基于用电可靠性的配电网规划

摘要

配电网规划中，在满足需求各项指标的前提下减少建设成本并提高用电可靠性是建设的关键。本文通过建立基于目标优化的单供、双供树状结构配电网供电模型，求解出了能够满足题设要求的理想配电网，并对其相关数据（分叉点、可靠性和建设成本等）进行了研究。

针对于问题一，本文利用最小生成树的优良拓扑特性，首先将已知的电源和负荷节点进行拓扑结构的确认，进而通过结合线路造价成本，微调在最小生成树中的分支节点，以此来构造电网中的分叉点，并将其坐标作为决策变量，构建单目标优化模型进行求解。在确认了网络最终的拓扑结构之后，进一步求解其建造成本及用电可靠性。最终得到了**均值为 86%的用电可靠性**，且方差控制在了 0.31%，得知建立的模型稳定性好。

针对于问题二，本文在尝试了几种传统聚类方法（如 K 均值聚类、层次凝聚聚类等）后，发现这些方法或多或少都存在着些许问题，不能达到构建的预期目标。于是本文提出了一种针对问题目标的启发式二分类算法，结合问题一建立的单目标优化模型进行拓展，最终得到了**两个网络的可靠性均值都在 88%以上**，且数据的方差进一步缩小到了 0.28%以下。结合现实生活也可知，在增加了电源后，通过优化配电网分配，必定可以增加用电可靠性，如此验证了模型的正确性。

对于问题三、四，都是在问题二建立的配电网基础上添加联络线，使其变为双供电网。问题三需要在控制成本上限的情况下，将最低的用电可靠性得到最大的提升；而问题四则需要保证用户的最低用电可靠性在阈值之上，控制建造成本最低。本文基于贪心算法的思想，首先对最低用电可靠性的用户进行联络线连接的考虑，这样必定能在一定限度内提升这些用户的用电可靠性，由此提升整个网络中最低用电可靠性的水平。最后，根据两个问题不同的条件限制，本文分别设立了两个目标函数，结合问题二的两个单供配电网模型进行联络线的选择，以及模型成本、用户可靠性等信息进行求解。由于本文求解模型时朝着最优解的方向进行优化，因此得到的**问题三、四的联络线建造方案一致**。最终，模型经过了的多轮优化后，可以使得**用户的最低用电可靠性达到 91%**。相比最初的 76%，**提升了 15%**。如此验证了贪心算法下求解模型的可靠性。

在文章的最后，本文客观地指出了模型的优缺点。此外，以上问题的所有结果数据均存放于附录中。

关键词：最小生成树 单目标优化模型 启发式二分类算法 贪心算法

一、问题重述

1.1 问题背景

随着城市化的推进，解决供电问题成为了当下城市建设中的一个主要难点。由于配电线路逐渐规模化和集群化，在城市化的进程中需要解决针对城市配电网线路输送紧张，同时兼顾保障供电可靠性等问题[1]。因此，电网在规划阶段需要解决的问题是在满足需求指标的前提下减少投入并增加电网可靠性，工作人员应该积极研究供电可靠性提升的措施与方法，从而更加科学的进行配电网规划设计，进而提升配电网运行的稳定性与安全性[2]。

本题以上述生活需求为背景，讨论了如下的问题：

如果一批用户变压器（下面简称用户）仅由一个电源变电站（下面简称电源）供电，称为**单供**。

这时配电网由电线和开关联接成以电源为根节点的**树状结构图**，使得每个用户所在顶点都在图中有路（电线）联接到电源根节点。一些电力用户一旦发生停电，无论停电时间长短，都会带来较大损失，降低用电满意度。

因此，定义**用户用电可靠性**：指定时间段内不因配电网故障停电或限电的概率。为了提升用户用电可靠性，可在两个电源的单供配电网之间建立联络线，并增设开关和扩充电源可供电功率，形成**双电源供电配电网**（简称双供配电网，如图 1 所示）。

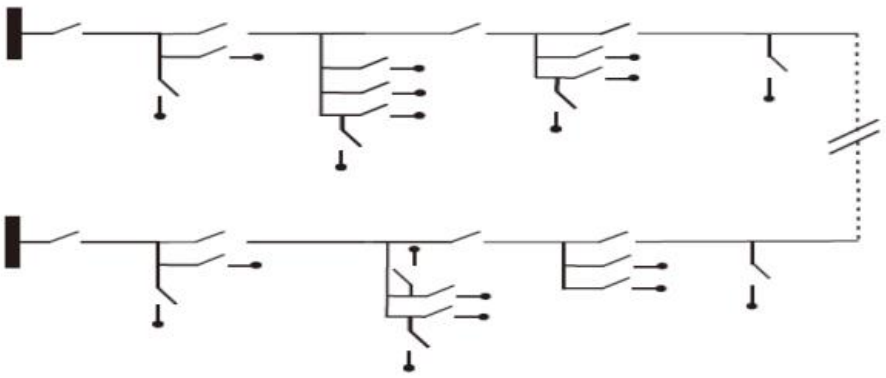

图 1　双电源供电配电网图示

此外，还需要满足以下要求：

开关设置原则：

1.配电网中开关（电源出线后开关除外）的设置必须使得网中某处发生故障时，通过开关隔离故障后，保持供电的用户需求功率之和最大化；

2.在网上任意一点到电源的所有路中，可以通过设置开关状态使得仅有一条是通路；

3.配电网中开关的设置包含但不限于以下情况：每个用户前端有开关，每个分岔点后端的每条支路上有开关，双供配电网的每条联络线上有开关。

配电网设施可靠性单元划分及其可靠性：

忽略电源至它的后端第一个开关部分，忽略用户至它的第一个前端开关部分，配电网设备可靠性（故障）单元由电源、用户、开关，以及仅含两个开关之间的路（下面称为故障单元路）构成。每个单元设备在指定时间段内正常运行的概率称为单元设备可靠性，它等于 1 减去该单元设备的故障率。

双供配电网用户供电调度原则：

1.满足一个用户全部需求功率，否则断开该用户；

2.首先满足各自单供配电网内用户的需求；双供电源多余功率的分配优先提高全配电网供电功率总和，然后提升全配电网最低的用电可靠性。

1.2 需要求解的问题

对题目进行分析总结，得到各个问需要解决的问题如下：

问题一：已知一个电源和一批用户的平面坐标、每个用户用电功率需求、每个设备单元建造费用（数据格式见附录）。设计建造费用最低的单供配电网供电所有用户，给出树状配电网的分叉点坐标，并计算该配电网中每个用户的用电可靠性。

问题二：已知两个电源和一批用户的平面坐标、每个用户用电功率需求、每个设备单元建造费用（数据格式见附录）。设计建造费用最低的两个单供配电网，使得每个用户都被供电。给出树状配电网的分叉点坐标，并计算该配电网中每个用户的用电可靠性。

问题三：在第 2 题结果的基础上，通过建立两个单供配电网之间的联络线，增设开关，并扩充电源可供电功率，形成双供配电网，以提高用户的用电可靠性。假设两个电源各自能扩充可供电功率 50%，建造双供配电网总花费上限为 X，求

使得双供配电网中最低的用电可靠性达到最大的联络线和开关设计。画出联络线拓扑简略图，并计算双供配电网中每个用户的用电可靠性。

问题四：在第 2 题结果的基础上，通过建立两个单供配电网之间的联络线，增设开关，并扩充电源可供电功率形成双供配电网，以提高用户的用电可靠性。假设两个电源各自能扩充可供电功率 50%，设计建造总费用最低的双供配电网，使得双供配电网中每个用户的用电可靠性不低于 Y%。画出联络线拓扑简略图，并计算双供配电网中每个用户的用电可靠性。

二、问题分析

2.1 问题一的分析

问题一要求，针对一批已知的用户及其需求和电源及设备的建造费用，设计一个费用最低的单供配电网络，设计要素包括网络的部分节点（因建模需要，将树状输电网中除了电源（根节点）和用户（叶节点）的节点称为“分叉点”）坐标以及各个节点之间的拓扑连接关系。

我们发现，同时处理这两个要素比较困难，故采取先确定连接电源与用户之间的拓扑结构，再考虑优化分叉点的坐标位置。

设计拓扑结构时，我们发现最小生成树有良好的拓扑特性，舍去冗余的连线，并且最小化了网络线路的长度总和。因此，本文在**最小生成树算法**的基础上确定网络的拓扑结构，之后以分叉点坐标为决策变量，以建造费用为目标函数，建立**基于单目标优化的单供配电网模型**，并且利用 $MATLAB$ 的 $fminunc()$ 函数求解最值，得到了一个较优的网络设计。

图 2　求解问题一的思路流程图

2.2 问题二的分析

问题二要求，设计建造费用最低的**两个**单供配电网，与问题一唯一不同在于提供了两个电源，需要建立两个相互独立的单供配电网。

我们基于问题一进行分析，确定每个负荷的供电电源，即**对负荷集合进行二分类**后，可对两个节点集合各自建立问题一中的模型进行求解。

由于假设负荷节点的分布是随机、均匀的，通过传统的聚类算法难以求解最优分类。因此，我们提出针对问题目标，即使建造费用最低的**启发式二分类方法**，结合问题一的模型完成设计要求。

2.3 问题三的分析

问题三要求，在问题二求解得到的两个单供配电网的基础上，增加联络线，并在限制了建造的双供配电网中的总花费上限的前提下，求得使**双供配电网中最低用电可靠性达到最大**的联络线和开关设计。

由题意得知，扩展电源会优先提高最低可靠性的负荷的用电可靠性。因此，我们将采用**贪心算法**来构造联络线，在成本允许的范围内，优先考虑对可靠性最低的负荷搭建联络线，以此来最大程度上地提升可靠性最低负荷的用电可靠性。

模型中电源遵从**优先供给原则**，首先考虑使用本供电网中电源，如果本供电网电源无法连接，才会考虑使用另一个网络中的电源。

故计算可靠性时，可以使用**条件概率**的方式，将可靠性表述为：负荷自身所在网络的可靠性与负荷自身所在对自身网络电源的故障率与对另一个电源可靠性的乘积的和。

同样的，遵循该原则，我们使用**期望**的方式计算扩展后网络的功率。即将每一个电源所负担的功率写为自身网络中负荷乘上连接本网络电源的概率与另一个网络中负荷乘上连接自身网络电源的概率的和。这样就可以相对准确地估计双供配电网中的功率。

2.4 问题四的分析

问题四与问题三类似，但不同的地方在于，问题四限制了建造的双供配电网中的每个用户用电可靠性下限，需要我们求解建造总花费最低的新电网。

我们同样采用**贪心算法**的策略，在满足用户可靠性的最低阈值下，逐步调整网络构建的成本，以使其达到最低，从而完成对**双供配电网中建设成本达到最小**的联络线和开关设计，并最大程度上保证用户的用电可靠性。

三、模型假设

为了适当地对模型进行合理简化，本文做出如下假设：

1、负荷均匀、随机地分布于一定区域，并且电源位于区域的中心；

2、与电源直接相连的线路使用主线；

3、双供电源中，联络点一定位于负荷到电源的路上；

4、供电线路为单向供电，即：不考虑在双供电路中，原先由电源 1 所供电的负荷，在由于线路故障问题后通过联络线的导通及连线上开关的开闭，使得由电源 2 出现与原先供电方向相反的情况，供电方向不会随着更改电源而改变（原树中电流不会通过子节点流向父节点）。

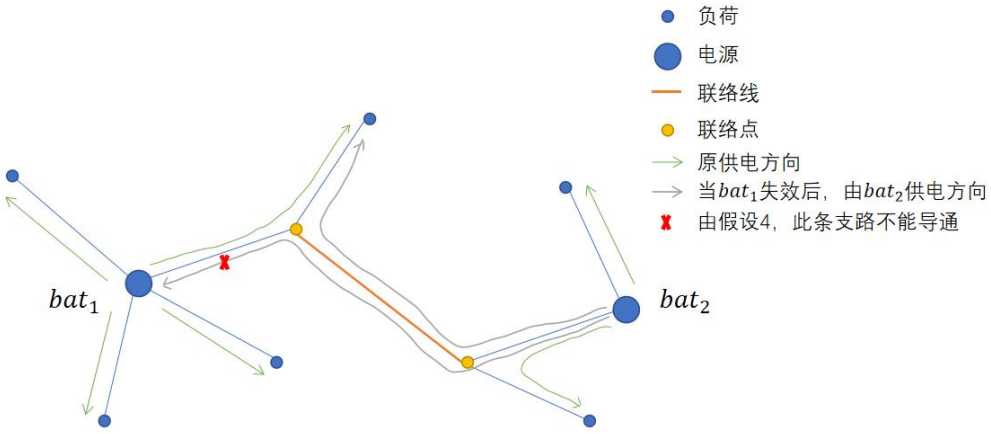

图 3　模型假设 4 的示意图

四、符号说明

序号	符号	意义
1	bat	树状配电网的电源
2	K_i	树状配电网中电源的位置
3	G	树状配电网的邻接矩阵
4	A_i	树状配电网中第 i 个负荷节点
5	B_i	树状配电网中的第 i 个分叉点
6	$cost_{circuit}$	线路单价向量
7	$cost_{switch}$	开关单价向量
8	$length$	线路长度向量
9	num	开关数量向量
10	$load[i]$	树状配电网中节点 i 所在子树的负荷数
11	fa_i	树状配电网中节点 i 的父亲节点
12	$type(ifa_i)$	配电网中，线路 (i, fa_i) 的电线线路类型
13	$length(k)$	配电网中，线路类型为 k 的线路长度
14	$dis(i,j)$	配电网中，节点 i 到节点 j 的距离
15	$r[i]$	配电网中，节点 i 的用电可靠性
16	$C[i]$	用户用电的状态变量
17	P_i	双供电网中，网络 i 的配电功率
18	p_{k_i}	双供电网中，联络线位于 i 电源部分的端点（联络点）
19	N_i	双供配电网中，由 bat_i 供电部分的网络部分
20	$V[N_i]$	双供电网中，位于 i 电源部分的负荷集合

五、模型的建立与求解

5.1 问题一：设计建造费用最低的单供配电网络

针对于问题一，需要首先确定输电网络的拓扑结构，从而确定其线路走向，进而最终得到建造费用最低的单供配电网络。

分析题目后，有两种拓扑结构亟待研究：一个是类似于计算机网络中的星型拓扑，一个是结合现实生活中的辐射式结构（类似于题面的示例图）。前者，电源位于整个网络的中心位置，所有支路将从电源位置辐射出去，一个用户即对应一条支路，但考虑到在现实生活中，如此方案下布线复杂，且可能需要考虑到与道路交通建设的重叠部分，布线不能太过于复杂，这样不仅在线路修建甚至未来检修时可能需要破坏一定程度的交通状况（例如布线斜跨了一个交通路口），还达不到操控简易的效果，因此我们选择采用后者，即辐射式的布线方式。这样，不仅能使从电源出发的支路能管控多个用户，也能在与道路交通建设及日后的维修之中提供一定程度的便捷性。

此外，我们根据“电源——分叉点（类似于现实生活中的变压器，能控制多个用户或支路的汇点）——用户”三个层次，对应到树状结构中的三种状态：“根节点——分支节点——叶节点”，如此一来，不仅方便后续图的建立和操作，还能层次分明、逻辑清晰，且便于最终选择相应的线路进行建设，以求得最小成本耗费下的分叉点坐标和用户用电可靠性等目标结果。

