2018 年数学建模竞赛复赛试题

B 题-141 组

李悦 16030120
邢鸿飞 16030119
刘懿祺 16030108

城市交通学院
北京工业大学
邮箱：940482331@qq.com
联系方式：18810207077

基于可变邻域的模拟退火算法优化模型

摘要

本文首先用数学符号表达了题目中的约束条件。通过引入δ（i,j,k,r）——这个由站点i、站点j、级跳、线路编号四个变量决定的0，1变量，将题目中的各个约束进行了清晰的数学表达。然后又引入一个在输入与输出矩阵之间架起桥梁的中间矩阵，对问题进行优化求解。该中间矩阵将每个站点视作一列属性，每个属性表达约束条件中的一条或多条约束。该中间矩阵由5行n列组成，第一行为站点类型，第二行为该站点直接连接站点位置，第三行为该站点连接站点个数，第四行为该站点的级跳数，第五行为该站点对应的宿主站。

优化求解则以最小化平均建站成本与平均系统损耗为目标，运用基于可变邻域的模拟退火算法来完成。运用模拟退火首先需要生成初始解。本文中通过按随机顺序选中宿主站，然后将宿主站附近满足条件的站点设置为该宿主站的子站来生成初始解。在获得初始解后，可变邻域的设计可以分为两步。第一步是将孤立的宿主站变为子站连接到附近的子站，第二步是将随机选中的一个子站连接到满足约束条件的其它子站，在连接的时候为确保成本尽量小，该子站只连接比自己离宿主站近的子站。为了防止陷入局部最优解，本模型设计了一个较大的扰动。即在模拟退火算法温度变化时，对于宿主站选取的访问顺序进行随机调整，从而进入新的邻域进行优化搜索。

在得到由中间矩阵表示的优化解后，通过对中间矩阵信息的提取便可以输出Graph.csv与Posi.csv。由于整个过程用矩阵表示，结果的合理性判断较为困难，所以，为了验证解的合理性，本文设计了结果可视化函数。该函数可将宿主站、子站以及它们之间的连接关系用图像表达。

关键词：可变邻域，模拟退火算法，成本优化，可视化

一、问题重述

基站的建设需要考虑光线部署成本、ROI（投资回报率）等因素，而城市与农村的具体情况又各不相同。Relay 无线回传方案利用 FDD LTE 或 TDDLTE 制式承载来为站点回传，可以同时解决城区与农网的传统传输方式下的不可达问题。本题将以上背景问题抽象成无线回传拓扑规划问题，即需在已知给定一个地区候选站点位置分布的情况下，根据站点间的相互位置、站点间拓扑关系限制等条件，在满足一定回传质量（本题仅根据宿主站与子站的距离是否满足某门限来判断是否满足最低回传质量要求）的前提下，设计成本最优的部站方案。具体包括：

1. **输入**：每个地区内的所有站点列表，包括站点经纬度、站型，RuralStar 或蝴蝶站；各种站型的综合成本，包括宿主站、子站的综合成本以及卫星的设备成本。
2. **约束**：输出的拓扑关系应满足：首跳距离≤20km，之后每跳距离≤10km；站点包含 RuralStar 和蝴蝶站两种不同站型；其中，RuralStar 共包含 1 个扇区，蝴蝶站共包含 2 个扇区；若该站点为宿主站，则每个扇区第一级最大接入子站数 4，最大总接入子站数 6（为了简化问题，暂不考虑蝴蝶站的扇区覆盖方向）；宿主站之间采用微波连接，最大通信距离为 50KM；宿主站和子站以及子站之间采用无线回传连接；每个子站最多只能有 2 条无线回传连接；任意子站只能归属一个宿主站，到达所属宿主站有且只有一条通路，且该通路包含的跳数小于等于 3；任意宿主站都有且只有一颗卫星负责回传，成片连接的宿主站可共享同一颗卫星，但一颗卫星最多只能负担 8 个成片宿主站的回传数据；成片宿主站中，宿主站总数不设上限
3. **输出**：按输入数据中站点顺序，输出两个文件 1）Graph.csv 包含一个二维矩阵用以表示所有站点间的连接关系，0 表示没有连接关系，1 表示采用无线回传连接，2 表示采用微波连接；2）Posi.csv 包含一个一维数组表示站点类型，0 表示子站，1 表示宿主站。并将算法效率控制在 5 分钟以内，站点规模控制在 1000 站点左右。
4. **优化目标**：1）（最高优先级）使总体成本更低。总体成本=宿主站数量*宿主站成本+子站数量*子站成本+卫星数量*卫星成本；平均成本=总体成本/地区内站点总数。其中宿主站、子站、卫星成本分别为 10、5、50WUSD。2）使回传路径损耗更低。PL=32.5+20*lg（D）+20*lg（F），其中，PL 是路径损耗，D 是两个站点之间的距离，单位为 km，F 是发射频率，单位为 MHz，这里默认采用 900MHz。且系统平均损耗=所有无线回传连接的损耗之和/无线回传连接数。该路径损耗只考虑子站回传部分，宿主站之间采用微波传输，只需满足距离限制，不计算该损耗。

