无线可充电传感器网络充电路线规划问题

摘要

无线可充电传感器网络的正常运转，依赖于移动充电器（MC）定期为传感器的电池补充能量。通过建立 0-1 规划模型，求解单 MC 和多 MC 模式下行驶途中能量消耗最小的充电路线，并分别计算保证系统一直正常运行的传感器电池容量。

针对问题一：规划单 MC 行驶途中能量消耗最小的充电路线。MC 在路上的能量消耗与其行驶距离成正比，将 MC 充电路线规划问题转换成求解最短路径的旅行商问题。首先利用附件 1 中各节点经纬度坐标求解各节点间的距离，然后以 MC 总行驶距离最短为目标，通过建立 MC 只能经过传感器网络中的各节点一次和消除支路等约束，建立 0-1 规划模型求解（见公式 5），最后用 lingo 软件进行编程（见程序 T1）得到最优的充电路径（见表 2），得到的充电路径最短总路程为 11.4851km。

针对问题二：MC 从某传感器驶出遍历其他所有传感器后返回的过程称为一圈，通过分析两圈 MC 移动充电过程，得出各传感器充电时间随着圈数的增加而增加。观察到充电过程中某传感器耗电时间=充电路径行驶时间+除该传感器外所有传感器的充电时间，将某传感器的前一紧邻路径行驶时间与该传感器充电时间之和定义为传感器花费时间，然后根据实际应用场景设置移动速度、充电速率等参数，利用 MATLAB 将 MC 移动充电过程进行计算机模拟（见程序 T2），记录并更新传感器花费时间，得到最大的消耗电量为 13.8330mA。假设所有传感器电池容量相同，某传感器消耗最大耗电量后恰好达到电量阈值，通过公式 8 计算出满足所有传感器一直稳定运行的电池电量应至少为 17.2912mA。选择充电速率和 MC 移动速率两个参数对模型合理性进行检验，电池电量随两者增大成线性递减，符合实际情况。

针对问题三：首先要解决同时派出 4 个 MC 在路上能量消耗最小化的问题，与问题一类似，将规划 4 个移动充电器各自要访问的传感器及访问顺序使总路径最小的问题转换成一个多旅行商问题。以四个 MC 总充电路径最短为目标，建立多 MC 移动充电路线规划的 0-1 规划模型（见公式 13），通过 MATLAB 利用模拟退火算法求解（见程序 T3-6）。最后根据得出的四条路径，四条路径的总路程最短为 12.756km，利用问题二中的算法进行计算机模拟，得出传感器电量为 2.6108mA 时可以保证系统一直正常运行。

最后，给出每个问题解决方法的优缺点及评价。

关键词：传感器网络 0-1 规划模型 旅行商问题 计算机模拟 模拟退火算法

一、问题重述

由一个数据中心（DC）和一个传感器（S）组成的无线传感器网络（WSN）具有数据收集、处理以及传输三个功能，在日常生活中得到广泛应用。WSN 又根据能量供给方式的不同，分为能量收集式无线传感器网络和无线可充电传感器网络 WRSN 两种，传感器的电量只有在高于 f(mA)时才能正常工作，为保证传感器的正常工作，必须通过移动充电器（MC）定期为传感器电池补充能量。为了提高 WRSN 的充电效率，众多学者投入到 WRSN 移动充电的问题研究中[1]。

据以上背景，已知每个传感器的经纬度和能量消耗速率（见附件 2），并假设 MC 的移动速度为 v(m/s)、充电速率为 r（mA/s），需要解决以下问题：

(1)规划只派出一个 MC 时， MC 路上能量消耗最小的充电路线。

(2)采用问题(1)规划的充电路线，分析每个传感器的电池容量至少是多少 mA 才能保证网络中每个传感器的电量都不会低于 f(mA)。

(3)为提高充电效率，同时派 4 个 MC 充电，规划所有 MC 路上总能量消耗最小的充电路线，并分析此时每个传感器的电池容量至少是多大，才能保证整个系统一直正常运行。

其中，附件一给出了数据中心和传感器的经纬度坐标，附件二给出数据中心和传感器经纬度坐标外，还给出了每个传感器的能量消耗速率。

二、问题分析

2.1 问题一的分析

问题一要求规划单 MC 移动充电路径，使得 MC 在路上的能量消耗最小。假定 MC 行驶中的能量消耗与其行驶距离成正比，求能量消耗最小的路线也就是规划 MC 经过所有传感器的最短路线。由此，将数据中心和传感器组成的传感器网络看作一个完全图，将充电路径规划问题转化为一个经典的旅行商问题，即找到一条能够走过图中所有顶点（每个顶点只被访问一次）并且使得边权值之和最小的路径。在建立传感器网络时，利用附件 1 中数据中心和各传感器的经纬度坐标求出数据中心与传感器、各传感器之间的距离，从而得到传感器网络各边的权重。以 MC 经过各传感器的总行驶距离最短为目标，建立 0-1 规划模型来求得 MC 移动充电最短路径。通过 lingo 软件利用分支定界法求解，最终给出 MC 在路上能量消耗最小的充电路径。解决问题一的思路流程图如下：

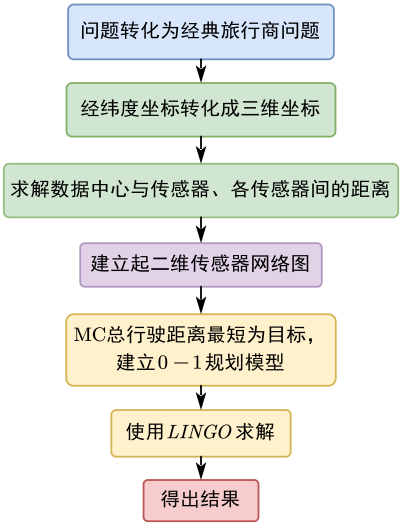

图 1　问题一的思路分析图

2.2 问题二的分析

基于问题一规划的充电路线，为使网络中各传感器的电量都不低于电量阈值，问题二要求计算各传感器的电池容量。考虑到实际生活中电池都是按规格生产不便于细分，并且最大的电池容量能够保证网络中所有传感器的电量在消耗时都不低于电量阈值，因此我们把每个传感器的电池容量都取最大的电池容量[2]。计算传感器电池容量的关键是保证各传感器在消耗电量过程中，电量不低于电量阈值（电池容量的百分比），又各传感器的耗电速率已知，可以通过计算传感器网络长期稳定运行时最大的消耗电量，假设该最大的消耗电量恰好将电池消耗到电量阈值，利用电量阈值与电池容量的百分比关系即可求出传感器网络中最大的电池容量。由于传感器数量较多且迭代次数较大，我们考虑通过计算机模拟的方法利用 MATLAB 软件进行求解。

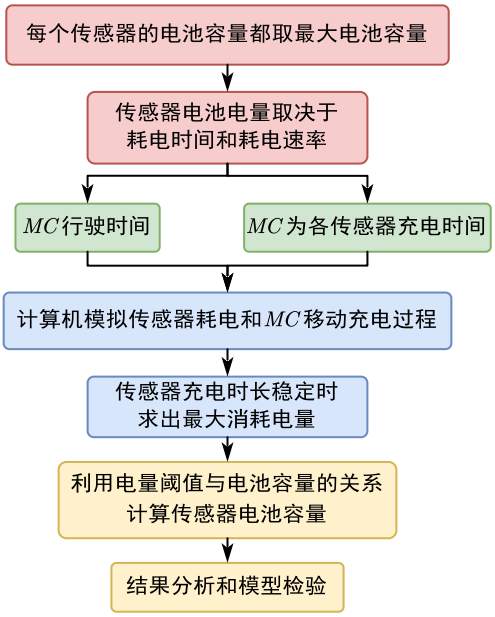

图 2　问题二的思路流程图

2.3 问题三的分析

问题三与问题一和问题二类似，要求首先规划四个 MC 在路上总能量消耗最小的充电路线，然后求出四 MC 模式下每个传感器的电池电量。路径规划时，参考第一问的思想，将该问题转化成 MTSP 问题，通过构建 0-1 规划模型进行求解。值得注意的是，四个 MC 至少经过一个传感器，而且规划出的最短路径的具有唯一性，我们需要重点考虑这些约束条件以保证结果的准确性。多 MC 充电路径规划属于 NP-hard 问题，迭代次数大且存在多个局部最优解，考虑使用启发式算法中模拟退火进行求解。为了求解传感器的电池容量，基于规划的路径，利用第二问传感器耗电和 MC 充电的计算机模拟程序进行求解即可。（加分析）