5.1.1 模型的建立——单供配电网模型

（1）最小生成树的优良拓扑特性

首先，设计建造费用最低的配电线路应尽量减少多余的连线，使得电源与每个负荷之间的通路唯一，这要求配电线路具有树形结构。如此，不仅满足了题目中的开关设置原则 2，并且也能寻找出成本最低的拓扑方案。

其次，由假设 1“负荷均匀、随机地分布于一定区域，并且电源位于区域的中心”，负荷配电网络应是以电源节点为中心节点的辐射式网络。

而最小生成树具有以电源为根节点的辐射式树形结构，同时它具有最小的线路总和，在不同线路造价比（如主线/支线 A 为：325.7/188.6=1.7）较小的情况下具有较低的线路造价。

（2）基于最小生成树的配电网拓扑设计

首先，对电源和负荷节点求最小生成树。此时负荷节点可能是树的分支节点，根据生活常识，用户往往是处于配电线路的末端，作为树状网络的叶子节点。因此，我们通过构造分叉点加入网络，微整网络拓扑，得到基于最小生成树的配电网拓扑设计。

分叉点的构造：

1、对最小生成树中原先作为分支节点的负荷节点 A_i，在其附近构造一个分叉点 A_i'；

2、删除节点 A_i 的临界线路：

$$\{A_iB_j \mid j = 1, 2, \ldots\} \qquad (1)$$

3、增加线路 $A_i A_i'$ 以及分叉点 A_i' 的邻接线路：

$$\{A_i'B_j \mid j = 1, 2, \ldots\} \qquad (2)$$

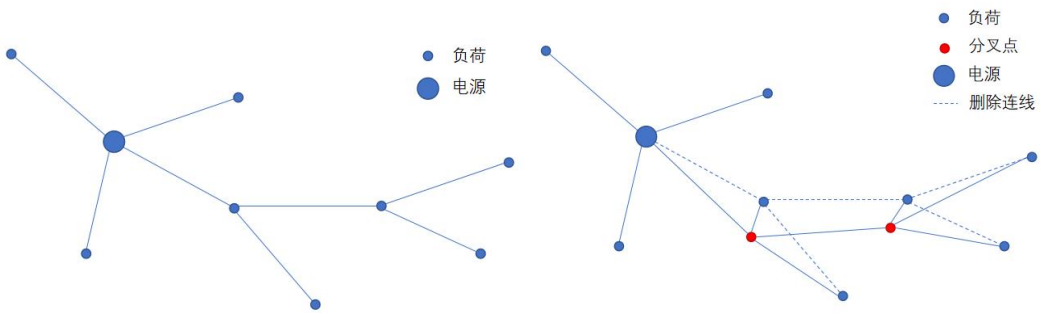

图 4　最小生成树示意图[左]　　分叉点的构造示意图[右]

最终，我们得到拓扑结构和原先最小生成树相似的树状配电网设计，其中电源作为根节点、分叉点作为分支节点、负荷作为叶子节点。

设树状配电网的邻接矩阵为 G，电源为 bat，分叉点集合为

$$B = \{b_i \mid i = 1, 2, \ldots, n_b\} \qquad (3)$$

叶子集合为

$$L = \{l_i \mid i = 1, 2, \ldots, n_l\} \qquad (4)$$

（3）开关的铺设以及线路的选型

在拓扑结构确定之后，我们考虑开关的铺设以及线路的选型。

根据开关设置原则 3“每个用户前端有开关，每个分叉点后端的每条支路上有开关”，基于成本考虑，满足最低的设计要求，即每条线路增设一个开关。

关于线型的选择，由附录第 4 点，主线可承载 2 倍的电源额定功率，支线 A 可承载至多两个负荷，支线 B 可以承载 3 个及以上的负荷。由于主线和支线 B 在该问题背景下无法区分，所以，根据经验，我们假设与电源直接相连的线路使用主线，其他情况使用支线 A、B。

设节点 i 的所在子树的负荷数为 $load[i]$，由 G 从 bat 出发的深度优先搜索可以求得每个节点 i 的孩子节点集合，将其设为

$$son_i = \{s_{ij} \mid j = 1,2,\dots,n_i\} \qquad (5)$$

设其父亲节点为 fa_i，由递推公式：

$$load[i] = \begin{cases} 1, & i \in L \\ \sum_{s_{ij} \in son_i} load[s_{ij}], & i \in B \end{cases} \qquad (6)$$

可以求得 $load[i]$，此时可以确定所有线路的选型。

对于节点 i，我们考察线路 (i, fa_i)，即 ifa_i，设其线型为 $type(ifa_i)$，有：

$$type(ifa_i) = \begin{cases} 1（主线）, & fa_i = bat \\ 2（支线\ A）, & load[i] \le 2 \\ 3（支线\ B）, & load[i] > 2 \end{cases} \qquad (7)$$

（4）分叉点的选址以及线路建造费用的计算

设线路单价向量为 $cost_{circuit}$，开关单价向量为 $cost_{switch}$，线路长度向量为 $length$，开关数量向量为 num。根据题目提供的数据得：

$$cost_{circuit} = (325.7 \quad 188.6 \quad 239.4) \qquad (8)$$

$$cost_{switch} = (2.6 \quad 58.6) \qquad (9)$$

为了确定 $length$，我们引入决策变量——分叉点坐标，设为 $\{(x_i, y_i) \mid i \in B\}$，同时，根据假设 1“负荷均匀、随机地分布于一定区域，并且电源位于区域中心”构造测试数据集。

设负荷坐标为$\{(x_i, y_i) \mid i \in L\}$，电源坐标为$\{(x_i, y_i) \mid i \in bat\}$。

使用欧氏距离公式，可以计算i, j两节点之间的距离：

$$dis(i,j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \qquad (10)$$

由此可以确定：

$$length(k) = \sum_{G(i,j)=1 \text{ 且 } type(i,j)=k} dis(i,j),\ k = 1,2,3 \qquad (11)$$

而开关数量向量的确定，容易知道每个负荷前有一个负荷前开关，每个分叉点（包括电源）后有一个主线开关，推得$num = (n_l, n_b)$。

因此，可得线路建造费用总和为：

$$cost_{sum} = cost_{circuit} * length + cost_{switch} * num \qquad (12)$$

（5）用户可靠性的计算

设节点i的用电可靠性为$r[i]$，已知其父亲节点为fa_i，有如下式子：

$$r[i] = r|fa_i| * (1 - 0.5\%) * (1 - 0.2\%) * (1 - 0.002 * dis(i, fa_i)) \qquad (13)$$

类似于$load[i]$，上式可通过深度优先搜索的过程中递推计算。最终，可以得到：$\{r[i] \mid i \in L\}$为用户用电可靠性。

（6）基于单目标优化的单供配电网模型

问题一要求建造费用最低的单供配电网供电所有用户，基于最小生成树算法，我们设计了以电源为根节点的辐射式树状结构，一方面，满足了供电给所有用户的需求，另一方面，这个优良的拓扑结构也为了减少成本上的支出打下了基础。

之后，我们通过引入决策变量——分叉点的坐标$\{(x_i, y_i) \mid i \in B\}$，量化描述线路造价，来建立基于单目标优化的单供配电网模型。

目标函数：最小化线路建造费用，即$min(cost_{sum})$，由于开关部分的花费在拓扑结构确定后也随即确定，所以式子进一步可简化为：

$$min(cost_{circuit} * length) \qquad (14)$$

约束条件：

1、配电网是以电源为根节点的树状结构；

2、开关设置原则以及安培限流；

3、线路安培限流；

以上约束通过构造性算法得以满足，因此最后得到的实际上是无约束模型。

综上所述，建立单目标优化模型：

$$min(cost_{circuit} * length)$$

$$\begin{cases} load[i] = \begin{cases} 1, & i \in L \\ \sum_{s_{ij} \in son_i} load[s_{ij}], & i \in B \end{cases} \\ type(if a_i) = \begin{cases} 1（主线）, & fa_i = bat \\ 2（支线\ A）, & load[i] \leq 2 \\ 3（支线\ B）, & load[i] > 2 \end{cases} \\ dis(i,j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \\ length(k) = \sum_{G(i,j)=1 \text{ 且 } type(i,j)=k} dis(i,j), k = 1,2,3 \\ r[i] = r|fa_i| * (1 - 0.5\%) * (1 - 0.2\%) * (1 - 0.002 * dis(i, fa_i)) \end{cases} \tag{15}$$

5.1.2 模型的求解

问题一建立的基于单目标优化的单供配电网模型求解的算法步骤如下：

Step1：测试数据集生成

借助随机数函数生成测试数据集，其中负荷节点均匀分布于 100*100 的区域内，电源节点位于区域的中心位置，电源默认为 1 号节点。生成的负荷数据存储于'test_data.xlsx'文件中。

序号	x 坐标（km）	y 坐标（km）	需求量（兆瓦）
1	60.49906419	38.72454315	9.7540405
2	14.21871593	2.513498571	27.84982189
3	42.11122538	18.41002894	54.68815192
...	...	...	...
50	84.3213338	92.23319978	11.89976816

表 1　部分负荷数据集示意（详见附录 8.2.1）

Step2：利用 Kruskal 算法求最小生成树

a）读取'test_data.xlsx'文件，将二维坐标点之间的距离转化为带权边集 E，将边集 E 中的边按边权从小到大排序；

b）初始化并查集，依次遍历边集 E 中的边 e，判断 e 的端点是否连通，若不连通，则将 e 在最小生成树的邻接矩阵 G_{mst} 中标记，并且并查集中合并两端点，否则跳过；

c）返回 G_{mst}。

Step3：构造分叉点

a）预处理 G_{mst} 中所有节点的度数 deg；

b）对 G_{mst}，从 1 号电源节点出发进行深度优先搜索（后面简称 dfs），对于 $deg(i) > 1$ 的节点构造其邻接的分叉点（构造过程详见模型的建立）；

c）返回包含构造分叉点的配电网拓扑的邻接矩阵 G。

Step4：确定线路选型

a）对 G 进行 dfs，根据公式（6）计算 $load$ 数组；

b）根据 dfs 得到 $load$ 数组以及公式（7）确定线路选型，并在 G 中赋予相应权重 $type$。

Step5：利用 MATLAB 求解器求无约束函数最值，计算最低建造费用

a）预处理邻接矩阵 G 获取对应边集 E_G；

b）对决策变量分叉点的坐标进行编码为：

$$X = [x_1, x_2, \ldots, x_n]$$

根据公式（8）-（12）定义目标函数，其中遍历边集 E_G 以达到遍历图的效果，优化算法效率；

c）通过优化函数 $fminunc()$ 求解最值 $fval_{best}$ 以及最值点 x_{best}，$fval_{best}$ 为模型求解的最低建造费用；

d）对 x_{best} 解编码获得分叉点的坐标，写入文件'bifurcation_coordinates_t1.xlsx'中。

Step6：代入最值点，求用户用电可靠性

将 step5 中获取分叉点的坐标与负荷电源坐标整合为配电网拓扑 G 的二维坐标信息 p_{xy}，对 G 进行 dfs 的过程中根据公式（13）计算用户用电可靠性为 r，写入文件'power_reliability_t1.xlsx'。

Step7：通过 GraphPlot 的相关 API 将最优配电网可视化

5.1.3 结果分析

基于单目标优化的单供配电网模型求解后，得到的配电网如下图所示：

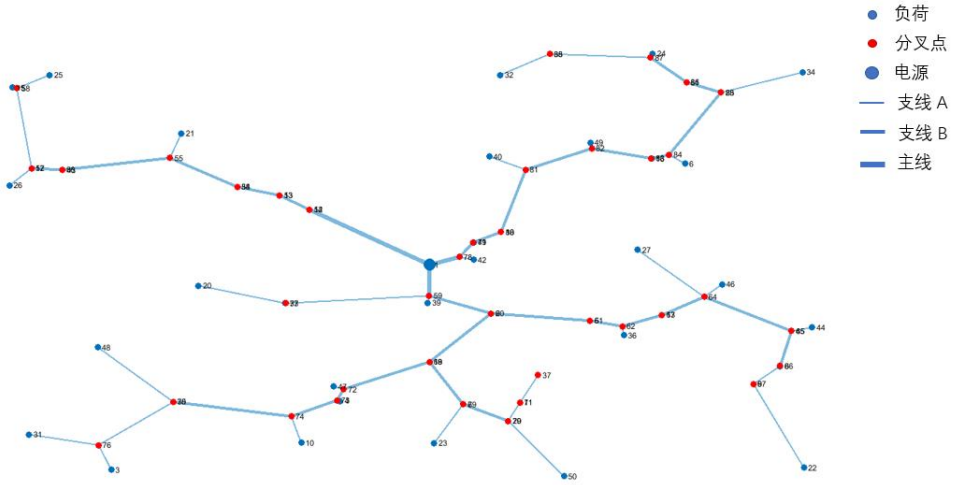

图 5 基于单目标优化的单供配电网模型结果示意图

并且可以得到，树状配电网的分叉点坐标（部分）如下表：

节点编号	x 坐标（km）	y 坐标（km）
52	38.40748071	62.72762528
53	34.74809205	66.06018075
54	29.55325777	68.0146831
...	...	...
88	67.67015597	98.83598478

表 2 问题一建立的分叉点坐标部分展示（详见附录 8.2.2）

由此计算出的用户用电可靠性（部分）如下表所示：

节点编号	x 坐标（km）	y 坐标（km）	用户可靠性（*100%）
2	60.49906419	38.72454315	0.944154412
3	14.21871593	2.513498571	0.786398791
4	42.11122538	18.41002894	0.872546006
...	...	...	...
51	84.3213338	92.23319978	0.810089476

表 3 问题一求解的用户用电可靠性部分展示（详见附录 8.2.3）

根据求解结果，可以建立出关于用电可靠性的分布直方图，如下图所展示：

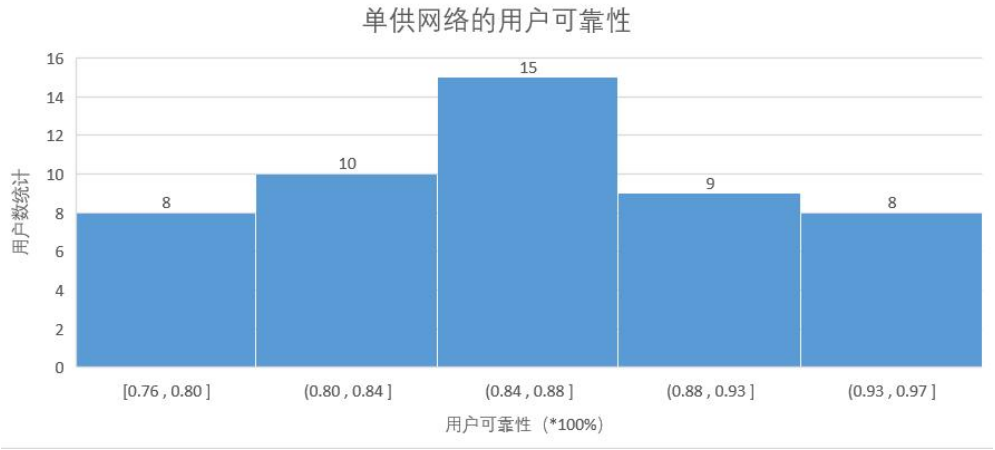

图 6　问题一求解的用电可靠性频率分布直方图

通过直方图，我们可以更加清晰地看出：用电可靠性以 86%为均值，较为集中地分布在 80%~90%之间。经统计，我们发现有 8 对数据低于 80%，有 13 对数据高于 90%，分布较为均匀，且可靠性高的占多数。

