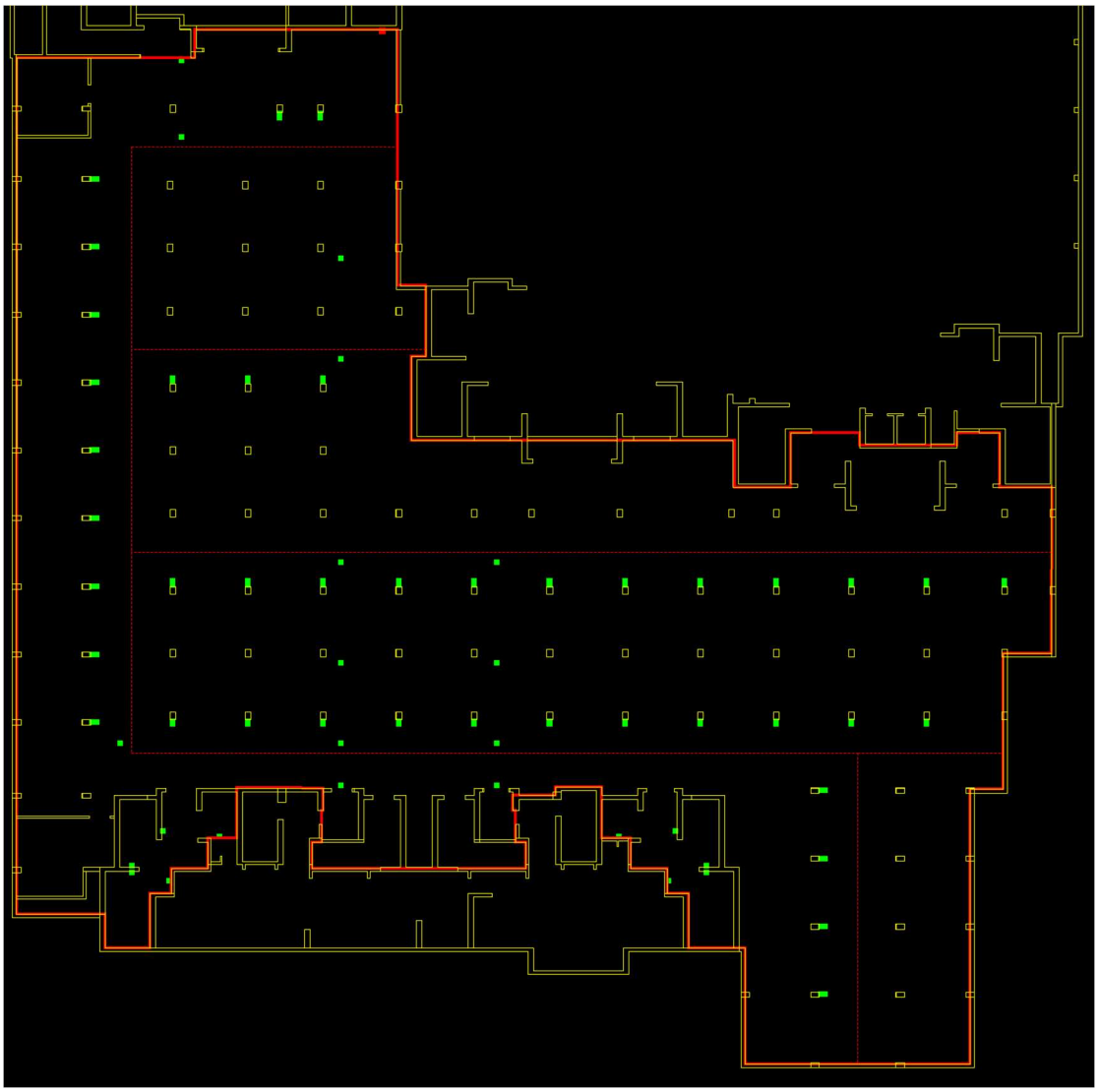
火灾报警项目管线连接部分开发文档

0 输入处理

读入数据后进行预处理，保证：

1）各个设备点位不重复
2）所有设备点位都在防火分区内，障碍物外，并且不会落在边缘线上