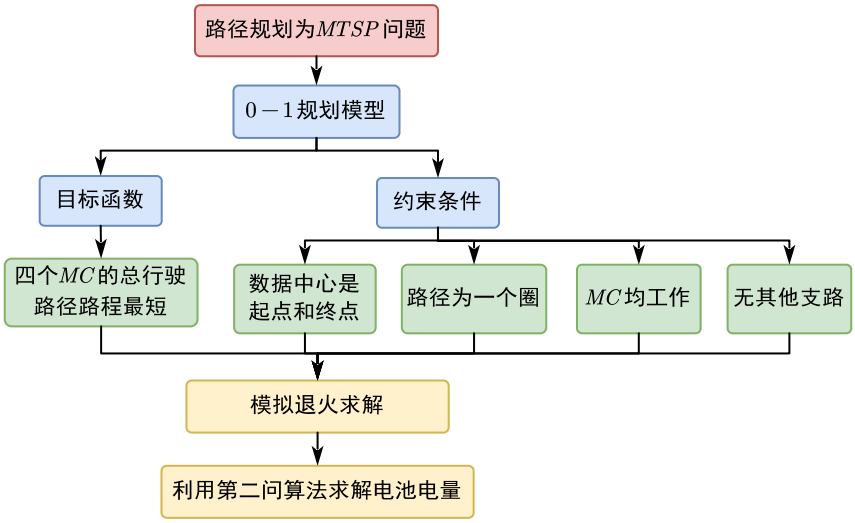

图 3 问题三的流程图

三、模型假设

假设 1：MC 行驶中的能量消耗与其行驶距离成正比；
假设 2：传感器网络中，所有传感器具有相同的电池容量；
假设 3：初始状态所有传感器的电池处于满电状态；
假设 4：不考虑 MC 在数据中心充电和维护时间。

四、符号说明

表 1　符号说明

符号	说明
x_{ij}	0-1 变量，1 表示充电路线中包含传感器 i 到 j 的路径
d_{ij}	两个传感器i与j之间的距离
D_T	拣货完成时的总距离，即为目标函数
ID	每一个货格的唯一标识，计算方法在 4.1.1 中说明
r	用于标识货格位于货架左侧或是右侧，计算方法在 4.1.1 中说明
P_i	确定起始复核台后，完成任务单 i 时的最短路径
E_i	任务单 i 的最短路径的结束复核台
B_i	任务单 i 的最短路径的起始复核台
T	问题四的完成 49 个任务单的花费时间

注：未列出符号及重复符号以出现处为准

五、问题一的模型建立与求解

5.1 解决问题一的准备

5.1.1 解决问题一的主要思想

问题一要解决 MC 在充电路上能量消耗最小化的问题，考虑到 MC 在路上的能量消耗与它的行驶距离成正比，能量消耗最小化其实是要找到一个路程最短的传感器访问顺序，将 MC 充电路线规划问题转换成经典的旅行商问题。

5.1.2 利用附件 1 数据求节点间边权值

为了确定传感器网络的边权值，我们将附件中传感器网络节点的经纬度坐标转换成三维坐标，得到任意两个传感器节点及数据中心与各传感器之间的直线距离[1]。

附件一中给出的各点经纬度坐标用有序数对(*u,v*)表示，*u* 为经度，*v* 为纬度。以地心 *O* 为坐标原点，赤道平面为 *xOy* 平面，0 度经线圈所在的平面为 *xOz* 平面建立三维直角坐标系[1]，则数据中心和各传感器的三维坐标为：

$$\begin{cases} x = R\cos u\cos v \\ y = R\sin u\cos v \\ z = R\sin v \end{cases} \tag{1}$$

其中，地球半径$R = 6370km$。

据解析几何相关知识，传感器网络中任意两节点$A(u_A, v_A), B(u_B, v_B)$间的实际距离为：

$$d = R\arccos[\cos(u_A - u_B)\cos v_A\cos v_B + \sin v_A\sin v_B]$$

将计算得到的各节点间的实际距离作为传感器网络各边的权重。

5.2 基于 TSP 问题的单 MC 充电路线规划

准备工作完成后，首先对问题相关参数进行描述。充电路线的起点和终点都是数据中心，为了便于表示，将数据中心看作 0 号传感器，将数据中心与所有的传感器记为一个节点集合$A\{a_0,a_1,...,a_{n-1}\}$。d_{ij}是两个传感器i与j之间的距离，据附件 1 中求得的节点间距离，得到数据中心与传感器之间、各个传感器之间的距离矩阵 D：

$$D=(d_{ij})_{n\times n}=\begin{bmatrix} 10000 & 4.883 & \cdots & 63.341 \\ 4.883 & 10000 & \cdots & \\ \vdots & \vdots & \ddots & \vdots \\ 63.341 & & \cdots & 10000 \end{bmatrix}_{30\times 30}$$

5.2.1 单 MC 充电路径规划问题的目标

MC 在充电路上行驶消耗的能量与其行驶路线长度成正比，要使 MC 在路上的能量消耗最小，就是要使 MC 移动充电总路线路程最短。为此，设 0-1 变量x_{ij}来记录总路线中是否包含某两节点间的通路，可得单 MC 充电路径规划问题的目标函数:

$$\min \sum_{i\neq j} d_{ij}x_{ij} \tag{2}$$

其中，$x_{ij}=\begin{cases}1，充电路线中包含传感器i到j的路径\\0，充电路线中不包含传感器i到j的路径\end{cases}$。

5.2.2 相关约束

为了让 WRSN 正常运转，MC 定期从数据中心a_0出发依次经过每个传感器为其充电，直到为所有传感器充电完毕后返回数据中心。所以， MC 只能经过传感器网络中的各节点一次，即网络中各节点满足只有一条入边和一条出边：

$$\begin{aligned}\sum_{j=1}^{n} x_{ij}=1,i=1,2,\cdots,n\\ \sum_{i=1}^{n} x_{ij}=1,j=1,2,\cdots,n\end{aligned} \tag{3}$$

为了保证 MC 的充电路线是图中唯一的圈，而不存在图中其他的支路的情况（如图中红线所示），也就是说，整个传感器网络中只能存在一个包含所有节点的圈，去掉一个或者几个点都不能形成闭圈[1]。记s为$\{1,2,...,n\}$的子集，则$|s|$表示集合 s 中元素的个数，则有：

$$\sum_{i,j\in s} x_{ij}\leqslant |s|-1,\ 2\leqslant |s|\leqslant n-1,\ s\subset\{1,2,...,n\} \tag{4}$$

其中，$x_{ij}\in\{0,1\}$，$i,j=1,2,\cdots,n$，$i\neq j$。

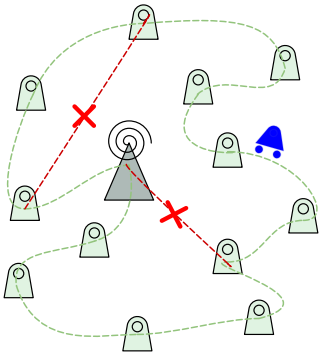

图 4 支路示意图

综上，单 MC 移动充电路线规划模型如下：

$$\text{目标函数：}\quad \min \sum_{i \neq j} d_{ij} x_{ij}$$

$$s.t. \begin{cases} \sum_{j=1}^{n} x_{ij} = 1 & i = 1, 2, \cdots, n \\ \sum_{i=1}^{n} x_{ij} = 1 & j = 1, 2, \cdots, n \\ \sum_{i,j \in s} x_{ij} \leqslant |s| - 1 & 2 \leqslant |s| \leqslant n-1 \quad s \subset \{1, 2, ..., n\} \\ x_{ij} \in \{0, 1\} & i, j = 1, 2, \cdots, n, i \neq j \end{cases} \tag{5}$$

5.3 问题一的求解

为了求解移动充电器的路径这样一个 NP 问题，我们根据上述模型，利用 lingo 软件（内部利用分支定界的剪枝算法）进行求解，结果显示为全局最优解，得到最短路径如下表（如表 2）：

表 2 单 MC 最短充电路径规划结果表

最短路线距离	11.4851km
最短路线	1→18→21→20→19→26→27→30→22→24→25→29→23→5→4→6→11→14→17→28→16→13→9→12→15→7→8→10→2→3→1

为了更直观地展示结果，利用 MATLAB 软件将数据中心及传感器各点标在以经度为横坐标以纬度为纵坐标的直角坐标系中，依据 lingo 软件的结果将这些点进行连线得到如下路线规划图（如图 5）：