同时，所设计的算法还应该具有以下特点：算法收敛速度快、尽可能覆盖更多的站点。

二、问题分析

2.1 对约束的分析

分析给出的拓扑约束条件，我们将约束总体归为以下五类：

（1） 距离约束：包括首跳距离约束、之后每跳的距离约束、总跳数约束以及宿主站之间的距离约束。

（2） 分支约束：即每个子站的连接约束。

（3） 连接站点个数约束：即不同站型总共可连接的一级最大接入子站与最大总接入子站数的约束。

（4） 连接方式约束：包括宿主站之间连接、宿主站与子站之间连接及子站与子站之间的连接方式约束。

（5） 其他唯一约束：包括每个子站归属宿主站的个数唯一，到达所属宿主站的通路唯一以及单一卫星所能供给的成片连接宿主站的最大负担宿主站个数。

2.2 模型建立的基本思路

本模型以最小化平均建站成本与系统平均损耗为目标，在满足约束条件的可行解的可变邻域内进行优化搜索。首先随机生成一个n×2的坐标矩阵，用以存放n个站点的经纬度信息。其次，用题中所给的球面距离公式生成n个站点两两之间的n×n的距离矩阵。然后，根据已有的坐标矩阵中的n个站点的排列顺序，生成可行解。我们将可行解的各个要素存放在一个5×n的矩阵中，每一列代表一个站点的5个要素信息，从上到下分别为：

（1） 站点类型：0表示子站，1表示宿主站。

（2） 该站点连接点的站点序号：即该站点连接点位于坐标矩阵中的行数，如果该站点没有前驱连接点则输出-1。

（3） 连接的站点个数：当该站点为宿主站时输出归属于该站点的全部子站个数，当该站点为子站时输出与该子站直接相连的站点个数。

（4） 级跳数：若是子站则输出该子站位于所属宿主站的第几级跳数，若是宿主站则输出-1。

（5） 对应关系：若是子站则输出对应的宿主站序号，若是宿主站则输出-1。

例如：坐标矩阵中第一个站点为宿主站，连接了4个子站，则其解的表示为：

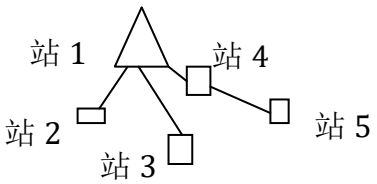


图1 引例示意图

站1 站2 站3 站4 站5

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -1 & 1 & 1 & 1 & 4 \\ 4 & 1 & 1 & 2 & 1 \\ -1 & 1 & 1 & 1 & 2 \\ -1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

2.3 优化模型的基本思路

我们对初始解做了三点优化：

（1） 由于题设中给出的宿主站成本高于子站，且增加宿主站时还要考虑卫星的成本，因此，我们设想将一些孤立的宿主站连接到其最近的子站上，使其成为二级或更低级的子站。

（2） 随机选定子站，使其脱离原本所属的宿主站而连接到符合约束关系的其他子站上。这种操作既可能导致成本的增加也可能导致成本的减少，由此，我们运用模拟退火的接受条件来判断其是否接受当前解。

（3） 由于初始解与站点的排列顺序有关，于是我们设置了较大的扰动，即在模拟退火算法温度变化时，将原来的位置进行重新随机排列，从而进入新的邻域进行优化搜索。

三、模型假设

1、假设影响回传质量的因素只与宿主站与子站的距离是否满足某门限有关，而与距离、地形阻挡、普通手机接入影响、ReBTS 干扰、相邻基站干扰等多种复杂因素无关。

2、假设路径损耗只考虑子站回传部分，宿主站之间采用微波传输，只需满足距离限制，不计算该损耗。

3、假设忽略无线回传中存在的 NLOS 影响，采用自由空间传播模型估计站点之间的路径损耗。

4、假设令 n 为 1000,即选取 1000 个站点。

四、符号说明

序号	*符号*	*符号说明*
1.	i	宿主站编号
2.	j	子站编号
3.	k	级跳数
4.	r	线路编号
5.	d	距离
6.	F	发射频率
7.	PL	路径损耗
8.	α	站点的经度
9.	β	站点的纬度
10.	R	地球半径
11.	Solution	中间矩阵

五、模型的建立与求解

5.1 输入的设定与选取

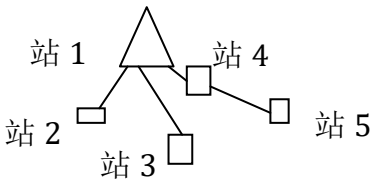
首先用 matlab 的 rand 函数随机生成东经 0-5 度，北纬 0-5 度的 1000 个站点坐标，形成一个 1000×2 的坐标矩阵。其次，运用题目所给的球面两点距离公式，其中设 A 点纬度β_1，经度α_1；B 点纬度β_2，经度α_2，R 为地球半径，取 6378km，则 A、B 点距离S为：

$$S = R \times \arccos\left[\cos\beta_1 \times \cos\beta_2 \times \cos(\alpha_1 - \alpha_2) + \sin\beta_1 \times \sin\beta_2\right]$$

可得出一个 1000×1000 的距离矩阵。然后，根据已有的坐标矩阵中的 1000 个站点的排列顺序，从第一个站点开始，先将其分配为宿主站，再在距离矩阵中寻找与其距离在 20KM 以内的站点，随机选取 4 个，分配它们为此宿主站的子站并作无线回传连接。再检索到坐标矩阵的第二个站点，若其已被分配为第一个站点的子站则再向下检索到第三个站点，若其未被分配，则将其分配为第二个宿主站，按照先前同样的方式寻找并连接相应的子站。以此类推可生成所有 1000 个站点的信息。同时，输入各种站型的综合成本，包括宿主站、子站的综合成本以及卫星的设备成本，其中宿主站、子站、卫星成本分别为 10、5、50WUSD。

5.2 约束的表示

为表达约束条件，本文中引入由 i,j,k,r 四个因素决定的 0，1 变量$\delta(i,j,k,r)(i \le n, j \le n, k \le 3, r \le 4)$表达站点之间的连接关系。即如果某子站 j 是宿主站 i 上第 r 条线路上的第 k 级，则$\delta(i,j,k,r) = 1$，否则$\delta(i,j,k,r) = 0$。



