 1;

for i10 = 1:m
P = v9;
n = count(i10) + 1;
P(i10, n) = 10;
v10 = P;

if i10 > 1 && P(i10 - 1, 1) == 0
break
end

for j = 1:m
w = P(j, :);
w(w == 0) = [];
u(j) = max(sum(A_0(w, w)));
maxu = max(u);
end

if maxu ≤ k
count(i10) = count(i10) + 1;

for i11 = 1:m
P = v10;
```

```
n = count(i11) + 1;
P(i11, n) = 11;
v11 = P;

if i11 > 1 && P(i11 - 1, 1) == 0
break
end

for j = 1:m
w = P(j, :);
w(w == 0) = [];
u(j) = max(sum(A_0(w, w)));
maxu = max(u);
end

if maxu ≤ k
count(i11) = count(i11) + 1;

for i12 = 1:m
P = v11;
n = count(i12) + 1;
P(i12, n) = 12;
v12 = P;

if i12 > 1 && P(i12 - 1, 1) == 0
break
end

for j = 1:m
w = P(j, :);
w(w == 0) = [];
u(j) = max(sum(A_0(w, w)));
maxu = max(u);
end

if maxu ≤ k
count(i12) = count(i12) + 1;

for i13 = 1:m
P = v12;
n = count(i13) + 1;
P(i13, n) = 13;
v13 = P;

if i13 > 1 && P(i13 - 1, 1) == 0
break
end

for j = 1:m
w = P(j, :);
```

```
w(w == 0) = [];
u(j) = max(sum(A_0(w, w)));
maxu = max(u);
end

if maxu ≤ k
count(i13) = count(i13) + 1;

for i14 = 1:m
P = v13;
n = count(i14) + 1;
P(i14, n) = 14;
v14 = P;

if i14 > 1 && P(i14 - 1, 1) == 0
break
end

for j = 1:m
w = P(j, :);
w(w == 0) = [];
u(j) = max(sum(A_0(w, w)));
maxu = max(u);
end

if maxu ≤ k
count(i14) = count(i14) + 1;

for i15 = 1:m
P = v14;
n = count(i15) + 1;
P(i15, n) = 15;
v15 = P;

if i15 > 1 && P(i15 - 1, 1) == 0
break
end

for j = 1:m
w = P(j, :);
w(w == 0) = [];
u(j) = max(sum(A_0(w, w)));
maxu = max(u);
end

if maxu ≤ k
count(i15) = count(i15) + 1;

for i16 = 1:m
P = v15;
```

```
395 n = count(i16) + 1;
396 P(i16, n) = 16;
397 v16 = P;
398
399 for j = 1:m
400 w = P(j, :);
401 w(w == 0) = [];
402 u(j) = max(sum(A_0(w, w)));
403 maxu = max(u);
404 end
405
406 if maxu ≤ k
407 % 一开始为m_min0
408 % 从小到大一一遍历，当的取值存在可行解之后，就将的值赋予mmmm_min，当,的值改变之后就停止循环。
      m_min
409 if m > m_min && m_min ≠ 0
410 All_solution_of_m_min(:, all(All_solution_of_m_min == 0)) = []; %把全的列去掉0
411 xlswrite('43.xlsx', All_solution_of_m_min)
412 % 在最终的前，在表格中输出所有当取最小值的满足要求的数组returnexcelmV
413 m_min % 输出满足条件的最小的m
414 x % 输出当取最小值时满足条件的解的个数m
415 return
416 end
417
418 m_min = m;
419 x = x + 1;
420 % 以下这部分将可行的还原成重排前数组v
421 PT = zeros(m, 16);
422
423 for i = 1:m
424
425 for j = 1:16
426
427 if P(i, j) > 0
428 PT(i, j) = M(P(i, j));
429 end
430
431 end
432
433 end
434
435 % 把算出的解记录下来
436 line1 = (m + 1) * x - m;
437 line2 = (m + 1) * x - 1;
438 All_solution_of_m_min(line1:line2, :) = PT;
439
440 end
441
442 end
443
```

```
444  count(i15) = count(i15) - 1;
445  end
446
447  end
448
449  count(i14) = count(i14) - 1;
450  end
451
452  end
453
454  count(i13) = count(i13) - 1;
455  end
456
457  end
458
459  count(i12) = count(i12) - 1;
460  end
461
462  end
463
464  count(i11) = count(i11) - 1;
465  end
466
467  end
468
469  count(i10) = count(i10) - 1;
470  end
471
472  end
473
474  count(i9) = count(i9) - 1;
475  end
476
477  end
478
479  count(i8) = count(i8) - 1;
480  end
481
482  end
483
484  count(i7) = count(i7) - 1;
485  end
486
487  end
488
489  count(i6) = count(i6) - 1;
490  end
491
492  end
493
```

```
494  count(i5) = count(i5) - 1;
495  end
496
497  end
498
499  count(i4) = count(i4) - 1;
500  end
501
502  end
503
504  count(i3) = count(i3) - 1;
505  end
506
507  end
508
509  count(i2) = count(i2) - 1;
510  end
511
512  end
513
514  count(i1) = count(i1) - 1;
515  end
516
517  end
```