3）至少有一个设备
4）至少有一个电源
5）有且只有一个防火分区



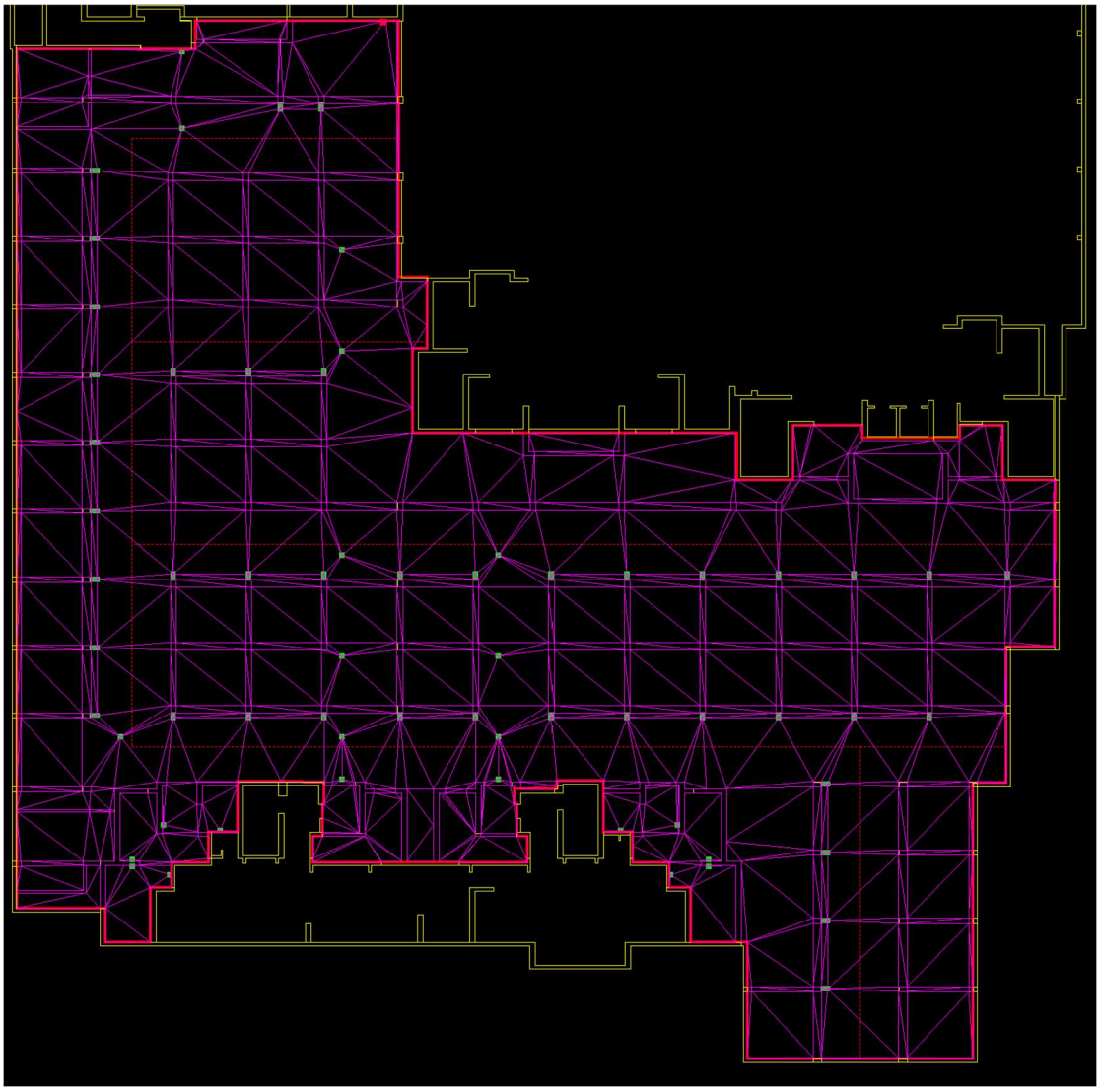
绿色点：设备点位
红色实线：防火分区
黄色实线：障碍物
红色虚线：中心线/车道线