站1 站2 站3 站4 站5

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -1 & 1 & 1 & 1 & 4 \\ 4 & 1 & 1 & 2 & 1 \\ -1 & 1 & 1 & 1 & 2 \\ -1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

图2 引例示意图

如图所示，设从左到右依次为线路 1，线路 2，线路 3。因为子站 2 是宿主站 1 上第一条线路上的一级子站，所以$\delta(1,2,1,1) = 1$，当 j=2 确定遍历 i,k,r 时，只有这一种情况使δ变量为 1，其余情况均为零。这样就保证了子站 2 与基站 1 的唯一相连关系。

输出的拓扑关系应满足如下的限制条件：

1. 距离约束：$\begin{cases} \delta_1(i,j,1,r) * d_{i,j} \le 20 \\ \delta_1(i,j_1,2,r) * \delta_1(i,j_2,1,r) * d_{j_1,}d_{j_2} \le 10 \\ \delta_2(i_1,i_2) * d_{i_1,i_2} \le 50 \end{cases}$

2. 分支约束：$\sum_i \sum_k \delta_1(i,j_1,k,r) + \sum_{j_2} \sum_k \delta_1(j_1,j_2,k,r) \le 2$

3. 连接站点个数约束：（1）RuralStar 类型的站点：$\begin{cases} \sum_r \sum_j \delta_1(i,j,1,r) \le 4 \\ \sum_k \sum_r \sum_j \delta_1(i,j,k,r) \le 6 \end{cases}$

（2）蝴蝶类型的站点：$\begin{cases} \sum_r \sum_j \delta_1(i,j,1,r) \le 8 \\ \sum_k \sum_r \sum_j \delta_1(i,j,k,r) \le 12 \end{cases}$

4. 连接方式约束：

$$\delta_1(i,j,k,r)=\begin{cases}1,\ \text{表示宿主站与子站间存在无线回传连接}\\0,\ \text{表示宿主站与子站间不存在无线回传连接}\end{cases}$$

$$\delta_1(j_1,j_2,k,r)=\begin{cases}1,\ \text{表示子站与子站间存在无线回传连接}\\0,\ \text{表示子站与子站间不存在无线回传连接}\end{cases}$$

$$\delta_2(i_1,i_2)=\begin{cases}1,\ \text{表示宿主站与宿主站间存在微波连接}\\0,\ \text{表示宿主站与宿主站间不存在微波连接}\end{cases}$$

5. 卫星数约束：$N_{sate}=\lceil N_i/8\rceil$

6. 其他约束：

$$\begin{cases}\sum_k\sum_r\sum_i\delta_1(i,j,k,r)=1,\text{表示任意子站只能归属一个宿主站}\\k\le 3,\text{表示子站的级跳数不超过三}\\\sum_r\delta_1(i,j,k,r)=1,\ \text{表示子站到达所属宿主站有且只有 一条通路}\end{cases}$$

5.3 模型的建立与求解

通过对约束条件的表达，我们确立了中间矩阵的形态。即每一列第一行为 0，1 变量，0 表示子站，1 表示宿主站，该列的列数决定该站点在输入位置矩阵中的位置。即如果第 30 列的第一行为 0，则表示在输入的位置矩阵中的第 30 个坐标为子站。由此可以得知，列数与每列的第一行共同决定 i，j。每一列的第 2 行记录该站点的直接前驱，如果没有则记录为-1。本行决定了每个站点只能唯一确定地连接到 1 条线路上，即$\sum_r\delta_1(i,j,k,r)=1$。每一列的第三行表示该站点连接的站点个数，可以用来表达约束 2 与约束 3，对于宿主站 i 来说为$\sum_k\sum_r\sum_j\delta_1(i,j,k,r)$，对于子站$j_1$来说可以用$\sum_i\sum_k\delta_1(i,j_1,k,r)+\sum_{j_2}\sum_k\delta_1(j_1,j_2,k,r)$来表达。每一列第四行表达该站点的级跳 k，可以表达子站的级跳不超过 3 级，每一列的第五行表示子站对应的宿主站，表达了任意子站只能归属于一个宿主站的约束:$\sum_k\sum_r\sum_i\delta_1(i,j,k,r)=1$。

通过对问题分析，我们得知：可以从一个满足约束条件的可行解出发，通过不断地调整可行解，就可以生成更多地可行解。因此，可以在这些可行解的集合中搜索最优解。但是，如果单纯的优化搜索容易陷入局部最优解中，所以需要向模型中加入较大的扰动。综上所述本文采用基于可变邻域的模拟退火算法对问题进行求解。

5.3.1 距离矩阵的获取

本问题中，因为有多处距离约束，所以首先将输入的$n\times 2$维的位置矩阵location转换成$n\times n$维的距离矩阵Dist。在距离矩阵中$Dist(i,j)$表示任意两点i与j之间的距离，显然，i与j的距离等于j与i的距离。可用公式表示为

$Dist(i,j)=R\times\arccos\left[\cos\beta_i\times\cos\beta_j\times\cos(\alpha_i-\alpha_j)+\sin\beta_i\times\sin\beta_j\right]$。

$Dist(i,j)=Dist(j,i)$。

算法的时间复杂度为$O(n^2)$

5.3.2 初始解的生成

从第一个站点开始，先将其分配为宿主站，再在距离矩阵中寻找与其距离在 20KM 以内的站点，随机选取 4 或 8 个（由宿主站类型决定），分配它们为此宿主站的子站并作无线回传连接。再检索到坐标矩阵的第二个站点，若其已被分配为第一个站点的子站则再向下检索到第三个站点，若其未被分配，则将其分配为第二个宿