进一步，我们求得数据的方差为 0.31%，也就是万分之三十一，数据浮动性较小。可以说明，该布线方案可以较好地保障大多数用户的供电稳定性和可靠性，验证了本文建立的基于单目标优化的单供配电网模型及其求解结果的准确性。

5.2 问题二：设计建造费用最低的两个单供配电网络

根据问题一奠定的基础，本问中再增设一个电源后，需要讨论求解新增电源后配电网的分叉点坐标位置确认，以及用电可靠性的大小。问题二中重要的是如何在双电源的情况下分配用户的供电区域，进而讨论其分叉点坐标以及用电可靠性。

5.2.1 模型的建立

（1）传统聚类方法的尝试

将问题转化为二分类问题后，首先想到应用传统的聚类/分类方法，如 k 均值聚类、层次凝聚聚类法以及均值漂移聚类等，但这些方法多少存在以下问题：

1、不适合发现非凸形状的簇，我们假设节点是随机均匀分布的，并不是一个个具有凸形状的簇；

2、无法确定最终分类的数量，根据问题需要要求将负荷点集合二分类，像均值漂移无法确定分类数量；

3、分类的适应度函数和问题目标函数关联较小，我们的目标是最小化线路建造费用，而传统聚类方法往往是基于距离、密度进行的。

（2）针对问题目标的启发式二分类算法

在尝试传统方法后，我们针对问题优化目标提出一种启发式二分类算法：

1、设 K_i 代表电源的位置，并作两个电源之间的连线

$$l = K_1K_2 \qquad (16)$$

设 l 的中点为 $O(x_O, y_O)$，令

$$k = \frac{y_{K_1} - y_{K_2}}{x_{K_1} - x_{K_2}} \qquad (17)$$

用直线方程一般表达式描述 l 可得

$$l: f(x,y) = y - y_O - k(x - x_O) = 0 \qquad (18)$$

2、作 l_1、l_2 垂直于 l，且关于点 O 对称，设 l_1、l_2 和 l 的交点分别为 T_1、T_2 令

$$t = T_1O = T_2O \qquad (19)$$

根据几何关系可得：

$$\begin{cases} x_{T_{1,2}} = x_O \pm \dfrac{t}{\sqrt{k^2+1}} \\ y_{T_{1,2}} = y_O \pm \dfrac{kt}{\sqrt{k^2+1}} \end{cases} \qquad (20)$$

同理，可以通过直线方程的一般表达式描述 l_1、l_2：

$$\begin{cases} l_1: f_1(x,y) = y - y_{T_1} + \dfrac{1}{k}(x - x_{T_1}) = 0 \\ l_2: f_2(x,y) = y - y_{T_2} + \dfrac{1}{k}(x - x_{T_2}) = 0 \end{cases} \qquad (21)$$

l_1、l_2 将平面点集划分为 M_1、M_2、M_3 三个部分，其中：

$$\begin{cases} M_1 = \{(x_i, y_i) \mid f_1(x_i, y_i) > 0\} \\ M_2 = \{(x_i, y_i) \mid f_2(x_i, y_i) < 0\} \\ M_3 = \{(x_i, y_i) \mid f_1(x_i, y_i) * f_2(x_i, y_i) < 0\} \end{cases} \quad (22)$$

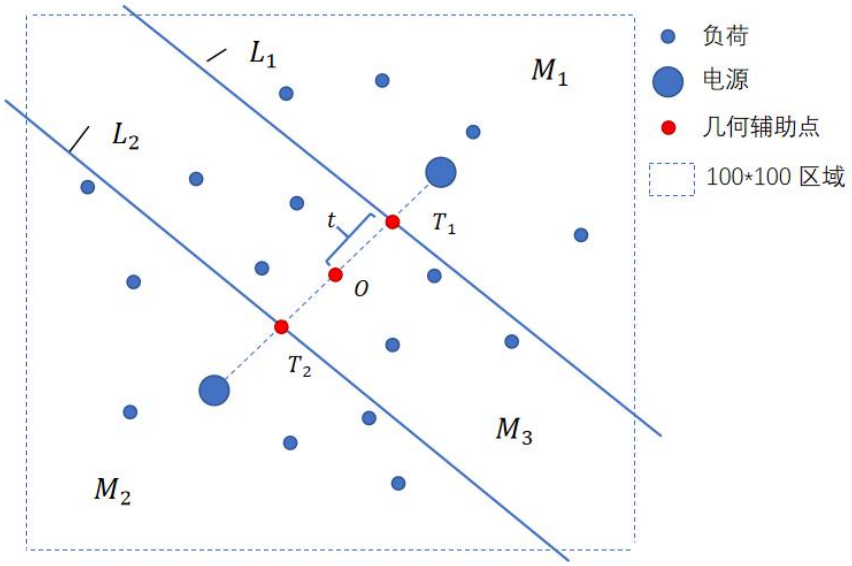

图 7　二分类算法构造示意图

3、选择一个较小的值 α，平移 l_1、l_2，即通过改变 t，使得 $|M_3| = \alpha$，这里可以利用二分法确定 t 的取值；

4、将 M_1 和 K_1 分类至集合 S_1 中，M_2 和 K_2 分类至集合 S_2 中，用一个 0-1 状态向量 s 描述 M_3 中点的分类情况，其中‘0’代表属于集合 S_1，‘1’代表属于集合 S_2。

设 M_3 中属于 S_1 的点集为 s_1，属于 S_2 的点集为 s_2，由此可以简洁地描述分类情况：

$$S_i = M_i \cup S_i, \quad i = 1, 2 \quad (23)$$

5、枚举状态向量 s；

6、对集合 S_1、S_2 各自建立问题一中的单供配电网模型，各自求得其对应的最低耗费 $min_{S1}(cost_{sum})$ 以及 $min_{S2}(cost_{sum})$。

设：

$$cost = min_{S1}(cost_{sum}) + (min_{S2}(cost_{sum}) \quad (24)$$

令 $cost_{best}$ 为最优的 $cost$，S_{best} 为最优 $cost$ 所对应的 S_1；

7、更新 $cost_{best}$ 以及 S_{best}；

8、若枚举还未结束，回到算法的步骤 5，否则结束。

（3）单供配电网模型的拓展

基于前面提出的启发式二分类算法，结合问题一的单供配电网模型，建立得到问题二的模型：

$$min(min_{S1}(cost_{sum}) + (min_{S2}(cost_{sum}))$$

$$\begin{cases} k = \dfrac{y_{K_1} - y_{K_2}}{x_{K_1} - x_{K_2}} \\ \begin{cases} l: f(x,y) = y - y_O - k(x - x_O) = 0 \\ l_1: f_1(x,y) = y - y_{T_1} + \dfrac{1}{k}(x - x_{T_1}) = 0 \\ l_2: f_2(x,y) = y - y_{T_2} + \dfrac{1}{k}(x - x_{T_2}) = 0 \end{cases} \\ \begin{cases} M_1 = \{(x_i, y_i) \mid f_1(x_i, y_i) > 0\} \\ M_2 = \{(x_i, y_i) \mid f_2(x_i, y_i) < 0\} \\ M_3 = \{(x_i, y_i) \mid f_1(x_i, y_i) * f_2(x_i, y_i) < 0\} \end{cases} \\ S_i = M_i \cup S_i, \quad i = 1, 2 \\ load[i] = \begin{cases} 1, & i \in L \\ \sum_{s_{ij} \in son_i} load[s_{ij}], & i \in B \end{cases} \\ type(ifa_i) = \begin{cases} 1（主线）, & fa_i = bat \\ 2（支线 A）, & load[i] \le 2 \\ 3（支线 B）, & load[i] > 2 \end{cases} \\ dis(i,j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \\ length(k) = \sum_{G(i,j)=1 \text{ 且 } type(i,j)=k} dis(i,j), k = 1, 2, 3 \\ r[i] = r|fa_i| * (1 - 0.5\%) * (1 - 0.2\%) * (1 - 0.002 * dis(i, fa_i)) \end{cases} \tag{25}$$

5.2.2 模型的求解

问题二提出的启发式二分类算法的求解步骤如下：

Step1：测试数据集生成

负荷数据同问题一，增设一电源后，两个电源 K_1、K_2 位于各自区域的中心。

节点编号	x 坐标（km）	y 坐标（km）
1	72.7884	75.8243
35	27.862	27.2309

表 4　问题二建立的电源坐标（详见附录 8.2.7）

Step2：实现启发式二分类算法

a）根据公式（16）-（22）构建基本几何关系，得到平面点集 $\{P_i \mid i = 1,2,3\}$；

b）选取参数 α 的值为 10，通过**二分法**确定 t 的值，具体过程：

①设左右边界取值为：$L = 0$，$R = 1e2$；

②设 $mid = (L + R)/2$，通过 $check$ 函数判断，当 $t = mid$ 时，$|P_3|$ 与 α 的关系：

$$\begin{cases} \text{若 } |P_3| > \alpha\text{，则调整 } R = mid; \\ \text{若 } |P_3| < \alpha\text{，则调整 } L = mid; \\ \text{若 } |P_3| = \alpha\text{，则 } t = mid \text{ 为所求;} \end{cases}$$

③若 $|P_3| = \alpha$，则算法结束；否则回到②。

c）设置状态向量描述 s，枚举 s 值，通过公式（23）确定集合 S_1、S_2；

d）将问题一中的单供配电网模型封装为函数 SSDN_model(全称 Single Supply and Distribution Network model)，将 S_1、S_2 分别代入 SSDN_model 进行求解最值，更新最低耗费 $cost_{best}$ 以及相关的状态信息 $info_{best}$（分叉点坐标、用户用电可靠性等等），直至完成 s 值的枚举；

e）将最优状态 $info_{best}$ 写入文件'result_t2.xlsx'。

Step3：通过 GraphPlot 的相关 API 将拥有两个电源的最优配电网可视化

5.2.3 结果分析

基于单目标优化的单供配电网模型求解后，得到的配电网如下图所示：

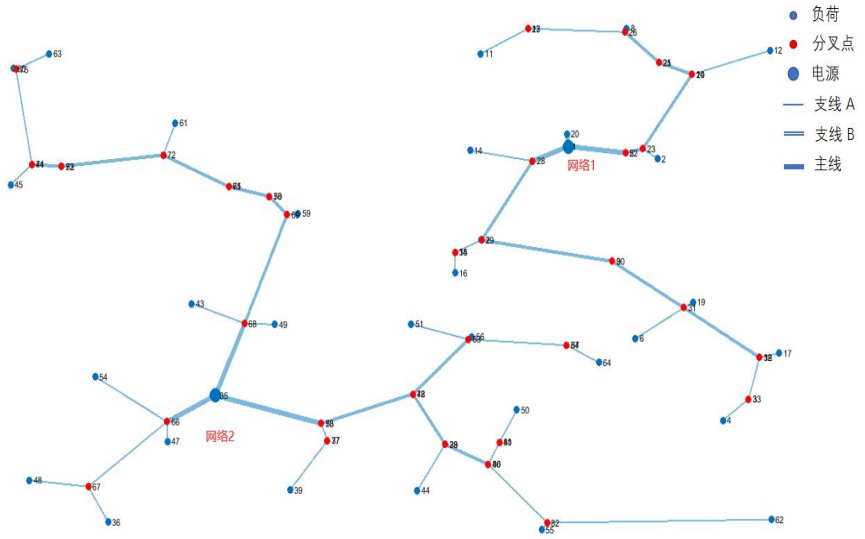

图 8　基于启发式二分类算法求解的两个单供配电网络结果示意图

并且可以得到，树状配电网的分叉点坐标如下表：

节点编号	x 坐标（km）	y 坐标（km）
22	80.05497168	74.5861947
23	82.25011159	75.37755193
24	88.53212733	89.8994971
...	...	...
84	72.57509576	37.03395166

表 5　问题二建立的分叉点坐标部分展示（详见附录 8.2.5）

由此计算出的用户用电可靠性为：

节点编号	x 坐标（km）	y 坐标（km）	用户可靠性（*100%）
2	84.15600875	73.42296912	0.950220486
3	95.73840226	26.53220362	0.815430414
4	92.45808952	22.37704047	0.806796731
...	...	...	...
65	29.55072508	68.01783712	0.874654371

表 6　问题二求解的用户用电可靠性部分展示（详见附录 8.2.6）

根据求解结果，可以建立出关于用电可靠性的分布直方图，如下图所展示：

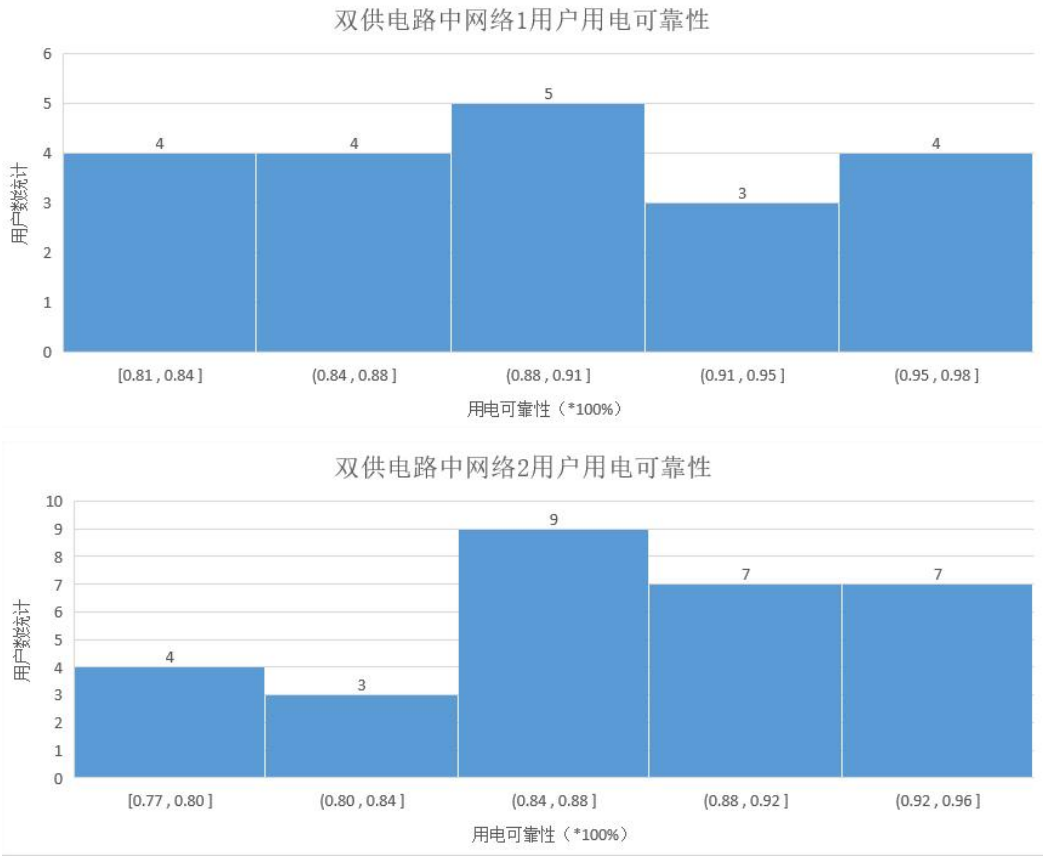

图 9　问题二求解的用电可靠性频率分布直方图

通过直方图，我们可以分别求出：网络 1 的用电可靠性均值为 89%，方差为 0.27%；网络 2 的用电可靠性均值为 88%，方差为 0.28%。

与问题一的单供网络数据进行对比可知，两个网络的用电可靠性均值都比单供网络的均值要高，且方差更小。可以发现，在从单电源的单供电源网络更变为双电源的两个单供电源网络之后，能够更好地保障用户的用电可靠性，符合题目要求，也契合现实生活。