1 分组

主要思路是对所有设备点位建立一个完全图 G，记录每两个设备点位之间绕过障碍可达距离的估算值，根据这个图来进行聚类。使用的聚类方法是，在图上生产最小生成树，然后使用贪心策略来分割最小生成树，每个子树即为一组。

1.1 Denaulay 三角化

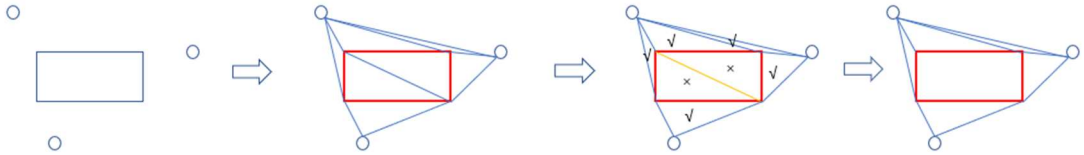
对点集进行三角化，会天然地先将距离较近的点连接。考虑障碍物的影响，先将障碍物信息一起加入三角化，再删去三角网中越过障碍物的边，从而得到在可布线区域的三角网，如图所示。



使用**限制性 Denaulay 三角化**（Constraint Denaulay Triangulation, **CDT**）

1) 将设备点位插入 CDT
2) 将防火分区和障碍物的边，作为**限制边**插入 CDT，限制边一定会出现在 CDT 中
3) **标记非障碍面片：**以设备点位所在的三角面片为起点，向三边邻接的面片拓展并标记为非障碍，若为限制边则不向那一边拓展。拓展结束后，在障碍物内部或防火分区外部的三角面片未被标记，即为障碍面片
4) **删去越障边：**遍历 CDT 中所有边，若边所关联的两个三角面片都为障碍面片，则该

边为越障边，删去，否则保留。



a) 点位与多边形障碍物　b) 生成CDT，红色边为障碍物形成的限制边　c) 标记面片，黄色边两边都没被标记，为越障边，删去

1.2 建立完全图

在 CDT 的顶点中，有设备点，也有障碍点（障碍物和防火分区轮廓顶点）。CDT 可以视为一个图 G_cdt，要从该图中抽取一个**子图 G**，G 是只包含设备点的完全图，包含每个设备之间的距离信息

1) **添加直接可达边：**CDT 中有些点之间没有直接相连，但其实他们连接起来并没有穿过障碍物，将这种边加入 G_cdt 中（包括设备点和障碍点）
2) **加权：**对 G_cdt 中的每条边，根据业务逻辑，比如穿过中心线或穿过房间框线时，进行加权，表示不希望走这条边
3) **抽取子图 G：**在 G_cdt 中，计算所有设备点之间的最短路径。此处采用的方法为，以每个设备点为起点做一次 Dijkstra 算法，取结果中设备点到起点的最短路径。

1.3 生成最小生成树

在图 G 中使用 PRIM 算法生成最小生成树 MST，并保存各边的权值（绕障距离）

1.4 分割

在 MST 中，每一条边若断开，都会将 MST 分为两个子树，再对子树不断分割，最终形成的森林中，每一棵子树包含的节点就被分为一组。采用**贪心策略**，分割一棵树时，先对树中的每一条边进行评估，若断开这条边形成子树 T1 和 T2，考虑：

- **位置关系：**T1 节点形成的最小外界矩阵 bbox1 中包含 T2 的节点数，越少越好；同理 bbox2 包含 T1 节点数越少越好

```
f1 = 3.0 / (2 + one_in_two + two_in_one);
```

- **数量均匀：**T1 与 T2 节点数的差的绝对值越小越好

```
f2 = 1.0 / (abs(size1 - size2) + 1);
```

- **组内凝聚力：**使用树内所有边权的方差来体现凝聚力，T1 和 T2 的该值越小越好

```
f3 = 1000.0 / ((off1 + 1) * (off2 + 1));
```