主站，按照先前同样的方式寻找并连接相应的子站。以此类推可生成一组初始解。

初始解只包含第一级跳的子站，不考虑二级、三级跳的子站。在生成初始解的过程中，遍历所有站点i，如果站点i未被分配，则分配站点i为宿主站，即$Solution(1,i)=1$。

然后搜索Dist矩阵的第i行，假设$Dist(i,j_1),Dist(i,j_2),Dist(i,j_3)\dots Dist(i,j_n)\leq 20(n\leq 4)$,则令$j_n$为子站，$j_n$连接到宿主站i，$j_n$连接的站点个数设为1，$j_n$的级跳设为1，$j_n$连接的基站设为i。同时将i的站点连接数设为n,算法描述为

```
if  Solution(1,i)=-1
   if Dist(i,jn)>0 && Dist(i,jn)<20
```

$$Solution(1,j_n)=0,$$
$$Solution(2,j_n)=i$$
$$Solution(3,j_n)=1$$
$$Solution(4,j_n)=1$$
$$Solution(5,j_n)=i$$
$$Solution(3,i)=n$$

时间复杂度为$O(n)$。

5.3.3 宿主站与宿主站之间微波连接

在确定了宿主站与子站的关系后，考虑宿主站与宿主站之间的关系。宿主站与宿主站之间只需满足距离约束即可，所以只要满足宿主站 i 与宿主站i_n之间的距离小于等于50km即可，即$Dist(i,i_1),Dist(i,i_2),Dist(i,i_3)\dots Dist(i,i_n)\leq 50$。

算法描述为：

```
   if Solution(1,i)==1
      if Dist(i,in) ≤ 50
```

$Solution(i_n,2)=i$

算法时间复杂度为$O(n)$。

5.3.4 单基站变子站操作

显然，当宿主站的数目减少时，总成本会下降。所以在设计可变邻域的过程中首先考虑将那些没有子站相连的宿主站变成子站。如果变化后存在j_n使得$Dist(i,j_n)<=10$,则寻找与 i 距离最短的站点 j,如果站点 j 的级跳小于 3 同时站点 j 对应的宿主站的连接站点个数小于连接上限，子站 j 的连接站点数为 1，则将原宿主站 i 变为子站,并连接到子站 j 上。

在将 i 连接到子站 j 上后，更新子站 j 的站点连接个数为 2，更新 j 的宿主站的站点连接数为原连接站点数+1。

算法描述为：

```
if Solution(1,i)==1 && Solution(3,i)==0  %是孤立的宿主站
   if Dist(i,jn)<=10                    %搜索与该宿主站距离小于10的子站
```

$j_{best}=\min(j_n)$

```
      if Solution(3,jbest)==1           %该子站可以被连接
         if Solution(3,Solution(5,jbest))<宿主站最大连接数
         Solution(1,i)=0;
         Solution(2,i)=jbest;
         Solution(3,i)=1;
```

Solution(4,i)=Solution(4, j_{best})+1;
Solution(5,i)=Solution(5,j_{best});
Solution(3,j_{best})= Solution(3,j_{best})+1;
Solution(3,Solution(5,j_{best}))= Solution(3,Solution(5,j_{best}))+1;

图 3 单基站变子站示意图

5.3.5 子站重连接操作

对于任意一个只有一级连接的子站j_1，即该列的第 3 行为 1，则搜索所有满足距离约束的子站。在搜索到的子站中删除目前已经连接的站j_2，若删除后还存在其他站点，则从中随机选取一个j_3。如果j_3的基站i_3没有连接满，j_3的级跳小于 3，j_3只有 1 个直接相连站点,则将j_1连接到j_3,并更新 Solution 矩阵。将j_1连接站点设为j_3,j_1的站点连接数设为 1，j_1的级跳设为j_3的级跳加 1，j_1的基站设为i_3。将j_3的站点连接数设为 2，将i_3的站点连接数加 1。

算法描述为：

if Solution（3, j_1）==1
  if $j_3 \neq j_1$ && Solution$(4,j_3) < 2$ && Solution(3, j_3)==1
    Solution(2, j_1)=j_3;
    Solution(3, j_1)=1
    Solution(4, j_1)= Solution(4, j_3)+1
    Solution(5, j_1)= Solution(5, j_3)
    Solution(3, j_3)=2;
    Solution(3, Solution(5, j_3))= Solution(3, Solution(5, j_3))+1;

图 4 子站重连示意图

5.3.6 结果绘制操作

显然可以通过遍历整个 Solution 矩阵来实现结果的输出。具体输出过程为：对于任意一列 i,如果是宿主站,则绘制红圈,如果是子站则绘制蓝星,位置由 location 矩阵的第 i 行决定。同时，如果是宿主站，则用黄虚线连接 Solution 矩阵中第 2 行的位置，如果是子站用绿实线连接。由于函数比较简单，请读者直接参考代码。

5.3.7 基于可变邻域的模拟退火算法设计

第一步：初始化：设定初始温度 T0，终止温度 Tend，降温速率 q，T=T0,gen=1。
第二步：输入位置矩阵 location，计算距离矩阵 Dist
第三步：生成初始解 best_Solution,计算初始成本 best_avgCost
第三步：判断：如果 T>Tend，则执行第四步，否则跳转到第十一步
第四步：以随机顺序访问输入位置，生成新解 new_Solution。

第五步：设定循环变量 i=0，如果 i<10，执行第六步，否则执行第九步
第六步：执行单基站变子站操作,更新 new_Solution
第七步：执行子站重连接操作,更新 new_Solution
第八步：计算 new_avgCost，i=i+1;返回第 5 步；
第九步：如果 new_avgCost<best_avgCost，
或者 exp((new_avgCost-best_avgCost))/Trand，rand 为 0-1 之间的随机数，则更新 Solution=new_Solutiion,best_avgCost=new_avgCost;
第十步：T=T*q,gen=gen+1;返回第三步
第十一步：执行基站微波相连操作
第十二步：绘制结果图，保存到 csv 文件