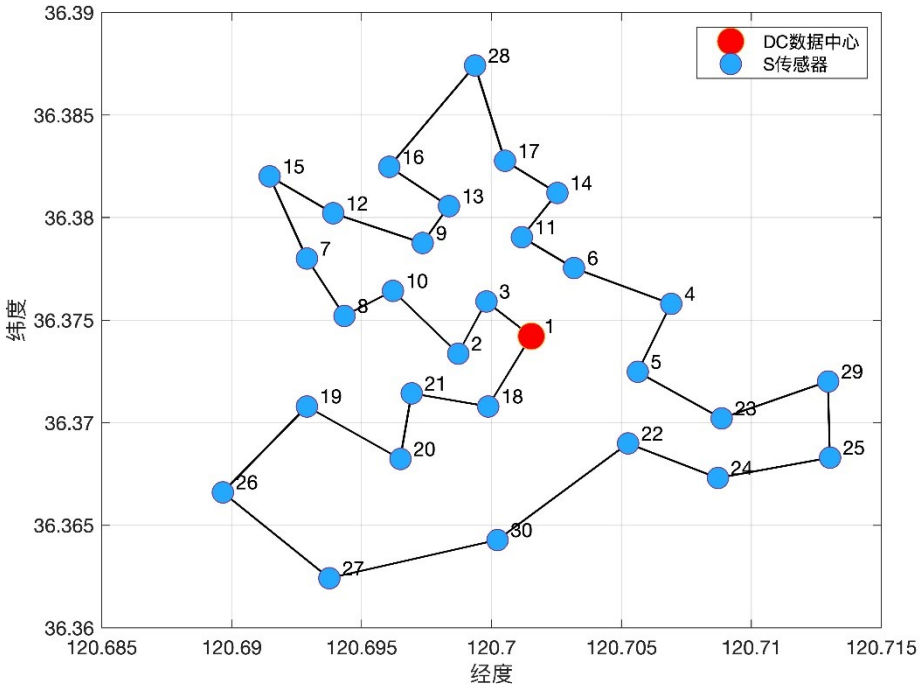

图 5 单 MC 最短充电路径规划图

六、问题二的模型建立与求解

6.1 问题分析

基于问题一规划的充电路线，为使网络中各传感器的电量都不低于电量阈值，问题二要求计算各传感器的电池容量。计算传感器电池容量的关键是保证各传感器在消耗电量过程中，电量不低于电量阈值（电池容量的百分比），又各传感器的耗电速率已知，可以通过计算传感器网络长期稳定运行时最大的消耗电量，假设该最大的消耗电量恰好将电池消耗到电量阈值，利用电量阈值与电池容量的百分比关系求出电池容量。由于传感器数量较多且迭代次数较大，我们考虑通过计算机模拟的方法利用 MATLAB 软件进行求解。

6.2 传感器耗电和 MC 移动充电过程

求解传感器电池容量的关键点是耗电时间的计算。在传感器网络工作过程中，各传感器均以不同的耗电速率工作放电，假如以一次 MC 到达给传感器电池充满电时刻为计时起点，以下一次 MC 到达为传感器充电时刻为记时终点，称这样一次 MC 沿着规划路线对传感器充电的过程为一圈，在这段过程中该传感器耗费的电量与其特定的单位时间耗电速率和该过程总时间密切相关。每个传感器的耗电速率在附件 2 中已给出，那么解决此问题的关键在于求解耗电时间。

首先假设初始状态的所有传感器满电，MC 在数据中心开始沿着问题一规划的路径行驶为路径上各传感器充电，各传感器也从此刻开始耗电。通过讨论，我们发现由于第一圈 MC 行驶充电时各传感器电池消耗都很小，比如图 6 离数据中心最近的 1 号传感

器，若它是第一个访问的传感器，那么它的电池消耗时间只有 MC 在 0-1 路上行驶的时

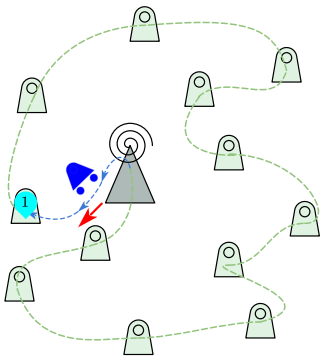

图 6　MC 充电第一圈耗电时间示意图

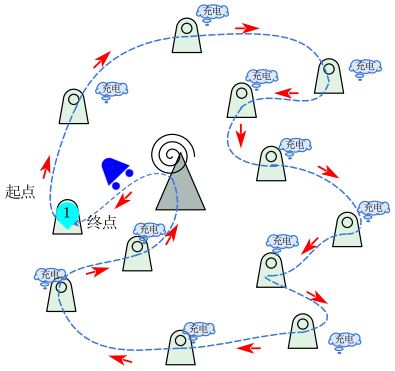

图 7　MC 充电第 n 圈耗电时间示意图

间。以 1 号传感器为例，第二圈过程中 1 号充电器耗电时间是由 MC 沿整个路径行驶的时间加上除一号传感器外所有传感器的充电时间，可以发现第二圈的耗电时间比第一圈耗电时间大，以此类推，其他的传感器也具有相同的情况，并且各传感器的耗电时间随着行驶圈数的增大而增大，最终趋于一个稳定的值。可以得到 i 号传感器在第 a 圈的耗电时间T_{ai}的表达式：

$$T_{ai} = t_d + t_{ai} \tag{6}$$

其中，t_d 表示整个路径的 MC 行驶时间，t_{ai} 表示第 i 号传感器在第 a 圈的充电时间，行驶时间计算方式：

$$t_d = \frac{\sum_{x_{ij} \in S} x_{ij}}{v} \tag{7}$$

为了计算第 i 号传感器在第 a 圈的充电时间t_{ai}，将第 a 圈中一个节点的充电时间加从前一个点到该点的行驶时间记为t_{ai}'，在模拟每一圈的过程中进行记录并更新，通过叠加上一圈除 i 号传感器外所有的耗电时间即可得到t_{ai}。

6.3 计算机模拟求解最大消耗电量

计算最大的消耗电量，除耗电时间外，还要给出电量消耗和充电过程中的相关参数，综合了对实际应用场景的模拟以及对前人研究的成果，我们对相关进行设置（如下表）。

表 3　WRSN 参数表

参数名称	符号	数值
传感器充电速率	r	174mA/h
MC 移动速率	v	60km/h
电量阈值	f	0.2E
传感器耗电速率	u	见附件 2

根据前面对传感器耗电和 MC 移动充电时间计算的描述，我们采用算法实现这个计算机模拟的过程，算法流程图如下（如图 8）：

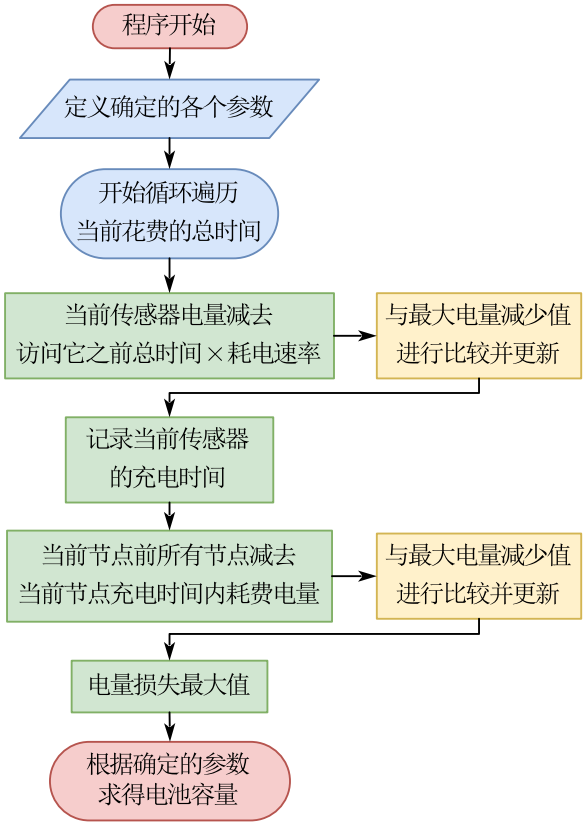

图 8 传感器充电耗电计算机模拟程序流程图

利用 matlab 软件实现上述过程，得到随着遍历最优序列的圈数，电量损失最大值的变化曲线（如图 9）：

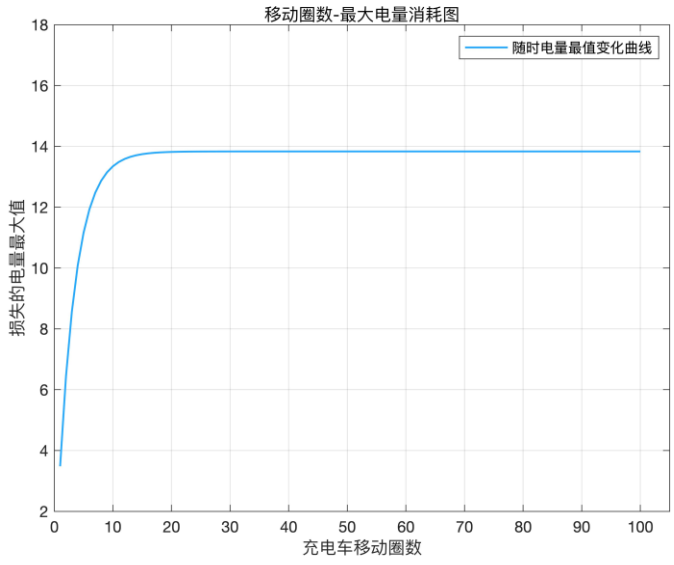

图 9 网络中最大耗电量随 MC 行驶圈数变化图

从上图可以看出在循环 20 圈后目标值的变化已经很小，说明 20 圈后每个传感器的充电时间和耗电时间基本不变，结合具体数据得到最大损失电量E_u稳定在 13.8330 mA。结合我们查询的资料，得到最少工作电量大约占电池总电量的 20%，以 BA 表示电池总电量，f 即电量阈值，可知传感器电池电量计算公式：

$$\frac{E_u}{BA} \leq 1-f \qquad (8)$$

代入求得 BA =17.2912 mA，由此可知各传感器电池容量最小是 17.2912 mA。

6.4 模型合理性检验

为了检验模型和算法的合理性，我们选取 MC 充电速率和 MC 移动速度两个参数，通过对数值进行一定范围扰动，观察电池电量与两者之间的变化关系，通过 MATLAB 软件求解结果如下：