5.3 问题三、四：设计建造总花费上限为 X，网中最低的用电可靠性达到最大的双供配电网络、设计网中最低的用电可靠性下限为 Y%，布线成本最小的双供配电网络

问题三是在问题二的基础上添加联络线产生的。在原有两个单供供电网中添加联络线，并保证功率约束与建造总花费合理的约束的基础上，对网中最低的用电可靠性这一目标函数求取最大值，最终得到用电可靠性最大的双供配电网络。

问题四与问题三类似，都是在问题二的基础上添加联络线产生的。在原有两个单供供电网中添加联络线，并保证功率约束与用户用电最低可靠性在要求的阈值之上，对网中最低建造成本这一目标函数求取最小值，最终得到建造费用最低的双供配电网络。

由于问题三与问题四背景相似，仅在约束条件和目标求解问题上有所不同。因此，我们都基于同样的假设建立求解模型，仅在目标函数上有所区别。

5.3.1 模型的建立——双供配电网模型

（1）供电分配原则

题目中说明，配电网要能够满足一个用户全部需求功率，否则断开该用户。这意味着用户只能选择一个电源，或都不选择，不可有两个电源共同供电。因此设用户用电的状态变量为离散的变量 $C[i]$：

$$C[i] = \begin{cases} 0, & \text{断开} \\ 1, & \text{选择电源 1} \\ 2, & \text{选择电源 2} \end{cases} \qquad (26)$$

又因为，电源需要首先满足其所在单供配电网内用户的需求，双供电源多余功率的分配优先提高全配电网供电功率总和，然后提升全配电网最低的用电可靠性，所以电源分配遵循优先供给原则。首先考虑使用本供电网中电源，如果本供电网电源无法连接，才会考虑使用另一个网络中的电源。

（2）双供配电网中的可靠性计算

设位于网络 1 的双端供电网中的点 i 的可靠性为 $r[i]$，将两点 x, y 间的可靠性表示为 $r(x, y)$。

由假设 3“双供电源中，联络点一定位于负荷到电源的路上”，令联络线为 l_k，其连接网络 1 和网络 2 的两端点分别为p_{k_1}、p_{k_2}，则可以对位于网络 1 的用户 i，分析其可靠性，为：该点到其自身网络的电源的可靠性叠加上，当该电源与本网络联络线的端点失效时，本网络中联络线端点到节点 i、联络线自身可靠性，与另一端网络的电源到联络线另一端的可靠性的积所构成。（本假设中节点 i 位于网络 1，对于在网络 2 中的节点求解同理）即：

$$r[i] = r(bat_1, i) + (1 - r(bat_1, p_{k_1})) * r(p_{k_1}, i) * r(p_{k_1}, p_{k_2}) * r(p_{k_2}, bat_2) \quad (27)$$

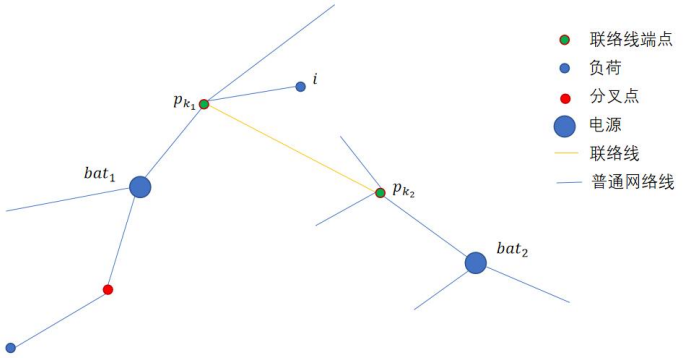

图 10　配电网中点 i 的可靠性示意图

也即，点 i 的可靠性为：使用电源 1 时的可靠性，叠加上电源 1 不通而使用电源 2 的可靠性，且其遵循条件概率原则。

进一步展开可知，联络线上的可靠性可以写为：

$$r(p_{k_1}, p_{k_2}) = (1 - 0.5\%) * (1 - 0.2\%) * (1 - 0.002 * dis(p_{k_1}, p_{k_2}) \quad (28)$$

（3）双供配电网中总功率的计算

由于上述供电原则不能确定究竟使用哪一个电源供电（因为这与电源与用户间通路是否连通有关），所以我们使用期望的方法来刻画功率。

设 W_u 为网络 N 中节点的负荷需求，$P_{1_{max}}$、$P_{2_{max}}$为拓展前网络 1、2 的负荷需求之和，则有：

$$P_{i_{max}} = \sum_{u \in N} W_u \ , i = 1, 2 \qquad (29)$$

令 P_1、P_2 为拓展后网络 1、2 的功率，网络 1、2 中的负荷设为集合 H_1、H_2，扩展后，负荷可以由两个电源中的一个提供。因此单个网络中的功率将不止包括有自身负荷的功率，还包括对方网络中无法由对方电源直接供电的部分。

故使用期望的方式将功率写为，己方正常供电部分与对方异常扩展两部分之和，即：

$$P_1 = \sum_{i \in H_1} P_i * r(bat_1, i) + \sum_{j \in H_2} P_j * (1 - r(bat_2, p_{k_2})) * r(p_{k_2}, j) * r(p_{k_1}, p_{k_2}) * r(bat_1, p_{k_1}) \quad (30)$$

$$P_2 = \sum_{j \in H_2} P_j * r(bat_2, j) + \sum_{i \in H_1} P_i * (1 - r(bat_1, p_{k_1})) * r(p_{k_1}, i) * r(p_{k_1}, p_{k_2}) * r(bat_2, p_{k_2}) \quad (31)$$

而从题目中得知，电源扩展功率的上限为 50%，所以存在约束：

$$\begin{cases} P_1 \le (1.1 * P_{max}) * (1 + 50\%) \\ P_2 \le (1.1 * P_{max}) * (1 + 50\%) \end{cases} \quad (32)$$

（4）双供配电网中造价成本的计算

双供配电网中总花费分为 4 部分：1，2 号电源所在的网络花费分别为 $cost_1$、$cost_2$，联络线花费为 $cost_{link}$，扩展功率花费 $cost_{pow}$。即：

$$cost_{total} = cost_1 + cost_2 + cost_{link} + cost_{pow} \quad (33)$$

对扩展功率进行展开分析，有：

$$cost_{pow} = \begin{cases} (P_1 - 1.1 * P_{1_{max}}) * cost_{p_1} + (P_2 - 1.1 * P_{2_{max}}) * cost_{p_2}, P_1 \ge P_{1_{max}} \text{ 且 } P_2 \ge P_{2_{max}} \\ (P_1 - 1.1 * P_{1_{max}}) * cost_{p_1}, P_1 \ge P_{1_{max}} \text{ 且 } P_2 < P_{2_{max}} \\ (P_2 - 1.1 * P_{2_{max}}) * cost_{p_2}, P_2 \ge P_{2_{max}} \text{ 且 } P_1 < P_{1_{max}} \end{cases} \quad (34)$$

（5）双供配电网中的联络线模型

由题目中要求，扩展电源会优先提高最低可靠性的负荷的可靠性。因此，本模型使用贪心算法构造联络线——将从最低的可靠性用户入手，进行联络线的选择。

1、首先，设 1 号与 2 号网络为 N_1、N_2，$V[N_i]$ 为网络 i 中的负荷集合，$r[V(N_i)]$ 即为网络 i 中负荷的用电可靠性的集合。我们需要找到两个图中可靠性最低的两个负荷节点 x_1、x_2，对其进行优化，满足：

$$\begin{cases} r[x_1] = min(r[V(N_1)]) \\ r[x_2] = min(r[V(N_2)]) \end{cases} \quad (35)$$

2、根据树的性质（任意两节点之间有唯一通路）可知，此时 x_1、x_2 各自有唯一条到电源（根节点）的路 $path(x_1)$、$path(x_2)$，则可以选择这两条路上的点分别建立一条联络线，对这两个点的可靠性进行提升。

需要注意的是，为了保证联络线仅影响优化目标而不会影响其他负荷，默认仅将优化目标的两个负荷的前的开关设定为“故障时开启”，其他的负荷前开

关默认故障时关闭，这样保证了本次优化仅提高了优化目标的可靠性，不影响其他负荷的可靠性，也不会造成多余的功率负担。

3、设 $r'[x]$ 为优化后的 x 节点可靠性。若已有可用的联络线，则判断是否开关已设为“故障时开启”，如果是，则重新寻找联络线，使得：

$$r'[x] > r[x] \qquad (36)$$

如果不是，则启用已有的联络线，即将开关设为“故障时开启”。

4、如若没有可用的联络线，则搜索花费最少的联络线，令：

$$cost(p_{k_1}, p_{k_2}) = min(cost(p_i, p_j)) \ , p_i \in \ path(x_1) \ , p_j \in \ path(x_2) \qquad (37)$$

5、目标函数的建立

问题三：

依次重复进行下去，当有：

$$cost(p_{k_1}, p_{k_2}) \leq X \qquad (38)$$

时模型终止，找到最终结果。如果中途发现无法继续优化或者功率超过了上限，则终止程序，搜索失败。

问题四：

依次重复进行下去，当有：

$$min(r[V(N_1) + V(N_2)]) \geq Y \qquad (39)$$

时模型终止，找到最终结果。如果中途发现无法继续优化或者功率超过了上限，则终止程序，搜索失败。

统一的约束条件：

1、使用 2 类开关；

2、使用线路为主线。

基于前面所描述的贪心思想，结合问题二的两个单供配电网模型，建立总花费上限为 X，网中最低的用电可靠性达到最大的双供配电网络模型（问题三）：

$$cost(p_{k_1}, p_{k_2}) \leq X, \quad p_i \in \ path(x_1) \ , p_j \in \ path(x_2)$$

$$
\begin{cases}
r[i] = \begin{cases} r(bat_1, i) + (1 - r(bat_1, p_{k_1})) * r(p_{k_1}, i) * r(p_{k_1}, p_{k_2}) * r(p_{k_2}, bat_2), i \in N_1 \\ r(bat_2, i) + (1 - r(bat_2, p_{k_2})) * r(p_{k_2}, i) * r(p_{k_1}, p_{k_2}) * r(p_{k_1}, bat_1), i \in N_2 \end{cases} \\
r(p_{k_1}, p_{k_2}) = (1 - 0.5\%) * (1 - 0.2\%) * (1 - 0.002 * dis(p_{k_1}, p_{k_2}) \\
\begin{cases} P_1 = \sum_{i \in H_1} P_i * r(bat_1, i) + \sum_{j \in H_2} P_j * (1 - r(bat_2, p_{k_2})) * r(p_{k_2}, j) * r(p_{k_1}, p_{k_2}) * r(bat_1, p_{k_1}) \\ P_2 = \sum_{j \in H_2} P_j * r(bat_2, j) + \sum_{i \in H_1} P_i * (1 - r(bat_1, p_{k_1})) * r(p_{k_1}, i) * r(p_{k_1}, p_{k_2}) * r(bat_2, p_{k_2}) \\ \begin{cases} P_1 \le (1.1 * P_{1_{max}}) * (1 + 50\%) \\ P_2 \le (1.1 * P_{2_{max}}) * (1 + 50\%) \end{cases} \end{cases} \\
cost_{total} = cost_1 + cost_2 + cost_{link} + cost_{pow} \\
cost_{pow} = \begin{cases} (P_1 - 1.1 * P_{1_{max}}) * cost_{p_1} + (P_2 - 1.1 * P_{2_{max}}) * cost_{p_2}, P_1 \ge P_{1_{max}} \text{ 且 } P_2 \ge P_{2_{max}} \\ (P_1 - 1.1 * P_{1_{max}}) * cost_{p_1}, P_1 \ge P_{1_{max}} \text{ 且 } P_2 < P_{2_{max}} \\ (P_2 - 1.1 * P_{2_{max}}) * cost_{p_2}, P_2 \ge P_{2_{max}} \text{ 且 } P_1 < P_{1_{max}} \end{cases} \\
\begin{cases} r[x_1] = min(r[V(N_1)]) \\ r[x_2] = min(r[V(N_2)]) \end{cases} \\
r'[x] > r[x] \\
dis(i, j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \\
length(k) = \sum_{G(i,j) = 1 \text{ 且 } type(i,j) = k} dis(i, j), k = 1, 2, 3 \\
r[i] = r|fa_i| * (1 - 0.5\%) * (1 - 0.2\%) * (1 - 0.002 * dis(i, fa_i))
\end{cases}
\tag{40}
$$

此外，还能建立网中最低的用电可靠性下限为 Y%，布线成本最小的双供配电网络模型（问题四）：

$$min(r[V(N_1)+V(N_2)]) \geq Y$$

$$\begin{cases} r[i]=\begin{cases} r(bat_1,i)+(1-r(bat_1,p_{k_1}))*r(p_{k_1},i)*r(p_{k_1},p_{k_2})*r(p_{k_2},bat_2), i\in N_1 \\ r(bat_2,i)+(1-r(bat_2,p_{k_2}))*r(p_{k_2},i)*r(p_{k_1},p_{k_2})*r(p_{k_1},bat_1), i\in N_2 \end{cases} \\ r(p_{k_1},p_{k_2})=(1-0.5\%)*(1-0.2\%)*(1-0.002*dis(p_{k_1},p_{k_2}) \\ \begin{cases} P_1=\sum_{i\in H_1}P_i*r(bat_1,i)+\sum_{j\in H_2}P_j*(1-r(bat_2,p_{k_2}))*r(p_{k_2},j)*r(p_{k_1},p_{k_2})*r(bat_1,p_{k_1}) \\ P_2=\sum_{j\in H_2}P_j*r(bat_2,j)+\sum_{i\in H_1}P_i*(1-r(bat_1,p_{k_1}))*r(p_{k_1},i)*r(p_{k_1},p_{k_2})*r(bat_2,p_{k_2}) \\ \begin{cases} P_1\leq(1.1*P_{1_{max}})*(1+50\%) \\ P_2\leq(1.1*P_{2_{max}})*(1+50\%) \end{cases} \end{cases} \\ cost_{total}=cost_1+cost_2+cost_{link}+cost_{pow} \\ cost_{pow}=\begin{cases} (P_1-1.1*P_{1_{max}})*cost_{p_1}+(P_2-1.1*P_{2_{max}})*cost_{p_2}, P_1\geq P_{1_{max}}\text{ 且 }P_2\geq P_{2_{max}} \\ (P_1-1.1*P_{1_{max}})*cost_{p_1}, P_1\geq P_{1_{max}}\text{ 且 }P_2<P_{2_{max}} \\ (P_2-1.1*P_{2_{max}})*cost_{p_2}, P_2\geq P_{2_{max}}\text{ 且 }P_1<P_{1_{max}} \end{cases} \\ \begin{cases} r[x_1]=min(r[V(N_1)]) \\ r[x_2]=min(r[V(N_2)]) \end{cases} \\ r'[x]>r[x] \\ dis(i,j)=\sqrt{(x_i-x_j)^2+(y_i-y_j)^2} \\ length(k)=\sum_{G(i,j)=1\text{ 且 }type(i,j)=k}dis(i,j), k=1,2,3 \\ r[i]=r|fa_i|*(1-0.5\%)*(1-0.2\%)*(1-0.002*dis(i,fa_i)) \end{cases} \tag{41}$$