5.4 模型运行结果展示

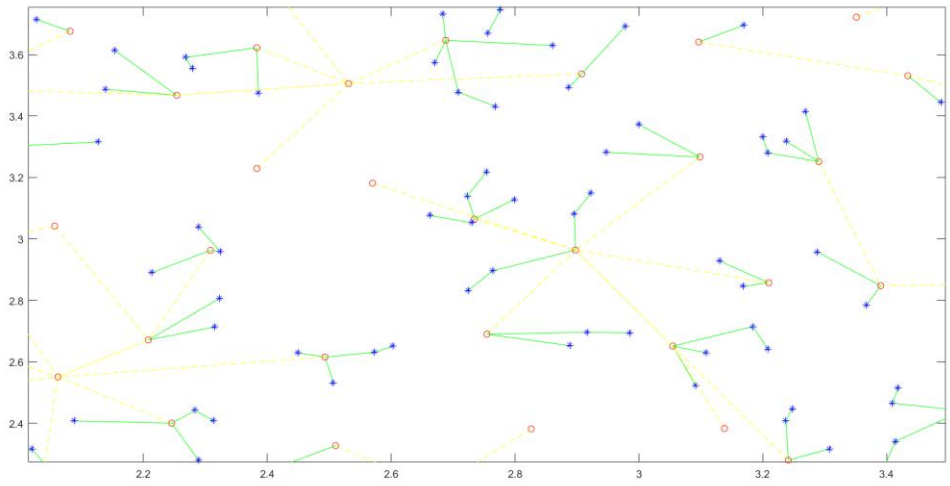


图 5 RuralStar 站型模型结果局部放大图

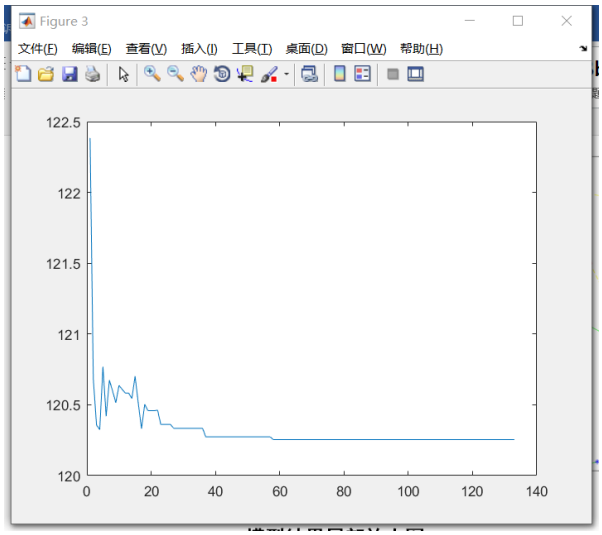


图 6 成本迭代曲线

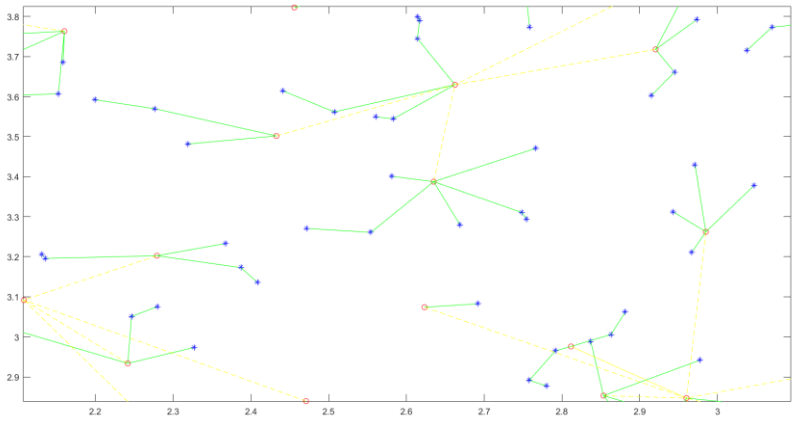


图 7 蝴蝶站站型局部放大图

局部放大图中，*符号表示子站，o 表示基站，他们之间通过实线相连。基站与基站之间通过虚线相连。有的由于显示精度问题可能线型不是很明显，如果对该图进一步放大即可看到更准确的结果。

成本迭代曲线中，横坐标是迭代次数，纵坐标是平均成本。从图中可以看出，在温度较高时，成本出现波动，在 40 代左右即收敛到成本较小的方案。同时，程序执行时间较短，1000 规模的站点位置执行时间在 30 秒左右。Rural 型宿主站的图执行了 35.753541 秒，最终结果为 120.2528。

六、模型的评价、改进与推广

本模型通过引入中间矩阵 Solution 记录模型的约束，使得每一次进行邻域变换时都能够生成可行解。然后在可行域内应用模拟退火算法对平均成本进行寻优操作，获得了较为合理的最终解。从图 8 中也可以看出，宿主站通过直接相连或级跳相连连接了周围尽可能多的子站。同时本模型中还同时考虑了 Rural 站与蝴蝶站的区别，输入允许用户设置宿主站类型。本模型最大的优点之一就是耗时少，结果展示中的算例，1000 个站点规模，30s 左右即可完成，同时得到的解接近全局最优解。

但是，模型存在部分不足，比如有的子站离宿主站更近，却连接到了稍远一些的其它子站，或者存在少量如图 8 所示的交叉分配或者折返现象。需要设计新的操作来优化，比如将末端点连接到较近的宿主站的操作。