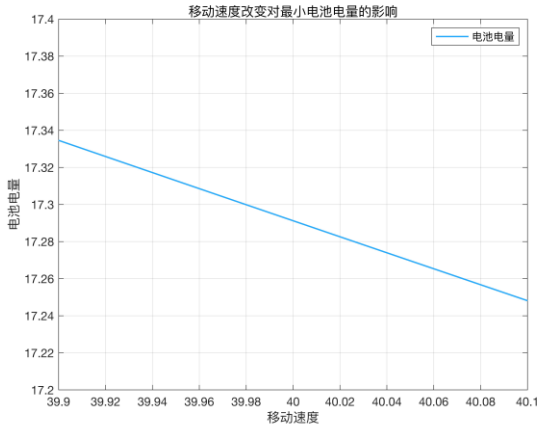

图 10 移动速度与电池电量变化关系图

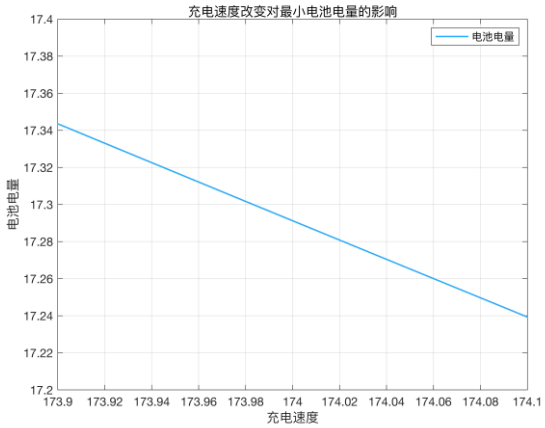

图 11 充电速度与电池电量变化关系图

由图 10、图 11 可知，随着 MC 的移动速度和充电速率的增加，传感器电池电量呈线性递减状态，在现实生活中，MC 移动速度越快会导致路途行驶耗时变短，同时会减小网络中所有传感器的耗电时间，那么保证该网络稳定运行的传感器电池电量也会减小；同理，充电速率也是通过影响耗电时间而对电池容量要求变小，所以得出的结果基本符合实际。而且移动速度在 0.2km/h 的扰动范围中就会引起电池电量大约 0.1mA 的变化范围，充电速率在 0.2mA/h 的扰动范围中会引起电池电量大约 0.1mA 的变化范围，说明 MC 的移动速度和充电速率对传感器电池电量影响较大，并且两者重要地位大致相同。

七、问题三的模型建立与求解

现实生活中，传感器之间相距较远，如果仅有一辆移动充电器往往难以满足众多传感器的充电任务。为提高无线可充电传感器网络的充电效率，现研究同时派出 4 个移动充电器进行充电的情况。

7.1 解决问题三的思想

问题三首先要解决同时派出 4 个移动充电器在路上能量消耗最小化的问题，与问题一类似，但移动充电器的数目增加到 4 个，也就是为这 4 个 MC 规划各自要访问的传感器及访问顺序使得总路径最小，这实质上是一个多旅行商问题（MTSP）问题。其次，为了求得多 MC 移动充电模式下的各传感器电池容量，仍假设网络中所有传感器电量相同，根据得出的四条路径，利用问题二中的算法进行计算机模拟，即可得出保证系统一直正常运行的传感器电量。

7.2 基于 MSTP 问题的多 MC 移动充电路线规划

（1）优化目标的确定

根据问题一中的描述，我们知道传感器网络其实是一个具有 30 个顶点的加权图 G，网络中各边权重以各节点间的距离d_{ij}表示。将本问中的四个移动充电器看作四个旅行商，传感器网络中的所有传感器以及数据中心看作城市，那么多 MC 总充电路线规划即在加权图 G 中求顶点集 V 的划分V_1,V_2,V_3,V_4，将 G 分成 4 个连通的子图[3]，每个子图中寻找一条包含数据中心的回路，并达到四条回路的总边权值之和最小。（示意图见图 12）

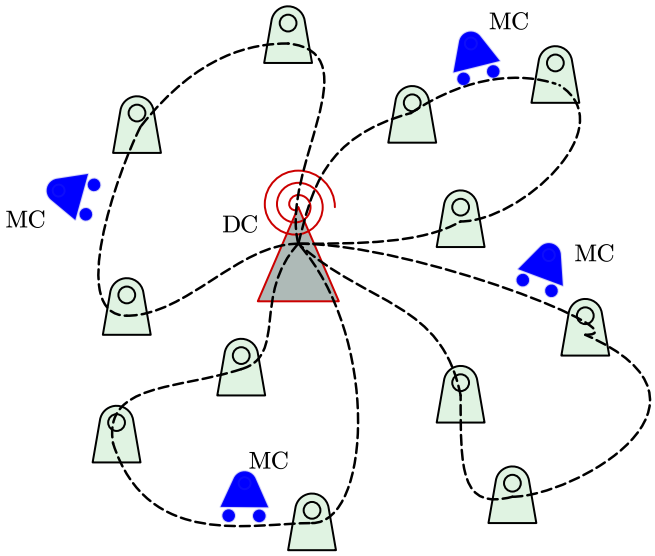

图 12 基于多旅行商的 MC 充电路线规划示意图

问题一中，传感器节点间的距离矩阵$D_{ij}=(d_{ij})_{n\times n}$已知，为表示各段路程的累加，首先通过定义 0-1 变量来表示各 MC 是否从某一传感器到另一传感器：

$$x_{ij}^{k}=\begin{cases}1,\text{移动充电器}b_k\text{从传感器}a_i\text{到达传感器}a_j,\\0,\text{否则}.\end{cases}$$

$$y_{i}^{k}=\begin{cases}1,\text{移动充电器}b_k\text{经过传感器}a_i,\\0,\text{否则}.\end{cases}$$

则第 k 个移动充电器经过传感器的路径之和$z_k=\sum_{i=0}^{n}\sum_{j=0}^{n}d_{ij}x_{ij}^{k}$，$k=1,2,3,4$，充电路线规划的目标就是使这 4 个 MC 充电的总路径最小：

$$\min Z=\sum_{i=1}^{k}z_k \tag{9}$$

（2）约束条件分析

MC 从数据中心a_0出发，除数据中心外其余传感器仅被某一个 MC 访问一次，数据中心是四个 MC 的起点和终点，由数据中心必须被访问 4 次来约束：

$$\sum_{k=1}^{m}y_i^k=\begin{cases}m,i=0\\1,\ i=1,2,...,n\end{cases} \tag{10}$$

为了保证每个 MC 的充电路线都能形成一个圈，无向图中每个终点传感器只能有一个起点传感器和它相连；每个起点传感器只能有一个终点传感器和它相连：

$$\sum_{i=0}^{n} x_{ij}^{k} = y_{j}^{k}, (\forall j = 0,1,...,n; k = 1,2,...,m)$$
$$\sum_{j=0}^{n} x_{ij}^{k} = y_{i}^{k}, (\forall i = 0,1,...,n; k = 1,2,...,m) \quad (11)$$

规划过程中，因为三角形两边之和大于第三边，易得四个 MC 中只派出一个 MC 工作，而剩余 MC 在数据中心闲置是充电总路线距离最短的情况（见图 13）。为了满足问题三的要求保证每个 MC 均正常工作且使得，即每个 MC 都必须访问非数据中心外的至少一个节点，可以通过式(10)和式(11)条件相结合达成约束。