5.3.2 模型的求解

基于贪心算法的双供电网模型的求解步骤如下：

Step1：预处理

a）使用 $MATLAB$ 中的 $cell$ 数组进行网络的存储，$cell$ 允许存储维度不同的对象，以便对两个供电网络进行对称性的操作；

b）通过 dfs 计算基础用电可靠性，同时处理得父子关系数组 $fa[i]$；

c）根据公式（29）计算网络负荷需求之和 $P[i]$；

Step2：定义算法执行相关的原子操作

a）clac_optimization()：计算网络优化后的用电可靠性，由公式（27）、（28）得到优化用户的可靠性，同时基于公式（30）-（32）和 P[i]计算拓展供电量 pow_{add} 和相应的费用支出 $cost_{add}$；

b）set_ncp()：设置最近联络点 $ncp[i]$，函数递归执行；

c）get_ancestors()：获取所有祖先分叉点，即求 $path(x_1, x_2)$，以备构造可能的联络线；

d）find_kid_ncp()：搜索子树中的联络点，根据模型约束，联络点的子树不含进行联络点，通过该函数进行联络点的调整；

Step3：贪心算法进行联络线方案设计

a）设置最大优化轮数 t，进行迭代；

b）调用 clac_optimization()，根据公式（33）、（34）计算当前方案费用，根据公式（35）求两个供电网络中可靠性最低点 x_1, x_2；

c）判断 x_1, x_2 是否已优化，若已优化算法结束；

d）判断是否存在联络线，若存在则相应增加供电方案，跳过本轮；

e)调用 get_ancestors()获取 $path(x_1, x_2)$，通过排序选择备选的联络点 c_1, c_2；

f）对 c_1, c_2 调用 find_kid_ncp()，若均返回 0 则进行增加联络线操作，调用 set_ncp()设置联络点，否则进行联络线的调整；

g）若未达最大迭代轮数，转到 b）；否则算法结束。

Step4：通过 GraphPlot 的相关 API 将拥有两个电源的最优配电网可视化

5.3.3 结果分析

问题三、四中设置的电源变电站信息如下表所示：

节点编号	出线电压(千伏)	额定供电量(兆瓦)	可扩展供电量	扩展供量价格	x 坐标(km)	y 坐标(km)
1	10kV	负荷需求之和*1.1	50%	10 元/兆	72.7884	75.8243
35	10kV	负荷需求之和*1.1	50%	10 元/兆	27.862	27.2309

表 7 电源变电站相关信息

基于上述建立的双供电路网模型求解后，得到的配电网如下图所示（由于求解的是最优方案，所以问题三、四的拓扑图一致）：

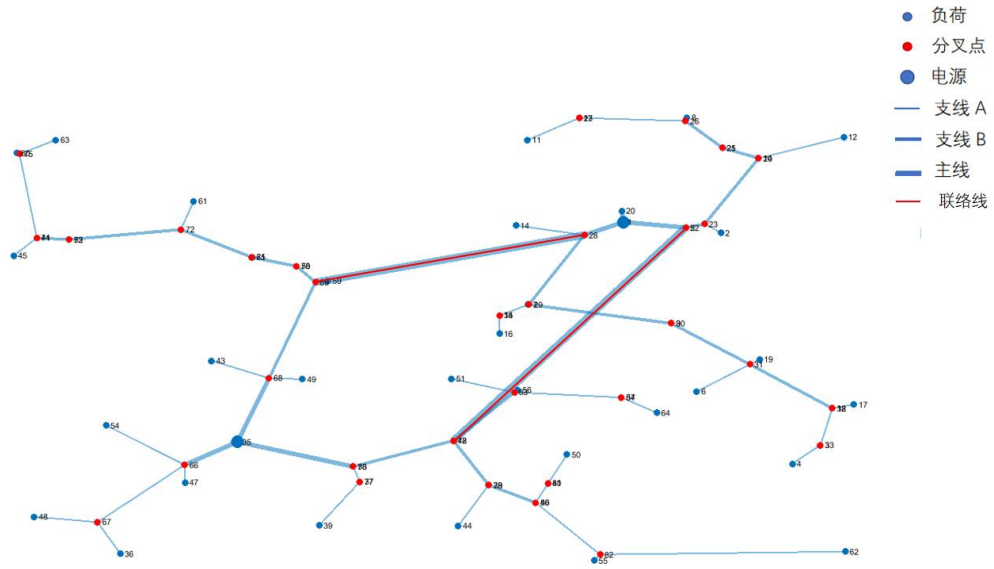

图 11 所建立的双供电路网络结果示意图（问题三、问题四）

并且可以得到，用户用电可靠性及建造花费为：

优化轮数	联络线方案花费	最低用户用电可靠性
0	0	0.766735674
1	8302.628607	0.774243156
2	8302.628607	0.780277867
...	...	...
41	102577.4109	0.91372766

表 8 问题三求解的用户用电可靠性部分展示（详见附录 8.2.8）

根据求解结果，可以建立出关于建造联络线的花费及用电可靠性的折线统计图，如下所示：

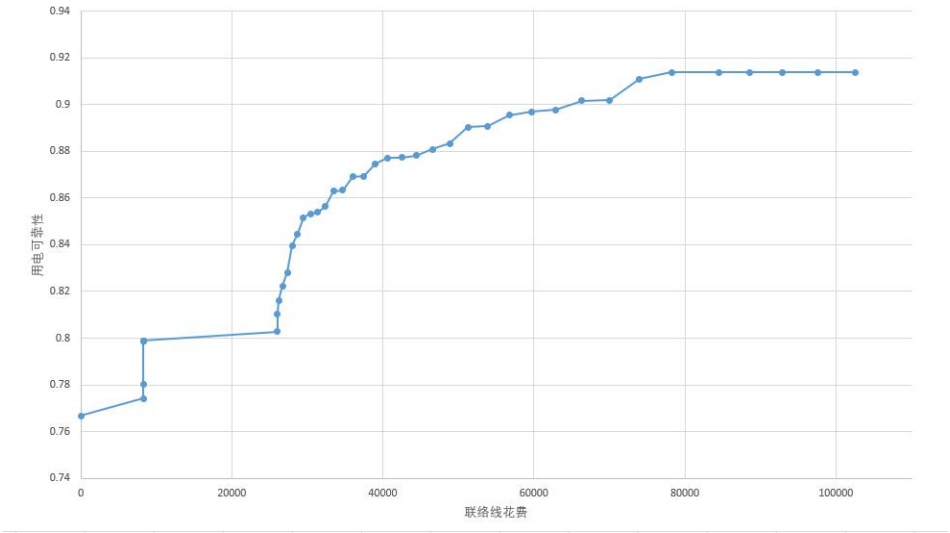

图 12　用户用电可靠性与联络线花费的函数图

借助该图像，我们可以看到，随着成本的增加（联络线的选择及连接），用户的用电可靠性得到了提升，最终随着优化轮数的提高，在限制的成本之内，最大的最低用电可靠性可以达到 91.4%。

六、模型的评价与优化

6.1 模型的优点

1、本文利用辐射式的拓扑结构来建立电网模型，符合实际情况，并在用电可靠性的求解方面较为精确。

2、本文基于传统的几种聚类方法，别出心裁地提出了一种启发式的二分类算法，能够良好地契合问题目标，进行建模求解。

6.2 模型的缺点

1、由于专业知识的不足以及模型假设条件导致的简化，本文建立的模型无法完全模拟生活中配电网建造时所要考虑的各种情况。

2、问题三、四中使用期望的方法来刻画电网功率，在一定程度上存在数据误差。

6.3 模型的优化

1、模型可深入结合现实生活，对限制条件进行更为精细的描述，并且对功率的描述进行进一步优化后，可以良好地模拟现实生活中电网建造时候的拓扑情况，且获得的可靠性亦可作为一定的价值参考。

七、参考文献

[1]王聃，孙哲军，张敬思，周刚.基于供电可靠性的配电网规划[J].设计应用（电力），2021（12）：59-63.

[2]张宗东，杨立.基于供电可靠性的配电网规划实践浅谈[J].教育论坛，2017（6）：339.

[3]王聃，陆渊超，张敬思，周刚.基于改进最小生成树的配网线路优化[J].设计应用（工业控制），2022（1）：46-49.

八、附录

【说明】由于上传格式限制,现将求解中所用到的所有数据转化成表格形式,贴于附录处。

8.1 附录清单：

（1）负荷测试数据集'test_data.xlsx'；

（2）问题一分叉点坐标'bifurcation_coordinates_t1.xlsx'；

（3）问题一用户用电可靠性'power_reliability_t1.xlsx'；

（4）问题一电源、分叉点及用户用电可靠性信息' result_t1.xlsx' ；

（5）问题二分叉点坐标'bifurcation_coordinates_t2.xlsx'；

（6）问题二用户用电可靠性'power_reliability_t2.xlsx'；

（7）问题二电源、分叉点及用户用电可靠性信息'result_t2.xlsx'；

（8）问题三、四的优化结果信息'algo_optim_curve.xlsx'；

（9）问题一代码

（10）问题二代码

（11）问题三代码

（12）问题四代码

8.2 附录数据：

8.2.1 测试数据集'test_data.xlsx'：

序号	x 坐标（km）	y 坐标（km）	需求量（兆瓦）
1	60.49906419	38.72454315	9.7540405
2	14.21871593	2.513498571	27.84982189
3	42.11122538	18.41002894	54.68815192
4	72.57752675	37.03626865	95.75068354
5	84.15600875	73.42296912	96.48885352
6	57.10258728	17.68550576	15.76130817
7	95.73840226	26.53220362	97.05927818
8	92.45808952	22.37704047	95.71669482
9	37.35638076	8.750034958	48.53756487
10	64.01165482	18.06168878	80.02804689
11	4.505110747	72.31734792	14.18863386
12	34.74376456	66.06168245	42.17612826
13	38.38686011	62.73465024	91.57355252
14	2.164981463	91.05699885	79.22073296

15	80.05586563	74.58474843	95.94924264
16	81.31128136	38.33063186	65.57406992
17	61.72792323	57.54948597	3.571167857
18	53.00517048	27.50697558	84.91293059
19	24.86289597	45.16387705	93.39932478
20	22.7712826	80.44495836	67.87351549
21	98.61042419	2.999195027	75.77401306
22	53.56641907	8.70772199	74.31324681
23	80.20914406	98.91449097	39.22270195
24	6.69462584	93.93983619	65.54778902
25	1.817753364	68.38386137	17.11866878
26	78.37364801	53.41375679	70.6046088
27	88.53594509	89.90048989	3.183284638
28	62.59376261	13.78689924	27.6922985
29	21.78015937	18.21410759	4.617139063
30	4.181986397	10.69416586	9.713178124
31	61.64434851	93.96610102	82.34578283
32	35.4455731	41.06290901	69.4828623
33	98.4349417	94.5579189	31.70994801
34	67.66446784	98.83022623	95.02220488
35	76.68313872	33.66992644	3.44460805
36	66.23818604	24.41652868	43.87443597
37	29.55072508	68.01783712	38.15584571
38	52.78468304	41.15935134	76.55167881
39	60.2638218	75.05200559	79.51999011
40	58.35331743	55.17925149	18.68726046
41	58.35706188	51.181992	48.97643958
42	8.2592727	71.95701349	44.55862007
43	99.61561113	35.4534305	64.63130101
44	97.12588152	34.64487613	70.93648309
45	88.65438618	45.4694865	75.4686682
46	41.3427289	21.77320684	27.6025077
47	12.56545874	30.89145936	67.97026769
48	72.61044317	78.2872073	65.5098004
49	69.3787615	0.980225226	16.26117352
50	84.3213338	92.23319978	11.89976816

8.2.2 问题一分叉点坐标'bifurcation_coordinates_t1.xlsx'：

节点编号	x 坐标（km）	y 坐标（km）
52	38.40748071	62.72762528
53	34.74809205	66.06018075
54	29.55325777	68.0146831

55	21.34988112	74.73255275
56	8.260983903	71.96536804
57	4.507352534	72.3143648
58	2.722982578	90.79433517
59	52.97786298	43.12745711
60	60.4809788	38.70681006
61	72.57718558	37.03998762
62	76.4872367	35.72684841
63	81.31217919	38.33446225
64	86.61669052	42.73937497
65	97.1153152	34.65067812
66	95.7386418	26.53693161
67	92.48340528	22.37237898
68	53.00566906	27.49425993
69	57.11102433	17.69177753
70	62.59311436	13.78395011
71	64.0124269	18.06128383
72	42.5038995	20.96176244
73	41.97012391	18.40838504
74	36.17722724	14.74930281
75	21.77822377	18.22479915
76	12.65823902	8.144701024
77	35.44471093	41.06739834
78	56.57282698	51.8047929
79	58.35689723	55.17683214
80	61.72278912	57.55315359
81	64.11211433	72.44721806
82	72.67741891	77.8780715
83	80.04929367	75.35397636
84	81.68981787	76.04016139
85	88.53106393	89.89728459
86	84.31448936	92.22511298
87	79.99271345	98.03622052
88	67.67015597	98.83598478

8.2.3 问题一用户用电可靠性'power_reliability_t1.xlsx':

节点编号	x 坐标（km）	y 坐标（km）	用户可靠性（*100%）
2	60.49906419	38.72454315	0.944154412
3	14.21871593	2.513498571	0.786398791
4	42.11122538	18.41002894	0.872546006
5	72.57752675	37.03626865	0.914698236
6	84.15600875	73.42296912	0.849843266