- **被割边的权值：**越大越好

```
f5 = 5.0 * tree[now].weight / (avg1 + avg2);
```

以一定的权重将各个评价值加起来得到评价函数，取评价最好的一条边进行分割，对子树进行迭代的分割操作，出口为节点数满足分组数量要求。

2 组内确定连接关系

分组时得到的子树所确定的连接关系，由于从最小生成树分割而来，已经可以满足总线路尽量短的要求，但由于其他约束，如一个设备最多连 4 根线（节点的度<=4）、考虑电源位置、最少回头路等，所以需要重新确定连接关系。

使用一棵无根树来表示连接关系的结果，在树中相连的两个节点即为需要连线的两个设备。

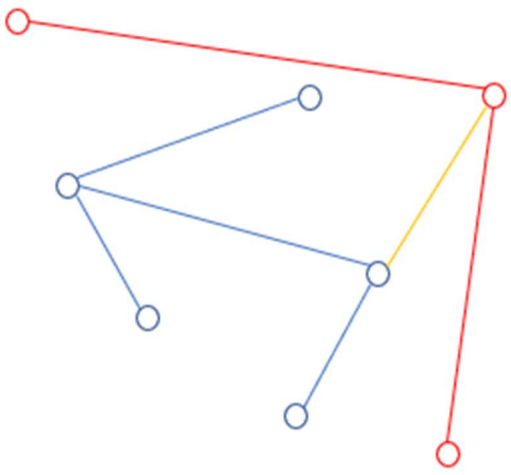
2.1 建立完全图

与分组的建图思路相同，三角化，添加边，加权，抽取子图 G。
区别在于将电源加入图中，在添加直达边后，将电源加入图中，每个电源与所有障碍点和设备点尝试连接，若没有穿过障碍则将该边加入图中。抽取子图时，将电源一并抽出，得到电源和设备的完全图 G。

2.2 遗传算法

遗传算法基本上就是上图的迭代过程，使结果不断逼近评价最高的解。

2.2.1 生成解



本问题中，一棵无根树即一个解。如图，红色点为电源，蓝色点为设备点，过程如下：
1） 将电源点按固定顺序连接（红色线）
2） 选择一个电源和一个设备点连接（黄色线）
3） 以 2 中设备点为起点，将各设备连到树中（蓝色线）
另外，使用交叉操作来生成解，以加快收敛和跳出局部最优。为了方便进行交叉，引入一种编码方式：Prüfer 编码，该编码可以将解表示成一个数字序列，类似于染色体。

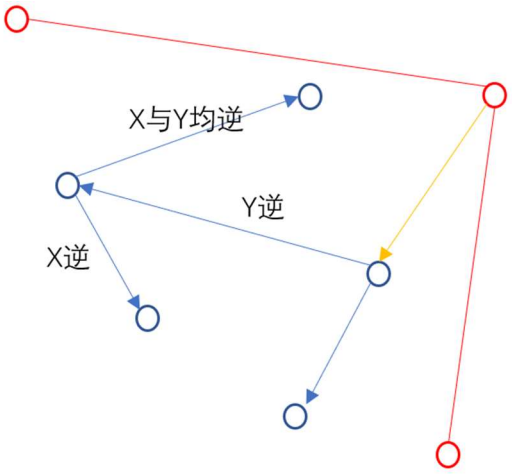
2.2.2 评价解

有效性：

1） 没有节点度大于 4（一个设备最多连 4 条电线）
2） 电源按固定顺序连接，有且只有一个电源连到设备（交叉可能会破坏这个结构）
3） 每条边在完全图 G 中的权值是有效的（连接的两点都是可达的）

优化目标：

1） 边权的合越小越好（总线长）
2） 以电源为树根，每条边都有方向，从父节点指向孩子节点。黄色线的方向视为理想的电流延申方向，分别在 X 和 Y 方向求出所有蓝线向量的投影，与黄色向量相反的量越小越好（逆回路尽量少）