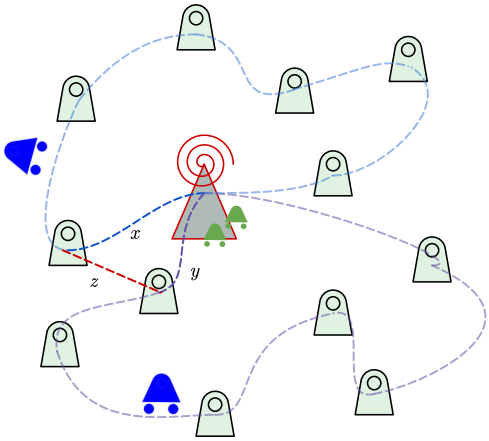

图 13 总距离上单 MC 优于多 MC 示意图

每个传感器集合的子集中都不能产生不与数据中心相连的支路，比如传感器集合 $R = \{a_1, a_2, a_3\}$ 产生闭路 $\{a_1 - a_2 - a_3 - a_4\}$，其中并没有与数据中心 a_0 相连，因此 $R = \{a_1, a_2, a_3\}$ 作为支路被消除[4]。消除图中其他支路的条件可表示为：

$$S = \left\{ \left(x_{ij}^{k}\right) | \sum_{i \in R} \sum_{j \in R} x_{ij}^{k} \leq |R| - 1, R \subseteq \{a_o, a_1, ..., a_n\} \right\}$$
$$X = \left(x_{ij}^{k}\right) \in S \quad (12)$$

（3）优化模型汇总

综合目标函数和约束条件的讨论，我们得到多 MC 移动充电路径规划的 0-1 规划模型如下：

目标函数： $\min Z = \sum_{i=1}^{k} z_k$

$$
s.t.\begin{cases}
z_k=\sum_{i=0}^{n}\sum_{j=0}^{n}d_{ij}x_{ij}^{k} & k=1,2,3,4\\
\sum_{i=0}^{n}x_{ij}^{k}=y_j^{k} & (\forall j=0,1,...,n;k=1,2,...,m)\\
\sum_{j=0}^{n}x_{ij}^{k}=y_i^{k} & (\forall i=0,1,...,n;k=1,2,...,m)\\
\sum_{k=1}^{m}y_i^{k}=\begin{cases}m,i=0\\1,\ i=1,2,...,n\end{cases}\\
S=\left\{\left(x_{ij}^{k}\right)|\sum_{i\in R}\sum_{j\in R}x_{ij}^{k}\leqslant|R|-1,R\subseteq\left\{a_o,a_1,...,a_n\right\}\right\}\\
X=\left(x_{ij}^{k}\right)\in S
\end{cases}
\tag{13}
$$

（4）利用模拟退火算法求解

根据前面建立的模型，考虑到多 MC 充电路径方案数量大，一般算法收敛速度慢，因此通过 MATLAB 软件利用模拟退火算法求解单 MC 的最短充电路径，算法流程图如图（如图 14）：

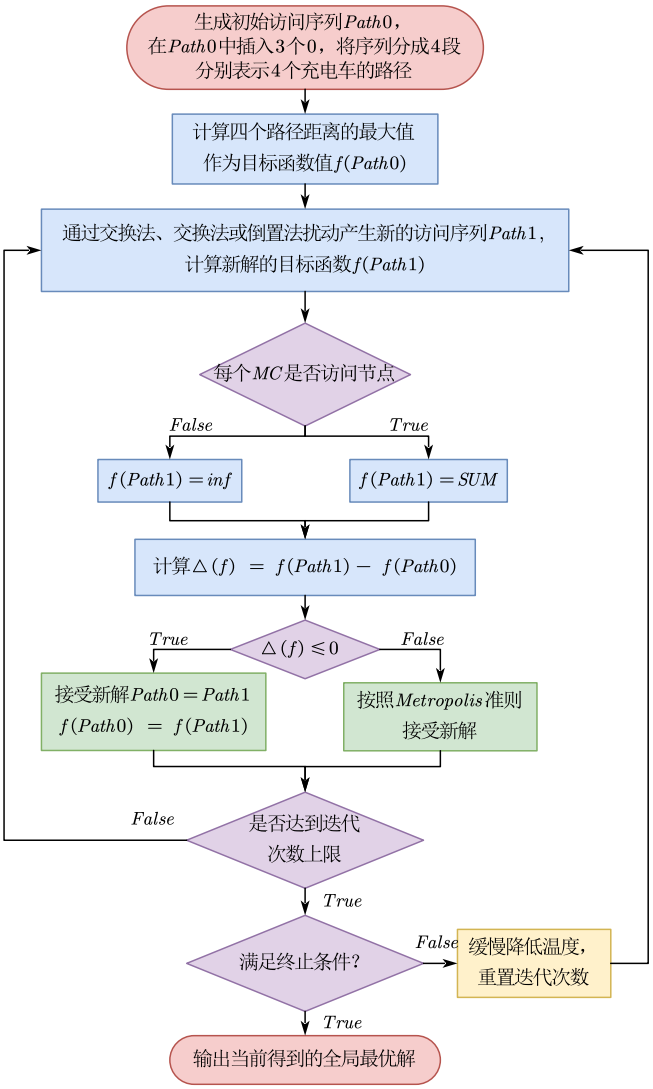

图 14 模拟退火算法流程图

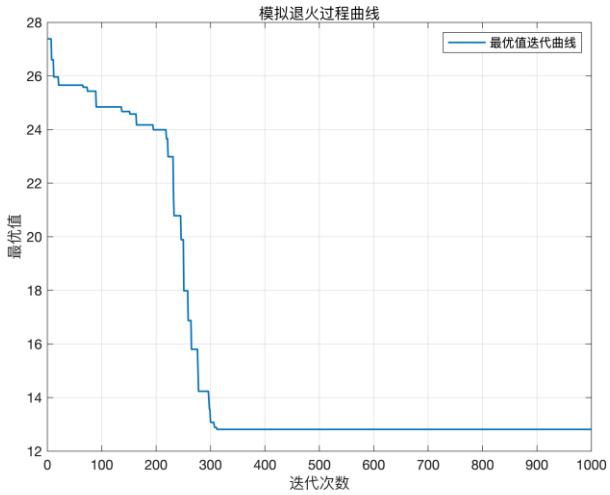

图 15　多 MC 移动充电路线规划图

如图 15，经过 313 次迭代得到最大耗电量的稳定值，即得到最优解，计算出 MC 最短的充电路线总路程为 12.756km，各 MC 的传感器访问路径如下（如图 16、表 4）：

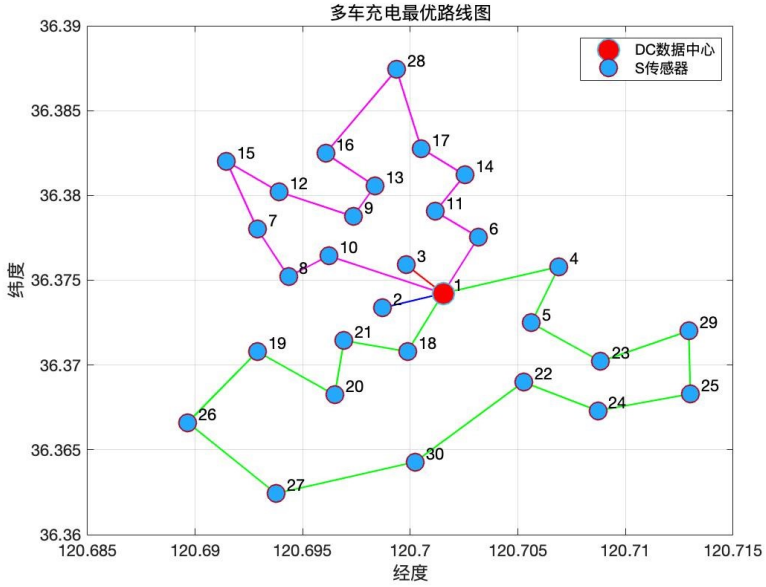

图 16　多 MC 移动充电路线规划图

表 4 多 MC 最短充电路径规划结果表

最短路线距离	12.756km
1 号 MC 路线	1→2→1
2 号 MC 路线	1→10→8→7→15→12→9→13→16→28→17→14→11→6→1
3 号 MC 路线	1→4→5→23→29→25→24→22→30→27→26→19→20→21→18→1
4 号 MC 路线	1→3→1