7	57.10258728	17.68550576	0.886680268
8	95.73840226	26.53220362	0.817998014
9	92.45808952	22.37704047	0.80365928
10	37.35638076	8.750034958	0.844362032
11	64.01165482	18.06168878	0.85479724
12	4.505110747	72.31734792	0.842890903
13	34.74376456	66.06168245	0.927210034
14	38.38686011	62.73465024	0.943039782
15	2.164981463	91.05699885	0.804931719
16	80.05586563	74.58474843	0.863771074
17	81.31128136	38.33063186	0.884702641
18	61.72792323	57.54948597	0.944552307
19	53.00517048	27.50697558	0.912308865
20	24.86289597	45.16387705	0.905972937
21	22.7712826	80.44495836	0.874559343
22	98.61042419	2.999195027	0.771039915
23	53.56641907	8.70772199	0.869571387
24	80.20914406	98.91449097	0.79135818
25	6.69462584	93.93983619	0.797759559
26	1.817753364	68.38386137	0.834868351
27	78.37364801	53.41375679	0.843040624
28	88.53594509	89.90048989	0.823734589
29	62.59376261	13.78689924	0.868639974
30	21.78015937	18.21410759	0.823674853
31	4.181986397	10.69416586	0.781611958
32	61.64434851	93.96610102	0.755910665
33	35.4455731	41.06290901	0.927002166
34	98.4349417	94.5579189	0.805711298
35	67.66446784	98.83022623	0.767795657
36	76.68313872	33.66992644	0.897095731
37	66.23818604	24.41652868	0.84328676
38	29.55072508	68.01783712	0.910509077
39	52.78468304	41.15935134	0.963875872
40	60.2638218	75.05200559	0.901210007
41	58.35331743	55.17925149	0.959108443
42	58.35706188	51.181992	0.969615535
43	8.2592727	71.95701349	0.855264428
44	99.61561113	35.4534305	0.83315063
45	97.12588152	34.64487613	0.837529252
46	88.65438618	45.4694865	0.860507377
47	41.3427289	21.77320684	0.881042413
48	12.56545874	30.89145936	0.797890292
49	72.61044317	78.2872073	0.884249496

50	69.3787615	0.980225226	0.843470666
51	84.3213338	92.23319978	0.810089476

8.2.4 问题一电源、分叉点及用户用电可靠性信息’result_t1.xlsx’：

节点编号	节点性质	x 坐标（km）	y 坐标（km）	用户可靠性（*100%）
1	电源	53.0191	49.9741	无
2	负荷	60.49906419	38.72454315	0.944154412
3	负荷	14.21871593	2.513498571	0.786398791
4	负荷	42.11122538	18.41002894	0.872546006
5	负荷	72.57752675	37.03626865	0.914698236
6	负荷	84.15600875	73.42296912	0.849843266
7	负荷	57.10258728	17.68550576	0.886680268
8	负荷	95.73840226	26.53220362	0.817998014
9	负荷	92.45808952	22.37704047	0.80365928
10	负荷	37.35638076	8.750034958	0.844362032
11	负荷	64.01165482	18.06168878	0.85479724
12	负荷	4.505110747	72.31734792	0.842890903
13	负荷	34.74376456	66.06168245	0.927210034
14	负荷	38.38686011	62.73465024	0.943039782
15	负荷	2.164981463	91.05699885	0.804931719
16	负荷	80.05586563	74.58474843	0.863771074
17	负荷	81.31128136	38.33063186	0.884702641
18	负荷	61.72792323	57.54948597	0.944552307
19	负荷	53.00517048	27.50697558	0.912308865
20	负荷	24.86289597	45.16387705	0.905972937
21	负荷	22.7712826	80.44495836	0.874559343
22	负荷	98.61042419	2.999195027	0.771039915
23	负荷	53.56641907	8.70772199	0.869571387
24	负荷	80.20914406	98.91449097	0.79135818
25	负荷	6.69462584	93.93983619	0.797759559
26	负荷	1.817753364	68.38386137	0.834868351
27	负荷	78.37364801	53.41375679	0.843040624
28	负荷	88.53594509	89.90048989	0.823734589
29	负荷	62.59376261	13.78689924	0.868639974
30	负荷	21.78015937	18.21410759	0.823674853
31	负荷	4.181986397	10.69416586	0.781611958
32	负荷	61.64434851	93.96610102	0.755910665
33	负荷	35.4455731	41.06290901	0.927002166
34	负荷	98.4349417	94.5579189	0.805711298
35	负荷	67.66446784	98.83022623	0.767795657
36	负荷	76.68313872	33.66992644	0.897095731
37	负荷	66.23818604	24.41652868	0.84328676

38	负荷	29.55072508	68.01783712	0.910509077
39	负荷	52.78468304	41.15935134	0.963875872
40	负荷	60.2638218	75.05200559	0.901210007
41	负荷	58.35331743	55.17925149	0.959108443
42	负荷	58.35706188	51.181992	0.969615535
43	负荷	8.2592727	71.95701349	0.855264428
44	负荷	99.61561113	35.4534305	0.83315063
45	负荷	97.12588152	34.64487613	0.837529252
46	负荷	88.65438618	45.4694865	0.860507377
47	负荷	41.3427289	21.77320684	0.881042413
48	负荷	12.56545874	30.89145936	0.797890292
49	负荷	72.61044317	78.2872073	0.884249496
50	负荷	69.3787615	0.980225226	0.843470666
51	负荷	84.3213338	92.23319978	0.810089476
52	分叉点	38.40748071	62.72762528	无
53	分叉点	34.74809205	66.06018075	无
54	分叉点	29.55325777	68.0146831	无
55	分叉点	21.34988112	74.73255275	无
56	分叉点	8.260983903	71.96536804	无
57	分叉点	4.507352534	72.3143648	无
58	分叉点	2.722982578	90.79433517	无
59	分叉点	52.97786298	43.12745711	无
60	分叉点	60.4809788	38.70681006	无
61	分叉点	72.57718558	37.03998762	无
62	分叉点	76.4872367	35.72684841	无
63	分叉点	81.31217919	38.33446225	无
64	分叉点	86.61669052	42.73937497	无
65	分叉点	97.1153152	34.65067812	无
66	分叉点	95.7386418	26.53693161	无
67	分叉点	92.48340528	22.37237898	无
68	分叉点	53.00566906	27.49425993	无
69	分叉点	57.11102433	17.69177753	无
70	分叉点	62.59311436	13.78395011	无
71	分叉点	64.0124269	18.06128383	无
72	分叉点	42.5038995	20.96176244	无
73	分叉点	41.97012391	18.40838504	无
74	分叉点	36.17722724	14.74930281	无
75	分叉点	21.77822377	18.22479915	无
76	分叉点	12.65823902	8.144701024	无
77	分叉点	35.44471093	41.06739834	无
78	分叉点	56.57282698	51.8047929	无
79	分叉点	58.35689723	55.17683214	无
80	分叉点	61.72278912	57.55315359	无

81	分叉点	64.11211433	72.44721806	无
82	分叉点	72.67741891	77.8780715	无
83	分叉点	80.04929367	75.35397636	无
84	分叉点	81.68981787	76.04016139	无
85	分叉点	88.53106393	89.89728459	无
86	分叉点	84.31448936	92.22511298	无
87	分叉点	79.99271345	98.03622052	无
88	分叉点	67.67015597	98.83598478	无

8.2.5 问题二分叉点坐标'bifurcation_coordinates_t2.xlsx'：

节点编号	x 坐标（km）	y 坐标（km）
22	80.05497168	74.5861947
23	82.25011159	75.37755193
24	88.53212733	89.8994971
25	84.32147358	92.23350345
26	80.00236904	98.1503681
27	67.6653737	98.82849116
28	68.24322022	72.93540567
29	61.77881038	57.57414137
30	78.37338837	53.41342262
31	87.51619344	44.35956039
32	97.12631284	34.64479306
33	95.73844191	26.53223422
34	58.35342675	55.17884685
66	21.70816726	22.14683133
67	11.57488137	9.482466249
68	31.54184544	41.3031267
69	36.94821239	62.51424206
70	34.73961485	66.05488438
71	29.54699677	68.01785098
72	21.265507	74.15064934
73	8.259905183	71.95710762
74	4.506429801	72.31774309
75	2.482383575	90.85796249
76	41.33716851	21.77547378
77	42.106376	18.40855239
78	53.00733991	27.50262744
79	57.10345091	17.68708313
80	62.59039073	13.79575094
81	64.01115718	18.06128091
82	70.08798521	2.331152629
83	60.09236306	38.17997884

84	72.57509576	37.03395166

8.2.6 问题二用户用电可靠性'power_reliability_t2.xlsx':

节点编号	x 坐标（km）	y 坐标（km）	用户可靠性（*100%）
2	84.15600875	73.42296912	0.950220486
3	95.73840226	26.53220362	0.815430414
4	92.45808952	22.37704047	0.806796731
5	80.05586563	74.58474843	0.966670709
6	81.31128136	38.33063186	0.849458364
7	61.72792323	57.54948597	0.931555875
8	80.20914406	98.91449097	0.882668498
9	78.37364801	53.41375679	0.893492845
10	88.53594509	89.90048989	0.918727983
11	61.64434851	93.96610102	0.842942251
12	98.4349417	94.5579189	0.898626333
13	67.66446784	98.83022623	0.856191465
14	60.2638218	75.05200559	0.954545737
15	58.35331743	55.17925149	0.91741429
16	58.35706188	51.181992	0.910081507
17	99.61561113	35.4534305	0.830543342
18	97.12588152	34.64487613	0.834913125
19	88.65438618	45.4694865	0.861666919
20	72.61044317	78.2872073	0.983165336
21	84.3213338	92.23319978	0.903528424
36	14.21871593	2.513498571	0.913798014
37	42.11122538	18.41002894	0.939409416
38	57.10258728	17.68550576	0.895453142
39	37.35638076	8.750034958	0.919196375
40	64.01165482	18.06168878	0.863268536
41	4.505110747	72.31734792	0.810508286
42	53.00517048	27.50697558	0.921349724
43	24.86289597	45.16387705	0.93789876
44	53.56641907	8.70772199	0.878172476
45	1.817753364	68.38386137	0.802786492
46	62.59376261	13.78689924	0.877217716
47	21.78015937	18.21410759	0.95787983
48	4.181986397	10.69416586	0.91372766
49	35.4455731	41.06290901	0.945144979
50	66.23818604	24.41652868	0.851642866
51	52.78468304	41.15935134	0.877400084
52	8.2592727	71.95701349	0.822417099
53	41.3427289	21.77320684	0.952600275

54	12.56545874	30.89145936	0.941045796
55	69.3787615	0.980225226	0.84465135
56	60.49906419	38.72454315	0.890258693
57	72.57752675	37.03626865	0.863039983
58	34.74376456	66.06168245	0.890692718
59	38.38686011	62.73465024	0.901893019
60	2.164981463	91.05699885	0.774243156
61	22.7712826	80.44495836	0.839635534
62	98.61042419	2.999195027	0.798892972
63	6.69462584	93.93983619	0.766735674
64	76.68313872	33.66992644	0.853880789
65	29.55072508	68.01783712	0.874654371

8.2.7 问题二分叉点坐标及用户用电可靠性'result_t2.xlsx'：

节点编号	节点性质	x 坐标（km）	y 坐标（km）	用户可靠性（*100%）
1	电源	72.7884	75.8243	无
2	负荷	84.15600875	73.42296912	0.950220486
3	负荷	95.73840226	26.53220362	0.815430414
4	负荷	92.45808952	22.37704047	0.806796731
5	负荷	80.05586563	74.58474843	0.966670709
6	负荷	81.31128136	38.33063186	0.849458364
7	负荷	61.72792323	57.54948597	0.931555875
8	负荷	80.20914406	98.91449097	0.882668498
9	负荷	78.37364801	53.41375679	0.893492845
10	负荷	88.53594509	89.90048989	0.918727983
11	负荷	61.64434851	93.96610102	0.842942251
12	负荷	98.4349417	94.5579189	0.898626333
13	负荷	67.66446784	98.83022623	0.856191465
14	负荷	60.2638218	75.05200559	0.954545737
15	负荷	58.35331743	55.17925149	0.91741429
16	负荷	58.35706188	51.181992	0.910081507
17	负荷	99.61561113	35.4534305	0.830543342
18	负荷	97.12588152	34.64487613	0.834913125
19	负荷	88.65438618	45.4694865	0.861666919
20	负荷	72.61044317	78.2872073	0.983165336
21	负荷	84.3213338	92.23319978	0.903528424
22	分叉点	80.05497168	74.5861947	无
23	分叉点	82.25011159	75.37755193	无
24	分叉点	88.53212733	89.8994971	无
25	分叉点	84.32147358	92.23350345	无
26	分叉点	80.00236904	98.1503681	无
27	分叉点	67.6653737	98.82849116	无

28	分叉点	68.24322022	72.93540567	无
29	分叉点	61.77881038	57.57414137	无
30	分叉点	78.37338837	53.41342262	无
31	分叉点	87.51619344	44.35956039	无
32	分叉点	97.12631284	34.64479306	无
33	分叉点	95.73844191	26.53223422	无
34	分叉点	58.35342675	55.17884685	无
35	电源	27.862	27.2309	无
36	负荷	14.21871593	2.513498571	0.913798014
37	负荷	42.11122538	18.41002894	0.939409416
38	负荷	57.10258728	17.68550576	0.895453142
39	负荷	37.35638076	8.750034958	0.919196375
40	负荷	64.01165482	18.06168878	0.863268536
41	负荷	4.505110747	72.31734792	0.810508286
42	负荷	53.00517048	27.50697558	0.921349724
43	负荷	24.86289597	45.16387705	0.93789876
44	负荷	53.56641907	8.70772199	0.878172476
45	负荷	1.817753364	68.38386137	0.802786492
46	负荷	62.59376261	13.78689924	0.877217716
47	负荷	21.78015937	18.21410759	0.95787983
48	负荷	4.181986397	10.69416586	0.91372766
49	负荷	35.4455731	41.06290901	0.945144979
50	负荷	66.23818604	24.41652868	0.851642866
51	负荷	52.78468304	41.15935134	0.877400084
52	负荷	8.2592727	71.95701349	0.822417099
53	负荷	41.3427289	21.77320684	0.952600275
54	负荷	12.56545874	30.89145936	0.941045796
55	负荷	69.3787615	0.980225226	0.84465135
56	负荷	60.49906419	38.72454315	0.890258693
57	负荷	72.57752675	37.03626865	0.863039983
58	负荷	34.74376456	66.06168245	0.890692718
59	负荷	38.38686011	62.73465024	0.901893019
60	负荷	2.164981463	91.05699885	0.774243156
61	负荷	22.7712826	80.44495836	0.839635534
62	负荷	98.61042419	2.999195027	0.798892972
63	负荷	6.69462584	93.93983619	0.766735674
64	负荷	76.68313872	33.66992644	0.853880789
65	负荷	29.55072508	68.01783712	0.874654371
66	分叉点	21.70816726	22.14683133	无
67	分叉点	11.57488137	9.482466249	无
68	分叉点	31.54184544	41.3031267	无
69	分叉点	36.94821239	62.51424206	无
70	分叉点	34.73961485	66.05488438	无

71	分叉点	29.54699677	68.01785098	无
72	分叉点	21.265507	74.15064934	无
73	分叉点	8.259905183	71.95710762	无
74	分叉点	4.506429801	72.31774309	无
75	分叉点	2.482383575	90.85796249	无
76	分叉点	41.33716851	21.77547378	无
77	分叉点	42.106376	18.40855239	无
78	分叉点	53.00733991	27.50262744	无
79	分叉点	57.10345091	17.68708313	无
80	分叉点	62.59039073	13.79575094	无
81	分叉点	64.01115718	18.06128091	无
82	分叉点	70.08798521	2.331152629	无
83	分叉点	60.09236306	38.17997884	无
84	分叉点	72.57509576	37.03395166	无