本模型的推广较为容易，增加新的约束时，可以增加中间矩阵的行数来记录约束。在单基站变子站操作，子站重连接操作中增加约束判断条件即可生成新的符合约束的邻域。如果新的约束会产生成本，则在成本函数中增加该成本。这样模拟退火算法就能在该约束条件下搜索全局最优解了。

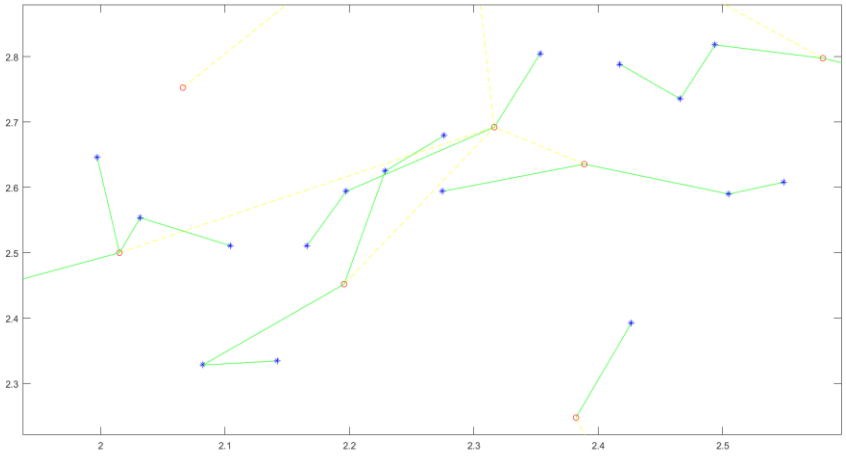


图 8

七、参考文献

[1] 熊杰,接驳地铁的社区公交微循环系统优化研究[D].北京:北京交通大学,2015:123-133;

[2] 郁磊 史峰 王辉 胡斐，MATLAB 智能算法 3 个案例分析（第二版）[M].北京：北京航空航天出版社，2016;

[3] 胡运权，运筹学教程（第 4 版）[M].北京：清华大学出版社，2016;

[4] 严蔚敏 吴伟民，数据结构（C 语言版）[M].北京：清华大学出版社，2016。

八、附录

8.1 距离矩阵获取的 matlab 代码（5.3.1）

```
function Dist=CalDist(location)
n=size(location,1);                                    %计算站点个数
Dist=zeros(n,n);                                       %确定目标矩阵维度
for i=1:n                                              %求任一一点到其他点的距离
    for j=i+1:n
        alpha1=location(i,2)*pi/180;                   %角度转为弧度
        alpha2=location(j,2)*pi/180;
        beta1=location(i,1)*pi/180;
        beta2=location(j,1)*pi/180;
        temp=cos(beta1)*cos(beta2)*cos(alpha1-alpha2)+sin(beta1)*sin(beta2
);                                                     %带入公式运算
        Dist(i,j)=6378*acos(temp);
        Dist(j,i)=Dist(i,j);
    end
end
```

8.2 初始解生成的 matlab 代码（5.3.2）

```
function solution=GetInitialSolution(location,BaseType,Dist)
%求一级连接初始解
%BaseType 0      Rural
%BaseType 1      蝴蝶站
if BaseType==0
    maxOneGradeIn=4;
    maxIn=6;
else
    maxOneGradeIn=8;
    maxIn=12;
end
n=size(location,1);
solution=zeros(5,n)-1;
for k=1:n
if solution(1,k)==-1
    solution(1,k)=1;
    PossibleSub= find(Dist(k,:)<=20 &Dist(k,:)>0 &solution(1,:)==-1);
    numPossibleSub=size(PossibleSub,2);
    if numPossibleSub<=maxOneGradeIn
        solution(1,PossibleSub)=0;                     %确定子站站点类型
        solution(2,PossibleSub)=k;                     %确定子站站点连接位置
        solution(3,k)=numPossibleSub;                  %确定基站站点连接个数
        solution(3,PossibleSub)=1;                     %确定子站站点连接个数
        solution(4,PossibleSub)=1;                     %确定子站站点级跳等级
        solution(5,PossibleSub)=k;                     %站点所在路线
```

```
        else
            solution(1,PossibleSub(1:maxOneGradeIn))=0;
            solution(2,PossibleSub(1:maxOneGradeIn))=k;
            solution(3,k)=maxOneGradeIn;
            solution(3,PossibleSub)=1;
            solution(4,PossibleSub(1:maxOneGradeIn))=1;
            solution(5,PossibleSub(1:maxOneGradeIn))=k;
        end
    end
    end
```

8.3 宿主站与宿主站之间微波连接的 matlab 代码（5.3.3）

8.4 单基站变子站操作的 matlab 代码（5.3.4）

```
    function solution_new=seperatedBase2Sub(solution,Dist,BaseType)
    if BaseType==0
        maxIn=6;
    else
        maxIn=12;
    end
    solution_new=solution;
    n=size(solution_new,2);                                    %计算站点数目
    for i=1:n                                                  %对于任意一个站点
       if solution_new(3,i)==0                          %如果是宿主站且连接站点数为 0
          couldConnect=find(Dist(i,:)<=10&Dist(i,:)>0&solution_new(1,:)==0);
          %求可以连接的最近的子站
          if ~isempty(couldConnect)                             %如果存在
              [minDistance,location]=min(Dist(i,couldConnect));
                 %如果不止一个，寻找这些站点中离该站最近的子站相连接
               if solution_new(1,couldConnect(location))==0
               if solution_new(4,couldConnect(location))<3
               %判断需要连接的子站是否为 2 级级跳或以下，如果是
                  If solution_new(3,solution_new(5,couldConnect(location)))<maxIn
            %判断基站连接站点个数是否小于连接上限
               solution_new(1,i)=0;                              %将该基站换成子站
               solution_new(2,i)=couldConnect(location);     %将该站连接到目标子站
               solution_new(3,i)=1;                          %将该站连接站点数设为 1
               solution_new(4,i)=solution_new(4,couldConnect(location))+1;
                                                  %将该站级跳设为上一站点级跳加 1
               solution_new(5,i)=solution_new(5,couldConnect(location));
                                                 %将该站线路设为上一级站点的基站
             solution_new(3,solution_new(5,i))=solution_new(3,solution_new(5,i))+1;
                                                 %将基站对应的连接站点数加 1
            solution_new(3,couldConnect(location))=
                solution_new(3,couldConnect(location))+1;
                    end
```

```
                end
            end
        end
    end
end
```