7.3 基于已得路径求解传感器电池容量

得到四个 MC 访问路径后，分别代入问题二中求解电池电量的程序利用计算机模拟的方法求解电池电量。

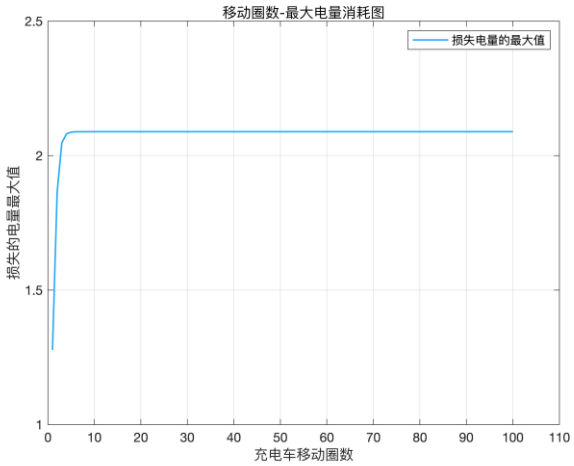

图 17 多 MC 移动充电模式下最大消耗电量随 MC 行驶圈数变化图

由上图（如图 17）可得，多 MC 移动充电模式下经过 5 圈最大消耗电量即达稳定，并且最大消耗电量为 2.0886 mA，比单 MC 移动充电模式下最大消耗电量小得多，这是由于多 MC 充电时，四个 MC 的最短路径均包含数量较少的传感器，减少了行驶路程的同时大大减少了路径中各传感器的耗电时间，因此最大耗电量较单 MC 模式的小得多。

将最大耗电量代入式(6)，即可求得多 MC 移动充电模式下传感器网络每个传感器电池容量为 2.6108 mA。

八、模型的评价与推广

8.1 模型的优点

1.利用计算机模拟方法模拟了完整的传感器耗电、MC 移动充电过程。

2.问题二通过充电速率和 MC 移动速率对模型算法进行了合理性检验，验证了模型的合理性和结果的可靠性。

3.根据移动充电器的充电特征，将所有传感器始终不低于电量阈值的电池电量问题，转化为求耗电最多的传感器电量问题，符合现实应用场景，同时简化了问题。

8.2 模型的缺点

1.确定参数时具有一定的主观性，与真实数值存在一定的误差。

2. 规划模型较为繁琐，不易理解；因此， 寻找更加简洁的规划刻画各个调度关系，是模型改进的方向。

8.3 模型的推广

问题一与问题三中充电路线的规划问题实际上是 TSP 问题，是已被论证的 NP-C 问题，本文利用 0-1 变量，根据题目要求建立了约束方程组来求解 TSP 问题，模型简单易懂，思路清晰，可根据具体的要求进行改进，同样适用于生活中各种最短路径问题。

在问题三派出四个移动充电器充电的过程中，除了使四个移动充电器在路中总的能量消耗最小之外，还可以考虑使四个移动充电器的总充电时间最短，四个充电器的充电任务平衡等目标。

参考文献

[1]司守奎, 孙兆亮, 孙玺菁. 数学建模算法与应用[M]. 国防工业出版社, 2015.
[2]刘康勇. 无线可充电传感器网络的移动充电规划方法研究[D].桂林理工大学,2019.
[3]罗卢杨,龙继东,唐小军.灾情巡视路线寻优模型[J].数学的实践与认识,1999(01):3-5.
[4]胡士娟. 基于改进遗传算法的多旅行商问题的研究[D].江南大学,2019.

附录

程序编号	T1	文件名称	TSP.lg4	程序说明	分支定界计算单车最短距离

```
model:
 title 充电路线规划问题;
 sets:
 sensor/1..30/:s;                    !定义 30 个传感器;
 distance(sensor,sensor):d,x;   !定义节点间距离 d、是否选择 ij 路径 x;
 endsets
 data:
 d = @ole('C:\Users\dell\Desktop\data\第一问','distance')
 !用@ole 函数从 excel 中定义的数据块读取数据并赋值给变量 d;
enddata
 n=@size(sensor);                    !将传感器总数赋值给变量 n;
 min=@sum(distance:d * x);       !目标函数：总距离最短;
 @for(sensor(k):
 @sum(sensor(i)|i#ne#k:x(i,k))=1;   !一个节点只有一个入边;
 @sum(sensor(j)|j#ne#k:x(k,j))=1;   !一个节点只有一个出边;
 @for(sensor(j)|j#gt#1#and#j#ne#k:
 s(j)>=s(k)+x(k,j)-
 (n-2)*(1-x(k,j))+
 (n-3)*x(j,k)));
 @for(distance:@bin(x));                !定义 0-1 变量，是否选取 ij 路径;
 @for(sensor(k)|k#gt#1:
 s(k)<=n-1-(n-2)*x(1,k);
 s(k)>=1+(n-2)*x(k,1));                !消除支路约束;
end
```

程序编号	T2	文件名称	main2.m	程序说明	计算路径距离

```
%% 读取数据
load('Data.mat');
Path = MinPath;Path=[Path,1];d=distance;
de = xlsread('C 题附件 2.xlsx');
de (:,1)=[];       % 剔除数据中心
de (:,1)=[];
%% 参数设定
MaxN = 100 ; % 充电车循环行走的次数
lv = 174; %充电的速率
mv = 40; %充电车移动的速度
ba = zeros(1,MoveV);
n = length(Path);
ne = zeros(1,n-1); %   每个结点当前电量
me = zeros(1,n-1); % 储存每个节点最小电量
```

```
me = 1./me; % 初始化最小值
lmin = zeros(1,MaxN); % 每一圈中电量损失最大值
nltime = zeros(1,30); % 存储每个节点充电的最大时间
%%   充电车移动求解最大电量消耗值
for iter = 1:MaxN
      tatime = 0; % 记录当前环行总时间
      for i = 2 : n % 访问所有节点
            % 此节点消耗时间加上移动至当前节点花费的时间
            tstime = d( Path(i-1) , Path(i) ) / mv;
            tatime = tatime + tstime;
            if i ~= n
                  % 减去访问此节点前消耗的总时间
                ne( Path(i) ) = ne( Path(i) ) - tatime * de( Path(i) );
                me( Path(i) ) = min( me( Path(i) ) ,   ne( Path(i) ) ); %更新最小值
                lmin(iter) = min( lmin(iter) , me( Path(i) ) ); % 更新当前遍历中的最小值
                ltime = ( abs( ne( Path( i ) ) )) / lv;        % 当前点充电的时间
                nltime(i) = max(ltime,nltime(i));
                tstime = tstime + ltime;        % 本次的时间加上充电的时间
                tatime = tatime + ltime;   % 总时间加上本次充电的时间
                ne( Path(i) ) = 0;          % 完成充电,电量损失为 0
            end
            for j = 2 : i-1 %之前的所有节点减去当前点花费的时间
                  ne( Path(j) ) = ne( Path(j) ) - tstime * de( Path(j) );
            end

%% 当前节点前减去这个节点充电时间和抵达路径时间和
            for j = 2 : i-1
                  me( Path(j) ) = min( me ( Path(j) ) , ne ( Path(j) ) );
                  lmin(iter) = min( lmin(iter) , me(Path(j)) ); %更新最小值
            end

      end
end
%%   最小值数据化可视化
x= 1:1:MaxN;
lmin=lmin.*(-1)
plot(x,lmin,'linewidth',1.3,'color',[36,169,255]/255);
title('移动圈数-最大电量消耗图');
legend('损失电量的最大值');
axis([0 110 95 600]);
xlabel('充电车移动圈数');ylabel('损失的电量最大值');
grid on
%% 求电池电量
% 每个的电量不能低于 20%
```

```
maxltime = max(nltime)
lose = max(lmin)
ba = lose / 0.8
```