8.2.8 问题三、四的优化结果信息'algo_optim_curve.xlsx'：

优化轮数	联络线方案花费	最低用户用电可靠性
0	0	0.766735674
1	8302.628607	0.774243156
2	8302.628607	0.780277867
3	8302.628607	0.798892972
4	8302.628607	0.798892972
5	8302.628607	0.798892972
6	8302.628607	0.798892972
7	26044.77873	0.802786492
8	26044.77873	0.810508286
9	26191.6318	0.816215867
10	26728.96569	0.822417099
11	27329.40659	0.82820732
12	27967.7294	0.839635534
13	28698.97757	0.84465135
14	29485.83682	0.851642866
15	30379.26507	0.853200615
16	31335.1658	0.853880789
17	32354.45708	0.856634025
18	33441.5424	0.863039983
19	34650.39001	0.863268536
20	36003.64023	0.869120935
21	37453.59673	0.869346379
22	38998.44753	0.874654371
23	40671.4215	0.877217716
24	42469.58467	0.877400084

25	44429.84978	0.878172476
26	46560.28296	0.88081781
27	48847.91528	0.883409366
28	51286.96553	0.890258693
29	53893.41285	0.890692718
30	56704.19621	0.895453142
31	59712.15258	0.896976774
32	62922.38118	0.897745753
33	66349.19918	0.901759663
34	70007.97428	0.901893019
35	73982.53401	0.910893117
36	78228.74981	0.91372766
37	84510.14503	0.91372766
38	88560.95439	0.91372766
39	92915.18159	0.91372766
40	97579.30086	0.91372766
41	102577.4109	0.91372766

8.3 附录代码

8.3.1 问题一代码

gen_data.m（获取数据集）

```
clear

% 在100*100（km）的区域内随机均匀生成负荷点
% n = 25;
% P_x0 = zeros(1, n);
% P_y0 = zeros(1, n);
% for i = 1 : n
%     P_x0(i) = 100 * rand;
%     P_y0(i) = 100 * rand;
% end
% scatter(P_x0, P_y0, 10, 'filled');
% save('P_xy2.mat', 'P_x0', 'P_y0')

% 节点需求区间为[20,100]
n = 50;
dem = zeros(1, 50);
for i = 1 : n
    dem(i) = rand * 100;
end
```

constructive_algorithm.m（构造性算法）

```
% 构造性算法实现分叉点构造、线路选型
% 输入：G——最小生成树的邻接矩阵
% (P_x0, P_y0)——负荷坐标
% 输出：G_2——配电网拓扑的邻接矩阵
% (P_i,P_j)——所有节点的坐标
function [G_2, P_i, P_j] = constructive_algorithm(G, P_x0, P_y0)
    global G_1  % 最小生成树的邻接矩阵
    global G_2  % 配电网拓扑的邻接矩阵
    global P_i  % 所有节点的坐标
    global P_j
    global n1   % 负荷数+1
    global n2   % 分叉点数+1
    global cnt  % 计数器，为分叉点编号
    global deg  % 点度数
    global pair % 分叉点与负荷的映射关系
    global Load % 子树的负荷数

    G_1 = G;
    deg = zeros(1, n1);
    n2 = cnt_bran_node();      % 计数分叉点

    % 节点总数为(n1+n2-1)
    G_2 = zeros(n1 + n2 - 1);
    P_i = zeros(1, n1 + n2 - 1);
    P_j = zeros(1, n1 + n2 - 1);
    P_i(1 : n1) = P_x0(1 : n1);
    P_j(1 : n1) = P_y0(1 : n1);

    cnt = n1;
    pair = zeros(1, n1);

    for i = 1 : n1
        for j = i + 1 : n1
            G_2(i, j) = G_1(i, j);
        end
    end
    G_2 = G_2 + G_2';

    % 构造分叉点
    construct_cross_node();
    Load = zeros(1, n1 + n2 - 1);
```

```
    % 通过负荷数确定线的类型
    judge_line_type();
    % 假设与电源直接相连的为主线
    set_main_line();

%     % 画图
%     figure
%     G_n = graph(G_2);
%     col = zeros(n1 + n2 - 1, 3);
%     for i = 1 : n1
%         col(i, :) = [0 0.4470 0.7410];
%     end
%     for i = n1 + 1 : n1 + n2 -1
%         col(i, :) = [1 0 0];
%     end
%     h = plot(G_n, 'NodeColor', col);  % 返回一个 Graphplot 对象
%     % 设置节点位置
%     h.XData = P_i;
%     h.YData = P_j;
%     % 设置节点属性
%
%     % h.NodeCData = col;
%     % 设置线型
%     h.LineWidth = 1.5 * G_n.Edges.Weight;
end

% 计数度数大于 1 的点
function cnt = cnt_bran_node()
    global G_1
    global n1
    global deg
    for i = 1 : n1
        deg(i) = sum(G_1(i, :));
    end
    bran_node = find(deg(:) > 1);
    cnt = size(bran_node, 1);
end

function [] = construct_cross_node()
    dfs(1, 0);
end

function [] = judge_line_type()
    dfs1(1, 0);
```

```
end

function [] = dfs(u, fa)
    global G_1
    global G_2
    global n1
    global cnt
    global P_i
    global P_j
    global deg
    global pair

    if deg(u) > 1 && u ~= 1
        % 构造一个新分叉点
        cnt = cnt + 1;
        P_i(cnt) = P_i(u) + rand * 5;
        P_j(cnt) = P_j(u) + rand * 5;
        G_2(cnt, u) = 1;
        G_2(u, cnt) = 1;
        pair(u) = cnt;
        for v = 1 : n1
            if G_1(u, v) == 1
                if pair(v) ~= 0
                    G_2(u, pair(v)) = 0;
                    G_2(pair(v), u) = 0;
                    G_2(cnt, pair(v)) = 1;
                    G_2(pair(v), cnt) = 1;
                else
                    G_2(u, v) = 0;
                    G_2(v, u) = 0;
                    G_2(cnt, v) = 1;
                    G_2(v, cnt) = 1;
                end
            end
        end
    end

    for v = 1 : n1
        if G_1(u, v) == 1 && v ~= fa
            dfs(v, u);
        end
    end
end
```

```
function [] = dfs1(u, fa)
    global Load
    global G_2
    global n1
    global n2
    if u <= n1 && u ~= 1
        Load(u) = 1;
    end
    for v = 1 : n1 + n2 - 1
        if G_2(u, v) >= 1 && v ~= fa
            dfs1(v, u);
            if Load(v) > 2
                G_2(u, v) = 2;
                G_2(v, u) = 2;
            end
            Load(u) = Load(u) + Load(v);
        end
    end
end


function set_main_line()
    global G_2
    global n1
    global n2
    for v = 1 : n1 + n2 - 1
        if G_2(1, v) >= 1
            G_2(1, v) = 3;
            G_2(v, 1) = 3;
        end
    end
end
```

get_MST.m（获取最小生成树）

```
% 获取最小生成树
% 输入：(P_x0,P_y0)——所有节点（包括电源）的坐标
% 输出：G——最小生成树的邻接矩阵
function G = get_MST(P_x0, P_y0)
    global fa          % 并查集
    global W           % 欧几里得距离
    global P_ij         % W 对应的线段端点
    global G           % 最小生成树的邻接矩阵
```

```
    global e            % 最小生成树的边集
    global n1           % 负荷数+1

    n1 = size(P_x0, 2);
    e = zeros(2, n1 - 1);
    G = zeros(n1);
    P_ij = zeros(2, n1 * (n1 - 1) / 2);
    W = zeros(1, n1 * (n1 - 1) / 2);
    fa = zeros(1, n1);
    for i = 1 : n1
        fa(i) = i;
    end

    cnt0 = 1;
    for i = 1 : n1
        for j = i + 1 : n1
            P_ij(1, cnt0) = i;
            P_ij(2, cnt0) = j;
            W(cnt0) = sqrt((P_x0(i) - P_x0(j))^2 + (P_y0(i) - P_y0(j))^2);
            cnt0 = cnt0 + 1;
        end
    end
    % kruskal 求解最小生成树
    get_mst();
    G = G + G';

%     % 画图
%     figure
%     G_2 = graph(G);
%     % G_2 = graph(e(1, :), e(2, :));
%     h = plot(G_2);
%     h.XData = P_x0;
%     h.YData = P_y0;
end

function [] = get_mst()
    global G
    global P_ij
    global W
    global fa
    global n1
    global e
    [W, iw] = sort(W);
    P_ij(:, :) = P_ij(:, iw);
```

```
    cnt = 0;
    for i = 1 : n1 * (n1 - 1) / 2
        u = P_ij(1, i);
        v = P_ij(2, i);
        fx = Find(u);
        fy = Find(v);
        if fx ~= fy
            fa(fx) = fy;
            G(u, v) = 1;
            cnt = cnt + 1;
            e(1, cnt) = u;
            e(2, cnt) = v;
            if cnt >= n1 - 1
                break;
            end
        end
    end
end

function res = Find(x)
    global fa
    if x == fa(x)
        res = x;
    else
        fa(x) = Find(fa(x));
        res = fa(x);
    end
end
```

SSDN_model.m（单供配电网模型）

```
% Single supply and distribution network model
% 输入：(P_x0, P_y0)——负荷以及电源坐标，电源为 1 号节点
% 输出：G_1——配电网络拓扑图
% (P_x, P_y)——网络中节点坐标
function [G_3, P_x, P_y, fval, r] = SSDN_model(P_x0, P_y0)
    % global 将变量放入全局工作区，其他部分可同样通过 global 获取该变量
    global P_x      % 所有节点坐标
    global P_y
    global G_3      % 配电网拓扑的邻接矩阵
    global e_3      % 配电网拓扑的边集
    global cnt_e
    global n        % 节点总数
    global n1       % 负荷数+1
```

```
    global cost_line    % 线路单价
    global cost_switch  % 开关单价
    global r            % 用户用电可靠性

    G = get_MST(P_x0, P_y0);
    [G_3, P_x, P_y] = constructive_algorithm(G, P_x0, P_y0);    % 增加构
造算法构造的分叉点

    n = size(P_x, 2);
    % 预处理获取边集，优化算法效率
    e_3 = zeros(n * (n - 1) / 2, 3);
    cnt_e = 0;
    for i = 1 : n
        for j = i + 1 : n
            if G_3(i, j) >= 1
                cnt_e = cnt_e + 1;
                e_3(cnt_e, 1) = i;
                e_3(cnt_e, 2) = j;
                e_3(cnt_e, 3) = G_3(i, j);
            end
        end
    end
%     cnt_e

    cost_line = [0 188.6 239.4 325.7];
    cost_switch = [2.6 56.8];


    % 求解器
    % 无约束极值问题
    [x, fval] = fminunc(@f1, rand(2 * (n - n1), 1) - 0.5);
    fval0 = f1(zeros(2 * (n - n1), 1));
    % fval,x

    for i = n1 + 1 : n
        P_x(i) = P_x(i) + x(2 * (i - n1) - 1);
        P_y(i) = P_y(i) + x(2 * (i - n1));
    end

    r = zeros(n, 1);
    clac_power_reliability();

%     % 可视化
%     figure
```

```
%     G_n = graph(G_3);
%     col = zeros(n, 3);
%     for i = 1 : n1
%         col(i, :) = [0 0.4470 0.7410];
%     end
%     for i = n1 + 1 : n
%         col(i, :) = [1 0 0];
%     end
%     h = plot(G_n, 'NodeColor', col);
%     h.XData = P_x;
%     h.YData = P_y;
%     h.LineWidth = 1.5 * G_n.Edges.Weight;
end

% function cost = f1(x)
%     global n
%     global n1
%     delta_x = zeros(n - n1);
%     delta_y = zeros(n - n1);
%     for i = 1 : n - n1
%         delta_x(i) = x(2 * i - 1);
%         delta_y(i) = x(2 * i);
%     end
%     cost = f(delta_x, delta_y);
% end

% 评价函数
function cost = f1(x)
    global P_x
    global P_y
    global e_3
    global cnt_e
    global n
    global n1
    global cost_line
    cost = 0;
    delta_x = zeros(n - n1);
    delta_y = zeros(n - n1);
    for i = 1 : n - n1
        delta_x(i) = x(2 * i - 1);
        delta_y(i) = x(2 * i);
    end
    for i = n1 + 1 : n
        P_x(i) = P_x(i) + delta_x(i - n1);
```

```
        P_y(i) = P_y(i) + delta_y(i - n1);
    end
%     for i = 1 : n
%         for j = i + 1 : n
%             if G_3(i, j) >= 1
%                 dis_ij = sqrt((P_x(i) - P_x(j))^2 + (P_y(i) - P_y(j))^2);
%                 cost = cost + dis_ij * cost_line(G_3(i, j) + 1);
%             end
%         end
%     end
    for i = 1 : cnt_e
        dis_ij = sqrt((P_x(e_3(i, 1)) - P_x(e_3(i, 2)))^2 + (P_y(e_3(i,
1)) - P_y(e_3(i, 2)))^2);
        cost = cost + dis_ij * cost_line(e_3(i, 3) + 1);
    end
    for i = n1 + 1 : n
        P_x(i) = P_x(i) - delta_x(i - n1);
        P_y(i) = P_y(i) - delta_y(i - n1);
    end
end

% 计算用户用电可靠性
function [] = clac_power_reliability()
    global r
    global G_3
    global n
    r(1) = 1 - 0.005;
    for v = 1 : n
        if G_3(1, v) >= 1
            dfs2(v, 1);
        end
    end
end

function [] = dfs2(u, fa)
    global r
    global G_3
    global n
    global P_x
    global P_y
    dis_ufa = sqrt((P_x(u) - P_x(fa))^2 + (P_y(u) - P_y(fa))^2);
    r(u) = r(fa) * (1 - 0.005) * (1 - 0.002) * (1 - 0.002 * dis_ufa);
    for v = 1 : n
        if G_3(u, v) >= 1 && v ~= fa
```

```
            dfs2(v, u);
        end
    end
end
```

main_t1.m（问题一主程序）

```
clear all
close all

load('P_xy.mat');
n1 = size(P_x0, 2);
P_x0(2 : n1 + 1) = P_x0(:);
P_y0(2 : n1 + 1) = P_y0(:);
% 电源位于区域的中心,(rand*10-5)
P_x0(1) = 50 + 3.0191;
P_y0(1) = 50 + (-0.0259);

[G_3, P_x, P_y, fval, r] = SSDN_model(P_x0, P_y0);
fval
save('t1_best', 'G_3', 'P_x', 'P_y', 'fval', 'r');
```