8.5 子站重连接操作的 matlab 代码（5.3.5）

```
function solution_new=subReconnect(solution,Dist,BaseType)
if BaseType==0
    maxIn=6;
else
    maxIn=12;
end
solution_new=solution;
n=size(solution_new,2);                                  %计算站的个数
for k=1:n                                                %对于任意一个站
  if solution_new(1,k)==0                                %如果是子站
      if solution_new(3,k)<2                             %并且该子站只有一级连接
          PossibleConnectStation=find(Dist(k,:)>0 & Dist(k,:)<=10 );
                                                 %搜索周围有没有可以连接的子站
          oldConnectStation_location=
              PossibleConnectStation==solution_new(2,k);
                                       %在可以连接的子站中删除现在连接的站
          PossibleConnectStation(oldConnectStation_location)=[];
          if ~isempty(PossibleConnectStation)
                                       %如果还存在其他可以连接的子站
            num=size(PossibleConnectStation,2);      %从中随机选出 1 个
            num=ceil(rand*num);
             if solution_new(1,PossibleConnectStation(num))==0
               if solution_new(3,solution_new
                 (5,PossibleConnectStation(num)))<maxIn
                                       %如果该站点对应的基站没有连接满
                  if solution_new(4,PossibleConnectStation(num))<3
                                  %如果该站点对应的级跳小于 3，则可以连接
                     if solution_new(3,PossibleConnectStation(num))<2
                                       %如果该站点位于末端
                        if solution_new(1,PossibleConnectStation(num))==0
                                       DistConnect2Base=Dist(PossibleConnectSta
                  tion(num),solution(5,PossibleConnectStation(num)));
                                       DistCurrent2Base=Dist(k,solution
                  (5,PossibleConnectStation(num)));
                                        if DistCurrent2Base>DistConnect2Base
                                         solution_new(2,k)=
                  PossibleConnectStation(num);
                                       %将该站点连接到目标子站
```

```
                                    solution_new(3,k)=1;
                                   %将该站点连接个数设为 1
                                    solution_new(4,k)=
                            solution(4,PossibleConnectStation(num))+1;
                                   %将该站点级跳设为上一等级级跳+1
                                    solution_new(5,k)=solution
                             (5,PossibleConnectStation(num));
                                   %将该站点基站设为目标子站的基站
                                  solution_new(3,PossibleConnectStation(num))
                               =solution_new(3,PossibleConnectStation(num))+1;
                                  %将该站点连接的目标子站的连接站点数加 1
                                    t=solution_new(5,k);
                                  solution_new(1,PossibleConnectStation(num));
                                    temp=solution_new(3,t);%
                                =solution_new(3,solution_new(5,k))+1;
                                   %将该站点对应的基站连接站点数加 1
                                    solution_new(3,t)=temp+1;
                                   else
                                        end
                                   end
                               end
                            end
                        end
                    end
                end
            end
        end
    end
end
```

8.6 结果绘制操作的 matlab 代码（5.3.6）

```
function plotResult(solution,location)                %将运算结果可视化
n=size(solution,2);                                   %获取站点个数
for k=1:n                                             %对于任意一个站点来说
    if solution(1,k)==0                               %如果是子站
        x1=location(k,1);                             %获取子站的横坐标
        x2=location(solution(2,k),1);                 %获取对应宿主站的横坐标
        y1=location(k,2);                             %获取子站的纵坐标
        y2=location(solution(2,k),2);                 %获取对应宿主站的纵坐标
    plot([x1,x2],[y1,y2],'g-');                       %连线
    plot(x1,y1,'b*');                                 %绘制子站
    hold on;
    elseif solution(1,k)==1 && solution(2,k)~=-1      %如果是基站且有连接
        x1=location(k,1);                             %获取基站 1 的横坐标
        x2=location(solution(2,k),1);                 %获取基站 2 的横坐标
```

```
            y1=location(k,2);                                %获取基站 1 的纵坐标
            y2=location(solution(2,k),2);                    %获取基站 2 的纵坐标
        plot([x1,x2],[y1,y2],'y--');                         %基站之间用虚线连接
        plot(x1,y1,'ro');                                    %绘制该基站
        hold on;
        else                                                 %如果是基站且无连接
            plot(location(k,1),location(k,2),'ro');          %直接绘制基站图标
            hold on;
        end
    end
```