程序编号	T3-1	文件名称	main3.m	程序说明	计算和最小的多充电车距离

```
clc;clear;
load('Data.mat')
[MinPath,MinDistance,record] = Min_MTsp(29 , 4 ,distance);
fprintf('求得的最短路线距离为      :      %d\n',MinDistance);
for i = 1 : length(MinPath)
    MinPath(i) = MinPath(i) + 1;
end
MinPath
index = find(MinPath == 1);

%% 分别得到 4 条路径的值
Path1 = MinPath(1:index(1)-1);Path1 = [1,Path1];
Path1(Path1==1) =[];
Path1 = [1,Path1];Path1 = [Path1,1]

Path2 = MinPath(index(1)+1:index(2)-1);Path2 = [1,Path2];
Path2(Path2==1) =[];
Path2 = [1,Path2];Path2=[Path2,1]

Path3 = MinPath(index(2)+1:index(3)-1);Path3 = [1,Path3];
Path3(Path3==1) =[];
Path3 = [1,Path3]; Path3=[Path3,1]

Path4 = MinPath(index(3)+1:end);Path4 = [1,Path4];
Path4(Path4==1) =[];
Path4 = [1,Path4];Path4=[Path4,1]

%% 画出 MTsp 路线图
figure(1);
length(Path1)-1
for a = 1:length(Path1)-1
    b = a+1;
    loc_a = location( Path1 (a) , : );    u_a = loc_a(1);        v_a = loc_a(2);
    loc_b = location( Path1 (b) , : );    u_b = loc_b(1);        v_b = loc_b(2);
    plot([u_a,u_b],[v_a,v_b],'-','linewidth',1.1,'color','b','HandleVisibility','off')
    hold on
end
length(Path2)-1
for a = 1:length(Path2)-1
```

```
        b = a+1;
        loc_a = location( Path2 (a) , : );      u_a = loc_a(1);          v_a = loc_a(2);
        loc_b = location( Path2 (b) , : );      u_b = loc_b(1);          v_b = loc_b(2);
        plot([u_a,u_b],[v_a,v_b],'-','linewidth',1.1,'color','r','HandleVisibility','off')
        hold on
end
length(Path3)-1
for a = 1:length(Path3)-1
        b = a+1;
        loc_a = location( Path3 (a) , : );      u_a = loc_a(1);          v_a = loc_a(2);
        loc_b = location( Path3 (b) , : );      u_b = loc_b(1);          v_b = loc_b(2);
        plot([u_a,u_b],[v_a,v_b],'-','linewidth',1.1,'color','g','HandleVisibility','off')
        hold on
end
length(Path4)-1
for a = 1:length(Path4)-1
        b = a+1;
        loc_a = location( Path4 (a) , : );      u_a = loc_a(1);          v_a = loc_a(2);
        loc_b = location( Path4 (b) , : );      u_b = loc_b(1);          v_b = loc_b(2);
        plot([u_a,u_b],[v_a,v_b],'-','linewidth',1.1,'color','m','HandleVisibility','off')
        hold on
end

for i=1:30
        text(location(i,1)+0.0005,location(i,2)+0.0005,num2str(i));
        hold on
end
hold on

plot(location(1,1),location(1,2),'o','MarkerSize',15,'MarkerFaceColor','r');hold on;
tmp=location(1,:);
location(1,:)=[];
plot(location(:,1),location(:,2),'o','MarkerSize',12,'MarkerFaceColor',[36,169,255]/255);
grid on
location=[tmp;location];
title('多车充电最优路线图');
xlabel('经度');ylabel('纬度');
legend('DC 数据中心','S 传感器');
%%   画出最优解随迭代次数的变化曲线
figure(2);
x=1:1:length(record);
plot(x,record,'linewidth',1.3);
grid on
title('模拟退火过程曲线');
```

```
legend('最优值迭代曲线');
xlabel('迭代次数');ylabel('最优值');

save('FourPath','Path1','Path2','Path3','Path4');
```

程序编号	T3-2	文件名称	Min_MTsp.m	程序说明	SA 计算 4 车距离最小路径

```
function [MinPath,MinMaxD,record] = Min_MTsp(city_num , salesman_num , d)
%% 参数初始化
    T0 = 1000;    % 初始温度
    T = T0;   % 迭代温度为初始温度
    MaxN = 1000;   % 最大迭代次数
    Lk = 500;    % 每个温度下的迭代次数
    alpha = 0.97;    % 温度衰减系数
%% 生成初始解的代码

path0 = randperm(city_num); % 1~n 的随机序列的初始路径
for i = 1:salesman_num-1       %path0 中随机插入 3 个 0
        len = length(path0); %计算此时 pat0 的长度,不断插入 0,长度不断增加
        ind = randi(len);      %生产个 1en 之间的随机数,ind 表示要插入 1 的位置
        path0 = [ path0(1:ind),0,path0(ind+1:end)]; %插入 0
end

%% 初始化用来保存中间结果的行走路径和距离的取值
iter_path = path0; %初始为迭代开始的初始路径
[path0_ , CarNum_ ] = DealPath(path0,salesman_num); %处理插入后的数据
[iter_result,~] = CalMTspDistance(path0_ , CarNum_ , d); % 计算当前路径的距离
%% 模拟退火过程
    record = [];
    for iter = 1 : MaxN   % 最大迭代次数
        MinResult = 10000000000;
        for i = 1 : Lk   %   每个温度迭代次数
            %处理插入后的数据
            [path0_ , CarNum_ ] = DealPath(path0,salesman_num);
            if CarNum_< 4
                result0=100000000;
            else
                % 计算当前路径的距离
                [result0 , ~] = CalMTspDistance(path0_ , CarNum_ , d);
            end
            MinResult = min(MinResult,result0);
            path1 = GetNewPath(path0);   % 生成新的路径
            [path1_,CarNum_] = DealPath(path1,salesman_num);
            if CarNum_< 4 % 设置惩罚
                result1=100000000;
```

```
                else
                    [result1 , ~] = CalMTspDistance(path1_,CarNum_,d);
                end
                MinResult = min(MinResult,result1);
                if ~isempty(record)
                    MinResult = min ( MinResult , record( length(record) ) );
                end
                if result1 < result0        % 新路径更优
                    path0 = path1; % 更新当前路径为新路径
                    iter_path = [iter_path; path1]; % 记录路径
                    iter_result = [iter_result; result1];    % 记录距离
                else
                    p = exp(-(result1 - result0)/T); % Metropolis 准则计算概率
                    if rand(1) < p      % 随机数和这个概率比较小于则接受劣解
                        path0 = path1;    % 更新当前路径为新路径
                    end
                end
            end
            record = [record,MinResult];
            T = alpha*T;      % 温度下降
        end
        [MinMaxD, idx] = min(iter_result);    % 找到最小距离的值及其下标
        MinPath = iter_path(idx,:); % 据下标找到最优路径
end
```

程序编号	T3-3	文件名称	DealPath .m	程序说明	根据隔板拆解得到 4 条路径

```
function [cpath, k] = DealPath(path, salesman_num)
    cpath = cell(salesman_num,1);            % 元胞数组,存储每个旅行商经过城市
    index = find(path == 0);       %     找到所有隔板位置
    k=1;
    for i = 1:salesman_num-1
        if i == 1          % 如果是第 1 个隔板
            c = path(i:index(i)-1); % 提取第一个旅行商经过的城市
        else
            c = path(index(i-1)+1:index(i)-1);% 提取中间的旅行商所经过的城市
        end
        if ~isempty(c)  % 只有 c 非空的话才保存
            cpath{k} = c; %把 c 保存到元胞 cpath 的第 k 个位置表示第 k 个旅行商的路径
            k = k+1;       %    参与工作的旅行商数目+1
        end
        end
            c = path( index(end) + 1 : end );  %     最后一个旅行商经过的城市
        if ~isempty(c)
            cpath{k} = c;
```

```
    else
        k = k-1;
    end
    cpath = cpath(1:k); %     只保留元胞中前 k 中非空的部分
end
```

程序编号	T3-4	文件名称	CalMTspDistance.m	程序说明	计算所有车距离和

```
function [cost, every_cost] = CalMTspDistance(cpath, k, d)
    every_cost = zeros(k,1);   %初始化每个旅行商的路程长度全为 0
    for i = 1:k        %对每个旅行商开始循环
        path_i = cpath{i};    %第 i 个旅行商所经过的路线
        n = length(path_i); %第 i 个旅行商经过的城市的数量
        if n>=26 %  惩罚因子
            every_cost(i) = every_cost(i) + 100000;
        elseif n==1 && path_i(1) == 0
            every_cost(i) = every_cost(i) + 100000;
        elseif n==0
            every_cost(i) = every_cost(i) + 100000;
        end
        for j = 1:n
            if j == 1% d 中的第一个位置表示起始的城市
                every_cost(i) = every_cost(i) + d(1,path_i(j) + 1);
            else
                every_cost(i) = every_cost(i) + d( path_i (j-1) + 1,path_i(j)+1);
            end
        end
        %  加上从最后一个城市返回到起始城市的路程
            every_cost(i) = every_cost(i) + d(path_i(end)+1,1);
    end
      cost = sum(every_cost);
end
```