2.2.3 选取解

GlobalMem：使用 5 个全局细胞保存全局评价最好的解
LocalMem：使用 10 个局部细胞保持局部评价最好的解

每次生成解后

1） 去除无效解
2） 与 GlobalMem 和 LocalMem 分别按概率交叉，去掉无效解后加入本次的解集
3） 从解集选取评价最好的 10 个解更新 LocalMem
4） 从解集选取评价最好的 1 个解，加入 GlobalMem 并淘汰 GlobalMem 中最差的解
5） 使用 LocalMem 更新每条边的选取概率（详见蚁群部分）

2.2.4 蚁群算法

参考蚁群算法的思想，将完全图 G 中的权值改为“信息素”（ph），信息素越高的边被选中的概率更高。

1） 根据原本的权值初始化信息素（$常数A + \frac{常数B}{权值W}$，距离越近越高）

2） 生产解时，黄色线与蓝色线都根据 ph 大小选取（选取概率为 $\frac{ph_i}{\sum ph}$）

3） 选取解后，信息素会挥发一部分，然后加上 LocalMem 带来的信息素

3 绕障连线

导航网格+A*寻路+漏斗平滑+曼哈顿平滑

3.1 导航网格

3.2 A*寻路

3.3 漏斗平滑

3.4 曼哈顿平滑

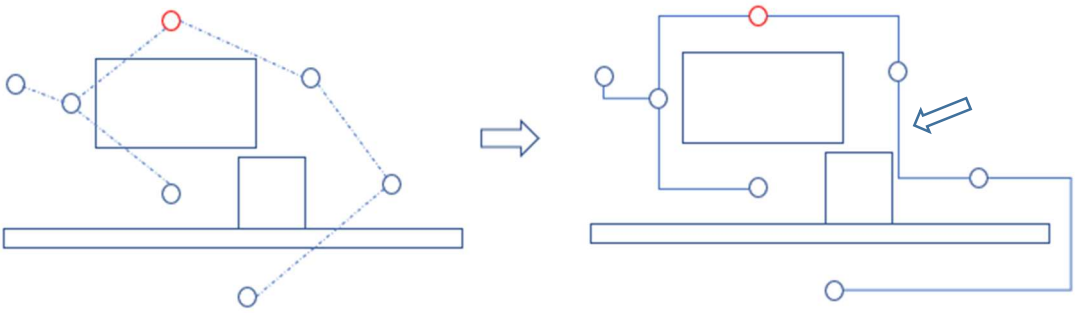
与分组的建图思路相同，使用三角网作为导航网格，生成的网格内部不会出现障碍物

4 组内连线

有了连接关系，现在只需要依次将点连起来即可。这部分先不考虑电源到设备的连线。

4.1 建立连线树

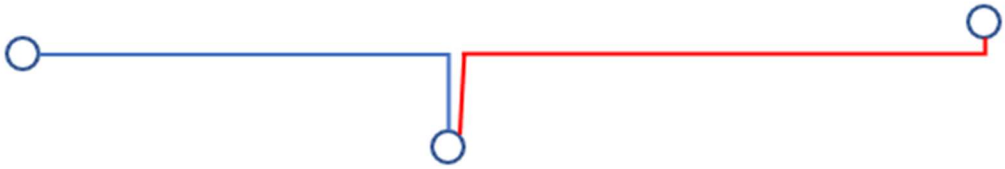
连接关系为无根树，以连接到电源的设备（即组内起点）为根，得到连线树。此时所有连线都是点到点的直线，从起点出发，将所有线变为绕过障碍的曼哈顿连线（即横平竖直）。



1） 两点 X 或 Y 相同，若直接连接不越过障碍，则直接连接，否则使用 A*算法连接
2） X 和 Y 都不同，考察两种曼哈顿连接（先 X 或先 Y），若都越过障碍，则 A*连接；若只有一种不越过障碍，则按那种连接；若都不越过障碍，按照平滑连接的思路，使方向与父节点最后的走线方向相同（如右图箭头所指连线）。

形成的连线树有以下性质:
1） 父节点与孩子节点要么 X 相同，要么 Y 相同
2） 任意非设备节点一定是拐弯处，否则应删去合并
3） 需相连的设备在树中的连接路径就是最终连线