8.7 基于可变邻域的模拟退火算法设计的 matlab 代码（5.3.7）

8.7.1 main 函数的代码

```
best_location=rand(1000,2)*5;
%随机生成东经 0-6，北纬 0-6 度的 1000 个坐标
tic;
T=1;q=0.95;Tend=1e-8;gen=1;
n=size(best_location,1);
BaseType=0;
Dist=CalDist(best_location);                                 %计算任意两点间的距离
best_Solution=GetInitialSolution(best_location,BaseType,Dist);   %获得初始解
best_Solution=Base2Base(best_Solution,Dist);                 %基站之间的微波连接
best_avgBSCost=calAvgBuildStationCost(best_Solution);
best_totalBSCost=calBuildStationCost(best_Solution);
best_avgPLCost=calAvgPLCost(best_Solution,Dist);
best_totalPLCost=calPLCost(best_Solution,Dist);
best_avgCost=best_avgBSCost+best_avgPLCost;
avgCost(gen)=best_avgCost;
avgBSCost(gen)=best_avgBSCost;
avgPLCost(gen)=best_avgPLCost;
totalBSCost(gen)=best_totalBSCost;
totalPLCost(gen)=best_totalPLCost;
plotResult(best_Solution,best_location);
while T>Tend
vector=randperm(n);
new_location=zeros(size(best_location));
for i=1:n;
    new_location(i,:)=best_location(vector(i),:);
end
new_Dist=CalDist(new_location);
new_Solution=GetInitialSolution(new_location,BaseType,new_Dist);
for i=1:30
new_Solution=seperatedBase2Sub(new_Solution,new_Dist,BaseType);
new_Solution=subReconnect(new_Solution,new_Dist,BaseType);
new_avgBSCost=calAvgBuildStationCost(new_Solution);
```

```
new_avgPLCost=calAvgPLCost(new_Solution,new_Dist);
new_totalBSCost=calBuildStationCost(new_Solution);
new_avgCost=new_avgBSCost+new_avgPLCost;
new_totalPLCost=calPLCost(new_Solution,new_Dist);
end
if new_avgCost<best_avgCost
    best_location=new_location;
    Dist=new_Dist;
    best_avgCost=new_avgCost;
    best_Solution=new_Solution;
    best_avgBSCost=new_avgBSCost;
    best_totalBSCost=new_totalBSCost;
    best_avgPLCost=new_avgPLCost;
    best_totalPLCost=new_totalPLCost;
elseif exp(-(new_avgCost-best_avgCost)/T)>rand
    best_location=new_location;
    Dist=new_Dist;
    best_avgCost=new_avgCost;
    best_Solution=new_Solution;
    best_avgBSCost=new_avgBSCost;
    best_totalBSCost=new_totalBSCost;
    best_avgPLCost=new_avgPLCost;
    best_totalPLCost=new_totalPLCost;
end

gen=gen+1;
avgCost(gen)=best_avgCost;
avgBSCost(gen)=best_avgBSCost;
avgPLCost(gen)=best_avgPLCost;
totalBSCost(gen)=best_totalBSCost;
totalPLCost(gen)=best_totalPLCost;
T=q*T;
end
figure(2);
best_Solution=Base2Base(best_Solution,Dist);
plotResult(best_Solution,best_location);
figure(3);
plot(avgCost);
Write2csv(best_Solution);
csvwrite('location.csv',best_location);
toc
```

8.7.2 计算路径损耗 PL 的代码

```
function totalPLCost=calPLCost(solution,Dist)
F=900;                                          %发射频率 900MHz
totalPLCost=0;
```

```
n=size(solution,2);
for i=1:n
    if solution(1,i)==0
    totalPLCost=totalPLCost+32.5+20*log10(Dist(i,solution(2,i)))+20*log10(F);
    end
end
```

8.7.3 计算平均路径损耗 PL 的代码

```
function avgPLCost=calAvgPLCost(solution,Dist)
F=900;                                                    %发射频率
900MHz
totalPLCost=0;
n=size(solution,2);
for i=1:n
    if solution(1,i)==0
    totalPLCost=totalPLCost+32.5+20*log10(Dist(i,solution(2,i)))+20*log10(F);
    end
end
SubStation=find(solution(1,:)==0);
numSubStation=size(SubStation,2);
avgPLCost=totalPLCost/numSubStation;
```

8.7.4 计算平均成本的代码

```
function avgBSCost=calAvgBuildStationCost(solution)
BaseStation=find(solution(1,:)==1);
SubStation=find(solution(1,:)==0);
numBaseStation=size(BaseStation,2);
numSubStation=size(SubStation,2);
numSatellite=ceil(numBaseStation/8);

avgBSCost=(numBaseStation*10+numSubStation*5+numSatellite*50)/(numBase
Station+numSubStation);
```

8.7.5 计算总体成本的代码

```
function totalBSCost=calBuildStationCost(solution)
BaseStation=find(solution(1,:)==1);
SubStation=find(solution(1,:)==0);
numBaseStation=size(BaseStation,2);
numSubStation=size(SubStation,2);
numSatellite=ceil(numBaseStation/8);

totalBSCost=numBaseStation*10+numSubStation*5+numSatellite*50;
```

8.7.6 输出两个矩阵 Graph.csv 和 Posi.csv 的代码

```
function Write2csv(solution)
%将 solution 矩阵按指定格式写入 csv 文件中
n=size(solution,2);
Graph=zeros(n,n);
BaseStop= solution(1,:)==1;
```

```
for i=1:n
	if solution(2,i)~=-1 && solution(1,i)==0
		Graph(i,solution(2,i))=1;
		Graph(solution(2,i),i)=1;
	end
	if solution(2,i)~=-1 && solution(1,i)==1
		Graph(i,solution(2,i))=2;
		Graph(solution(2,i),i)=2;
	end
end
strRow=1:n;
strCol=strRow';
stopType=[strCol,zeros(n,1)];
stopType(BaseStop,2)=1;
csvwrite('Posi.csv',stopType);
strCol=[0;strCol];
Graphcsv=[strRow;Graph];
Graphcsv=[strCol,Graphcsv];
csvwrite('Graph.csv',Graphcsv);
```