8.3.2 问题二代码

main_t2.m（问题二主程序）

```
clear all
close all

global k
global x_M
global y_M
global x_T1
global y_T1
global x_T2
global y_T2
global n_leaf
global P_x0
global P_y0
global P1
global P2
global P3
global cnt_P1
global cnt_P2
global cnt_P3
```

```
load('P_xy.mat');
% 生成两个独立电源,(rand*10)
x_K1 = 75 + (-2.2116);
y_K1 = 75 + 0.8243;
x_K2 = 25 + 2.8620;
y_K2 = 25 + 2.2309;


n_leaf = size(P_x0, 2);
P1 = zeros(1, n_leaf);
P2 = zeros(1, n_leaf);
P3 = zeros(1, n_leaf);

k = (y_K2 - y_K1) / (x_K2 - x_K1);
x_M = (x_K1 + x_K2) / 2;
y_M = (y_K1 + y_K2) / 2;

t = 0;
set_t(t);

% 二分法确定 t 值
alpha = 10;
L = 0; R = 100;
ret = 0;
while ret ~= alpha
    t = (L + R) / 2;
    ret = check(t);
    if ret > alpha
        R = t;
    elseif ret < alpha
        L = t;
    end
end
% t, cnt_P1, cnt_P2, cnt_P3
% P1, P2, P3
% t

% % 可视化
hold on
plot(P_x0, P_y0, 'ro')
plot(P_x0(P3(1 : cnt_P3)), P_y0(P3(1 : cnt_P3)), 'go')
plot(x_K1, y_K1, 'bo')
plot(x_K2, y_K2, 'bo')
```

```
plot(x_M, y_M, 'bo')
plot([x_K1, x_K2], [y_K1, y_K2], 'b');
plot([x_T1 - 75, x_T1 + 75], [y_T1 + 1 / k * 75, y_T1 - 1 / k * 75], 'b')
plot([x_T2 - 75, x_T2 + 75], [y_T2 + 1 / k * 75, y_T2 - 1 / k * 75], 'b')

% 结合问题一模型求解最优二分类
cost_best = inf;
for i = 0 : bitshift(2, alpha - 1) - 1
    s1 = zeros(1, alpha);
    s2 = zeros(1, alpha);
    cnt_s1 = 0; cnt_s2 = 0;
    for j = 0 : alpha - 1
        if bitand(bitshift(i, -j), 1) == 1
            cnt_s1 = cnt_s1 + 1;
            s1(cnt_s1) = P3(j + 1);
        else
            cnt_s2 = cnt_s2 + 1;
            s2(cnt_s2) = P3(j + 1);
        end
    end
    P1(cnt_P1 + 1 : cnt_P1 + cnt_s1) = s1(1 : cnt_s1);
    P2(cnt_P2 + 1 : cnt_P2 + cnt_s2) = s2(1 : cnt_s2);
    P1 = int32(P1);
    P2 = int32(P2);
    x_s1(1) = x_K1; y_s1(1) = y_K1;
    x_s2(1) = x_K2; y_s2(1) = y_K2;
    x_s1(2 : cnt_P1 + cnt_s1 + 1) = P_x0(P1(1 : cnt_P1 + cnt_s1));
    y_s1(2 : cnt_P1 + cnt_s1 + 1) = P_y0(P1(1 : cnt_P1 + cnt_s1));
    x_s2(2 : cnt_P2 + cnt_s2 + 1) = P_x0(P2(1 : cnt_P2 + cnt_s2));
    y_s2(2 : cnt_P2 + cnt_s2 + 1) = P_y0(P2(1 : cnt_P2 + cnt_s2));
    [G_3_s1, P_x_s1, P_y_s1, fval_s1, r_s1] = SSDN_model(x_s1, y_s1);
    [G_3_s2, P_x_s2, P_y_s2, fval_s2, r_s2] = SSDN_model(x_s2, y_s2);
    disp(strcat("第", int2str(i + 1), "个状态花费为", num2str(fval_s1 +
fval_s2), ",当前最优为", num2str(cost_best)))
    % 更新最优解
    if fval_s1 + fval_s2 < cost_best
        cost_best = fval_s1 + fval_s2;
        G_3_s1_best = G_3_s1;
        G_3_s2_best = G_3_s2;
        P_x_s1_best = P_x_s1; P_y_s1_best = P_y_s1;
        P_x_s2_best = P_x_s2; P_y_s2_best = P_y_s2;
        r_s1_best = r_s1;
        r_s2_best = r_s2;
    end
```

```
end
save('t2_best', 'cost_best', 'G_3_s1_best', 'G_3_s2_best',
'P_x_s1_best', 'P_x_s2_best', 'P_y_s1_best', 'P_y_s2_best',
'r_s1_best', 'r_s2_best');

function ret = check(t)
    global n_leaf
    global P_x0
    global P_y0
    global P1
    global P2
    global P3
    global cnt_P1
    global cnt_P2
    global cnt_P3
    global x_T1
    global y_T1
    global x_T2
    global y_T2
    global k
    cnt_P1 = 0; cnt_P2 = 0; cnt_P3 = 0;
    P1(:) = 0; P2(:) = 0; P3(:) = 0;
    set_t(t);
    for i = 1 : n_leaf
        if P_y0(i) - y_T1 + 1 / k * (P_x0(i) - x_T1) > 0 % f_1(P_x0(i),
P_y0(i)) > 0
            cnt_P1 = cnt_P1 + 1;
            P1(cnt_P1) = i;
        elseif P_y0(i) - y_T2 + 1 / k * (P_x0(i) - x_T2) < 0 % f_2(P_x0(i),
P_y0(i)) < 0
            cnt_P2 = cnt_P2 + 1;
            P2(cnt_P2) = i;
        else % if f_1(P_x0(i), P_y0(i)) * f_2(P_x0(i), P_y0(i)) < 0
            cnt_P3 = cnt_P3 + 1;
            P3(cnt_P3) = i;
        end
    end
    ret = cnt_P3;
end

function [] = set_t(t)
    global k
    global x_M
    global y_M
```

```
    global x_T1
    global x_T2
    global y_T1
    global y_T2
    sign = 1;
    if k < 0
        sign = -1;
    end
    % 保证直线 l_1 在上方
    x_T1 = x_M + sign * t / sqrt(k ^ 2 + 1);
    y_T1 = y_M + sign * t * k / sqrt(k ^ 2 + 1);
    x_T2 = x_M - sign * t / sqrt(k ^ 2 + 1);
    y_T2 = y_M - sign * t * k / sqrt(k ^ 2 + 1);
end

% function res = f_1(x, y)
%     global x_T1
%     global y_T1
%     global k
%     res = y - y_T1 + 1 / k * (x - x_T1);
% end
%
% function res = f_2(x, y)
%     global x_T2
%     global y_T2
%     global k
%     res = y - y_T2 + 1 / k * (x - x_T2);
% end
```

8.3.3 问题三、四代码

main_t3_t4.m（问题三、四主程序）

```
clear all



global G
global p
global r
global r_std
global r_outer
global ncp
global fa_
global is_optimized
global cost_add
```

```
global cost_main_line
global cost_switch
global expand_price
global pow
global P

load('t3_input.mat');

cost_main_line = 325.7;
cost_switch = 56.8;
expand_price = 10;
cost_add = 0;

r_std = r;
cnt_cont = 0;
cont_line = zeros(100, 2);

clac_basics(1);
clac_basics(2);

% 计算网络负荷需求之和
P = zeros(2, 1);
for i = 1 : 2
    for j = 1 : size(pow{i}, 1)
        P(i) = P(i) + pow{i}(j) * r{i}(j);
    end
end

for t = 1 : 100
    cost_add = 0;
    clac_optimization(1);
    clac_optimization(2);

    for i = 1 : cnt_cont
        u = cont_line(i, 1); v = cont_line(i, 2);
        dis_uv = sqrt((p{1}(u, 1) - p{2}(v, 1))^2 + (p{1}(u, 2) - p{2}(v,
2))^2);
        cost_add = cost_add + cost_main_line * dis_uv + cost_switch;
    end

    [min_r(1), m(1)] = min(r{1});
    [min_r(2), m(2)] = min(r{2});
    disp(strcat('第', num2str(t), '轮方案设计花费为', num2str(cost_add), ',
此时最低用电可靠性为', num2str(min(min_r))))
```

```
    X(t) = cost_add;
    Y(t) = min(min_r);

    % 1.两最低点均已优化，依据贪心设置原则无法继续优化，算法结束
    if is_optimized{1}(m(1)) + is_optimized{2}(m(2)) > 0
        break
    end
    % 2.单一最低点已优化，继续优化另一网络
    % todo
    % 3.最低点均未优化
    % 3.1 存在联络线，增加拓展供电方案
    if ncp{1}(m(1)) + ncp{2}(m(2)) > 0
        if ncp{1}(m(1)) > 0
            is_optimized{1}(m(1)) = 1;
        end
        if ncp{2}(m(2)) > 0
            is_optimized{2}(m(2)) = 1;
        end
        continue;
    end
    % 3.2 调整联络线

    % 搜索最低成本的联络线方案
    disp(m)
    if fa_{1}(m(1)) == 1 || fa_{2}(m(2)) == 1
        break
    end
    to_select1 = get_ancestors(m(1), 1);
    to_select2 = get_ancestors(m(2), 2);
    sz1 = size(to_select1, 2);
    sz2 = size(to_select2, 2);
    to_select_edge = zeros(sz1 * sz2, 3);
    for i = 1 : sz1
        for j = 1 : sz2
            u = to_select1(i); v = to_select2(j);
            dis_ij = sqrt((p{1}(u, 1) - p{2}(v, 1))^2 + (p{1}(u, 2) - p{2}(v,
2))^2);
            to_select_edge((i - 1) * sz2 + j, :) = [u, v, dis_ij];
        end
    end
    [~, one] = min(to_select_edge(:, 3));
    cont_1 = to_select_edge(one, 1); cont_2 = to_select_edge(one, 2);
%     cont_1 = fa_{1}(m(1)); cont_2 = fa_{2}(m(2));
```

```
    % 3.2.1 联络点上跳
    kid_ncp = find_kid_ncp(cont_1, fa_{1}(cont_1), 1);
    if kid_ncp > 0
        tmp = find(cont_line(1 : cnt_cont, 1) == kid_ncp);
        disp(strcat('jump in tree A: ', num2str(cont_line(tmp, 1)), '->',
num2str(cont_1)));
        cont_line(tmp, 1) = cont_1;
        c2 = cont_line(tmp, 2);
        disp(strcat('get contact line: ', num2str(cont_1), '-',
num2str(c2)));
        dis_12 = sqrt((p{1}(cont_1, 1) - p{2}(c2, 1))^2 + (p{1}(cont_1,
2) - p{2}(c2, 2))^2);
        r_outer{1}(cont_1) = r{2}(c2) * (1 - 0.002) * (1 - 0.002 * dis_12);
        r_outer{2}(c2) = r{1}(cont_1) * (1 - 0.002) * (1 - 0.002 * dis_12);
        set_ncp(cont_1, fa_{1}(cont_1), 1, cont_1);
        is_optimized{1}(m(1)) = 1;
        continue;
    end
    kid_ncp = find_kid_ncp(cont_2, fa_{2}(cont_2), 2);
    if kid_ncp > 0
        tmp = find(cont_line(1 : cnt_cont, 2) == kid_ncp);
        disp(strcat('jump in tree B: ', num2str(cont_line(tmp, 2)), '->',
num2str(cont_2)));
        cont_line(tmp, 2) = cont_2;
        c1 = cont_line(tmp, 1);
        disp(strcat('get contact line: ', num2str(c1), '-',
num2str(cont_2)));
        dis_12 = sqrt((p{1}(c1, 1) - p{2}(cont_2, 1))^2 + (p{1}(c1, 2) -
p{2}(cont_2, 2))^2);
        r_outer{1}(c1) = r{2}(cont_2) * (1 - 0.002) * (1 - 0.002 * dis_12);
        r_outer{2}(cont_2) = r{1}(c1) * (1 - 0.002) * (1 - 0.002 * dis_12);
        set_ncp(cont_2, fa_{2}(cont_2), 2, cont_2);
        is_optimized{2}(m(2)) = 1;
        continue;
    end

    % 3.2.2 增加联络线
    set_ncp(cont_1, fa_{1}(cont_1), 1, cont_1);
    set_ncp(cont_2, fa_{2}(cont_2), 2, cont_2);
    cnt_cont = cnt_cont + 1;
    cont_line(cnt_cont, :) = [cont_1 cont_2];
    disp(strcat('get contact line: ', num2str(cont_1), '-',
num2str(cont_2)));
    dis_12 = sqrt((p{1}(cont_1, 1) - p{2}(cont_2, 1))^2 + (p{1}(cont_1,
```

```
2) - p{2}(cont_2, 2))^2);
    r_outer{1}(cont_1) = r{2}(cont_2) * (1 - 0.002) * (1 - 0.002 * dis_12);
    r_outer{2}(cont_2) = r{1}(cont_1) * (1 - 0.002) * (1 - 0.002 * dis_12);
    is_optimized{1}(m(1)) = 1; is_optimized{2}(m(2)) = 1;
end

save('cont_line', 'cont_line', 'cnt_cont');
X = X(:);
Y = Y(:);
save('XY', 'X', 'Y');

% 搜索子树中的联络点，至多存在一个
function kid_ncp = find_kid_ncp(u, fa, k)
    global G
    global ncp
    n = size(G{k}, 1);
    kid_ncp = 0;
    for v = 1 : n
        if G{k}(u, v) >= 1 && v ~= fa
            if ncp{k}(v) > 0
                kid_ncp = v;
                return;
            end
            kid_ncp = max(kid_ncp, find_kid_ncp(v, u, k));
        end
    end
end

% 获取祖先分叉点
function ancestors = get_ancestors(u, k)
    global fa_
    cnt = 0;
    fa = fa_{k}(u);
    while fa ~= 1
        cnt = cnt + 1;
        ancestors(cnt) = fa;
        u = fa;
        fa = fa_{k}(u);
    end
%     cnt = cnt + 1;
%     ancestors(cnt) = 1;
end

% 设置最近联络点
```

```
function [] = set_ncp(u, fa, k, c)
    global G
    global ncp
    n = size(G{k}, 1);
    ncp{k}(u) = c;
    for v = 1 : n
        if G{k}(u, v) >= 1 && v ~= fa
            set_ncp(v, u, k, c);
        end
    end
end

% 计算基础用电可靠性
function [] = clac_basics(k)
    global G
    global p
    global r
    n = size(G{k}, 1);
    r{k}(1) = 1 - 0.005;
    for v = 1 : n
        if G{k}(1, v) >= 1
            dfs(v, 1, k);
        end
    end
end

function [] = dfs(u, fa, k)
    global G
    global p
    global r
    global fa_

    n = size(G{k}, 1);
    fa_{k}(u) = fa;
    dis_ufa = sqrt((p{k}(u, 1) - p{k}(fa, 1))^2 + (p{k}(u, 2) - p{k}(fa,
2))^2);
    r{k}(u) = r{k}(fa) * (1 - 0.005) * (1 - 0.002) * (1 - 0.002 * dis_ufa);
    for v = 1 : n
        if G{k}(u, v) >= 1 && v ~= fa
            dfs(v, u, k);
        end
    end
end
```

```
% 计算优化后用电可靠性
function [] = clac_optimization(k)
    global G
    global r
    global r_std
    global r_outer
    global ncp
    global is_optimized
    global cost_add
    global pow
    global expand_price
    global P

    n = size(G{k}, 1);
    pow_add = 0;
    for i = 1 : n
        if is_optimized{k}(i) > 0
            r_com = r{k}(i) / r_std{k}(ncp{k}(i));  % 本地线路和拓展线路的公
共部分
            r{k}(i) = (r_std{k}(ncp{k}(i)) + r_outer{k}(ncp{k}(i)) -
r_std{k}(ncp{k}(i)) * r_outer{k}(ncp{k}(i))) * r_com * (1 - 0.005);
            pow_add = pow_add + (r{k}(i) - r_std{k}(i)) * pow{k}(i);
        end
    end
%     pow_add, 0.1 * P(3 - k)
    pow_add = max([pow_add - 0.1 * P(3 - k), 0]);
    if pow_add > 0.55 * P(3 - k)
        disp('bomb!')
    end
    cost_add = cost_add + pow_add * expand_price;
end
```