共线平滑：

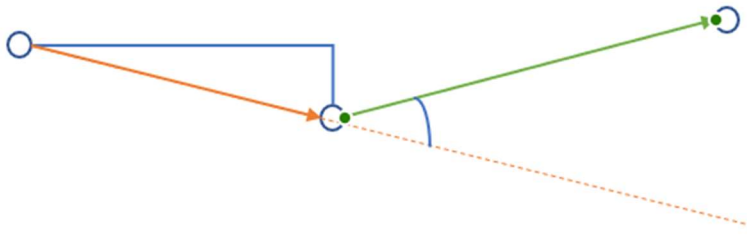


如图，当多个点大致共线时，希望所连线也能够平直

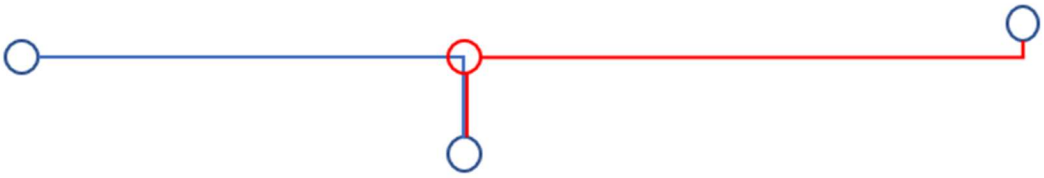
1）当父节点连过来的线最后一段很短时（小于一个阈值），认为可能出现上述情况

2）根据孩子节点位置判断是否为上述情况，当孩子所在方向与走线方向夹角小于 45 度时，

认为需要进行共线平滑处理

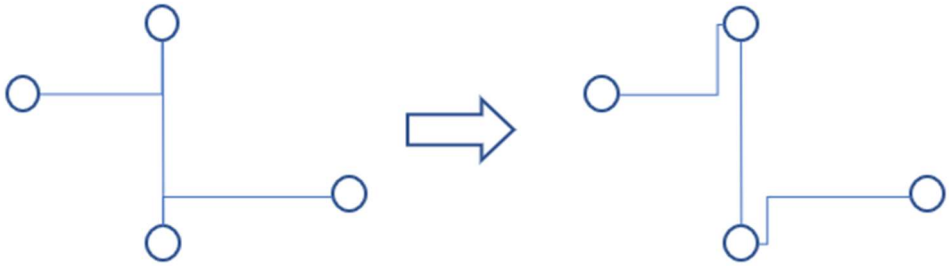


3） 添加红色节点到节点的孩子，走线方向改为父节点到红色点，然后以红色点来进行下一个节点的连线

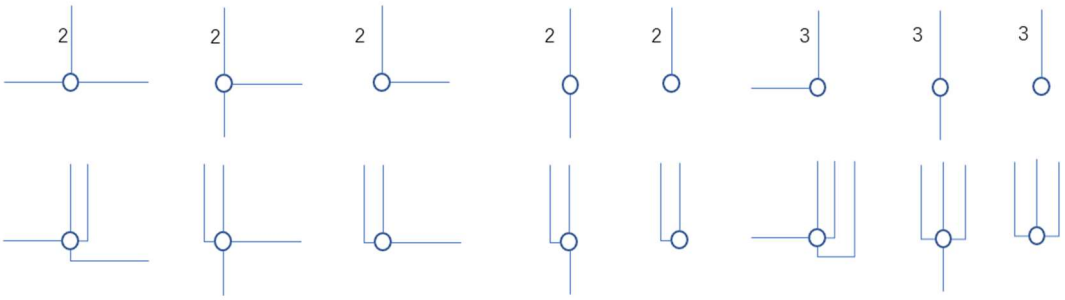


4.2 解重合

在连线树中，节点周围的线可能重合，需要分开，如下图



对每个节点考察所连的线，计算四个方向上的连线数，根据重合情况处理：



5 组间连线（To do）

6.1 电源连线到各组

6.2 将组间线合并为连线树

6.3 梳理连线树