程序编号	T3-5	文件名称	GetNewPath.m	程序说明	产生新路径

```
function path1 = GetNewPath(path0)
    % path0:  原来的路径
    n = length(path0);
    %  随机选择三种产生新路径的方法
    p1 = 0.33;    %  交换法产生新路径的概率
    p2 = 0.33;    %  移位法产生新路径的概率
    r = rand(1); %  随机生成一个[0 1]间均匀分布的随机数
%%  使用交换法产生新路径
    if r < p1
      c1 = randi(n);      % 1~n 中的随机整数
      c2 = randi(n);
```

```
        path1 = path0;   % 将 path0 的值赋给 path1
        path1(c1) = path0(c2);   %path1 第 c1 位置的元素为 path0 第 c2 位置的元素
        path1(c2) = path0(c1);   %path1 第 c2 位置的元素为 path0 第 c1 位置的元素
        path1(path1==1)=[];
        path1=[1,path1];
%% 使用移位法产生新路径
    elseif r < p1+p2
        c1 = randi(n);   % 生成 1-n 中的一个随机整数
        c2 = randi(n);
        c3 = randi(n);
        sort_c = sort([c1 c2 c3]);   % 对 c1 c2 c3 从小到大排序
        c1 = sort_c(1);   c2 = sort_c(2);   c3 = sort_c(3); %顺序大小满足 c1<=c2<=c3
        tem1 = path0(1:c1-1);
        tem2 = path0(c1:c2);
        tem3 = path0(c2+1:c3);
        tem4 = path0(c3+1:end);
        path1 = [tem1 tem3 tem2 tem4];
        path1(path1 == 1)=[];
        path1=[1,path1];
%%  使用倒置法产生新路径
    else
        c1 = randi(n);    % 生成 1-n 中的一个随机整数
        c2 = randi(n);    % 生成 1-n 中的一个随机整数
        if c1 > c2   % 如果 c1 比 c2 大，就交换 c1 和 c2 的值
            tem = c2;
            c2 = c1;
            c1 = tem;
        end
        tem1 = path0(1:c1-1);
        tem2 = path0(c1:c2);
        tem3 = path0(c2+1:end);
        path1 = [tem1 fliplr(tem2) tem3];
        % 改变移动顺序产生新路径
        path1(path1==1)=[];
        path1=[1,path1];
    end
end
```

程序编号	T3-6	文件名称	main3_2.m	程序说明	计算四车路径最小电池容量

```
clc;clear;
load('FourPath.mat');
load('Data.mat');d=distance;
de = xlsread('C 题附件 2.xlsx');
```

```
de (:,1)=[];       % 剔除掉出发点
de (:,1)=[];
%% 参数设定
MaxN = 100 ; % 充电车循环行走的次数
lv = 174; %充电的速率
mv = 40; %充电车移动的速度
n=31;
ne = zeros(1,n-1); %   每个结点当前的电量
mine = zeros(1,n-1); % 储存每个节点的最小电量
mine = 1./mine; % 初始化最小值
lmin = zeros(1,MaxN); % 存储每个
nltime = zeros(1,30); % 存储每个节点充电的最大时间
%%   充电车移动求解最大电量消耗值
for iter = 1:MaxN
    tatime = 0; % 记录当前环行走消耗的总时间
%% 第一个 充电车的行驶路径
    for i = 2 : length(Path1) % 访问所有节点
        tstime = d( Path1(i-1) , Path1(i) ) / mv;    % 移动到当前点花费的时间
        tatime = tatime + tstime; % 累计总时间
        if i ~= n
            ne( Path1(i) ) = ne( Path1(i) ) - tatime * de( Path1(i) ); % 减去总时间
            mine( Path1(i) ) = min( mine( Path1(i) ) ,    ne( Path1(i) ) ); %更新最小值
            lmin(iter) = min( lmin(iter) , mine( Path1(i) ) ); % 更新最小值
            ltime = ( abs( ne( Path1( i ) ) )) / lv;         % 当前点充电的时间
            nltime(i) = max(ltime,nltime(i));
            tstime = tstime + ltime;         % 本次的时间加上充电的时间
            tatime = tatime + ltime;    % 总时间加上本次充电的时间
            ne( Path1(i) ) = 0;          % 完成充电后当前节点无电量损失
        end
        for j = 2 : i-1 %之前的所有节点减去当前点花费的时间
            ne( Path1(j) ) = ne( Path1(j) ) - tstime * de( Path1(j) );
        end
        for j = 2 : i-1 %更新最小值
            mine( Path1(j) ) = min( mine ( Path1(j) ) , ne ( Path1(j) ) );
            lmin(iter) = min( lmin(iter) , mine(Path1(j)) ); %更新当前圈数的最小值
        end
    end
%% 第二个 充电车的行驶路径
    tatime = 0; % 记录当前环行走消耗的总时间
    for i = 2 : length(Path2) % 访问所有节点
        tstime = d( Path2(i-1) , Path2(i) ) / mv; %从上一个点移动到当前点花费的时间
        tatime = tatime + tstime; % 累计总时间
        if i ~= n
            ne( Path2(i) ) = ne( Path2(i) ) - tatime * de( Path2(i) ); % 减去总时间
```

```
                mine( Path2(i) ) = min( mine( Path2(i) ) ,    ne( Path2(i) ) ); %更新最小值
                lmin(iter) = min( lmin(iter) , mine( Path2(i) ) ); %  更新最小值
                ltime = ( abs( ne( Path2( i ) ) )) / lv;          %  当前点充电的时间
                nltime(i) = max(ltime,nltime(i));
                tstime = tstime + ltime;          %  本次的时间加上充电的时间
                tatime = tatime + ltime;    %  总时间加上本次充电的时间
                ne( Path2(i) ) = 0;           %  完成充电后当前节点无电量损失
            end
            for j = 2 : i-1 %之前的所有节点减去当前点花费的时间
                ne( Path2(j) ) = ne( Path2(j) ) - tstime * de( Path2(j) );
            end
            for j = 2 : i-1 %更新最小值
                mine( Path2(j) ) = min( mine ( Path2(j) ) , ne ( Path2(j) ) );
                lmin(iter) = min( lmin(iter) , mine(Path2(j)) ); %更新最小值
            end
        end
%% 第三个  充电车的行驶路径
        tatime = 0; %  记录当前环行走消耗的总时间
        for i = 2 : length(Path3) %  访问所有节点
            tstime = d( Path3(i-1) , Path3(i) ) / mv; %从上一个点移动到当前点花费的时间
            tatime = tatime + tstime; %  累计总时间
            if i ~= n
                ne( Path3(i) ) = ne( Path3(i) ) - tatime * de( Path3(i) ); %  减去总时间
                mine( Path3(i) ) = min( mine( Path3(i) ) ,    ne( Path3(i) ) ); %更新最小值
                lmin(iter) = min( lmin(iter) , mine( Path3(i) ) ); %  更新最小值
                ltime = ( abs( ne( Path3( i ) ) )) / lv;          %  当前点充电的时间
                nltime(i) = max(ltime,nltime(i));
                tstime = tstime + ltime;          %  本次的时间加上充电的时间
                tatime = tatime + ltime;    %  总时间加上本次充电的时间
                ne( Path3(i) ) = 0;           %  完成充电后当前节点无电量损失
            end
            for j = 2 : i-1 %之前的所有节点减去当前点花费的时间
                ne( Path3(j) ) = ne( Path3(j) ) - tstime * de( Path3(j) );
            end
            for j = 2 : i-1 %更新最小值
                mine( Path3(j) ) = min( mine ( Path3(j) ) , ne ( Path3(j) ) );
                lmin(iter) = min( lmin(iter) , mine(Path3(j)) ); %更新当前圈数的最小值
            end
        end
%% 第四个  充电车的行驶路径
        tatime = 0; %  记录当前环行走消耗的总时间
        for i = 2 : length(Path4) %  访问所有节点
            tstime = d( Path4(i-1) , Path4(i) ) / mv; %从上一个点移动到当前点花费的时间
            tatime = tatime + tstime; %  累计总时间
```

```
            if i ~= n
                ne( Path4(i) ) = ne( Path4(i) ) - tatime * de( Path4(i) ); % 减去总时间
                mine( Path4(i) ) = min( mine( Path4(i) ) ,   ne( Path4(i) ) ); %更新最小值
                lmin(iter) = min( lmin(iter) , mine( Path4(i) ) ); % 更新最小值
                ltime = ( abs( ne( Path4( i ) ) )) / lv;        % 当前点充电的时间
                nltime(i) = max(ltime,nltime(i));
                tstime = tstime + ltime;        % 本次的时间加上充电的时间
                tatime = tatime + ltime;   % 总时间加上本次充电的时间
                ne( Path4(i) ) = 0;          % 完成充电
            end
            for j = 2 : i-1 %之前的所有节点减去当前点花费的时间
                ne( Path4(j) ) = ne( Path4(j) ) - tstime * de( Path4(j) );
            end
            for j = 2 : i-1 %更新最小值
                mine( Path4(j) ) = min( mine ( Path4(j) ) , ne ( Path4(j) ) );
                lmin(iter) = min( lmin(iter) , mine(Path4(j)) ); %更新当前圈数的最小值
            end
        end
end
%%   最小值数据化可视化
figure
x = 1:1:MaxN;
lmin = lmin.*(-1);
plot(x,lmin,'linewidth',1.3,'color',[36,169,255]/255);
title('移动圈数-最大电量消耗图');
legend('损失电量的最大值');
axis([0 110 20 70]);
xlabel('充电车移动圈数');ylabel('损失的电量最大值');
grid on
%% 求电池电量
% 每个的电量不能低于 20%
Maxltime = max(nltime)
Lose = max(lmin)
Battery = Lose / 0.8
```