基于路径优化的医疗物资配送模型

摘要

本文针对货车给医院送物资的最小路径问题，基于 TSP 问题，目标优化，容量约束建立了数学模型，为货车周游医院的路径规划进行了优化，给出了给定交通道路约束下的最短路径规划方案。在疫情的背景下，进一步提高了物流调度的效率以及医疗物资保障能力。

对于问题一，首先我们利用决策变量结合单仓库旅行商问题建立一个简单的数学模型。之后，我们利用贪心策略中改进过的最邻近法再依次对 14 个节点进行求解，最后在所求解中找出最小的解，这个解就是所求的最优路径，总路程大小 131.5864，最优路径：节点5→ 节点13 → 节点10 → 节点1 → 节点8 → 节点9 → 节点7 → 节点3 → 节点6 → 节点4 → 节点12 → 节点2 → 节点14 → 节点11 → 节点5(顺序可逆)。

对于问题二，可以在问题一的基础上扩展为单仓库多旅行商问题，建立一个优化模型。之后，我们先忽略物资因素的影响使用遗传算法算出可能的最优路径集合，然后再验证是否满足物资需求条件，在满足条件的结果中选取最优值。最后我们计算出的最小总路程为 158.9719，最优路径：节点5→ 节点4 → 节点6 → 节点3 → 节点7 → 节点9 → 节点8 → 节点1 → 节点10 → 节点5和节点5 → 节点13 → 节点11 → 节点14 → 节点12 → 节点2 → 节点5(顺序可逆)。

对于问题三，由于交通道路的限制，不能直接采用问题一和问题二的数学建模方法。可以采用带约束的 TSP 或 VRP 的数学建模方法，我们为了简化模型，排除了有内环和多个过渡医院的情况，然后在问题二求解的基础上增加约束，最后求出的最优解为 281.7508，最优路径：节点5→ 节点7 → 节点8 → 节点9 → 节点5和节点5 → 节点1 → 节点11 → 节点12 → 节点13 → 节点14 → 节点10 → 节点4 → 节点6 → 节点3 → 节点2 → 节点5(顺序可逆)。

对于问题四，可以划分为带约束的多仓库车辆路径问题。其与问题三的数学模型基本一致，但是在求解的过程中，由于两辆货车的仓库不同，行驶的灵活度更高，我们为了减小误差，将可接受过渡医院的个数调整为两个，并用一个三维矩阵存储数据。在遗传算法的基础上，依次将各个节点作为新仓库求解最短距离，最后求得将节点 9 所对于医院作为医疗物资的第二仓库，可以得到最短路径为 205.0228，且最优路径不止一条。

关键词: 贪心算法　最短路径　遗传算法　目标优化

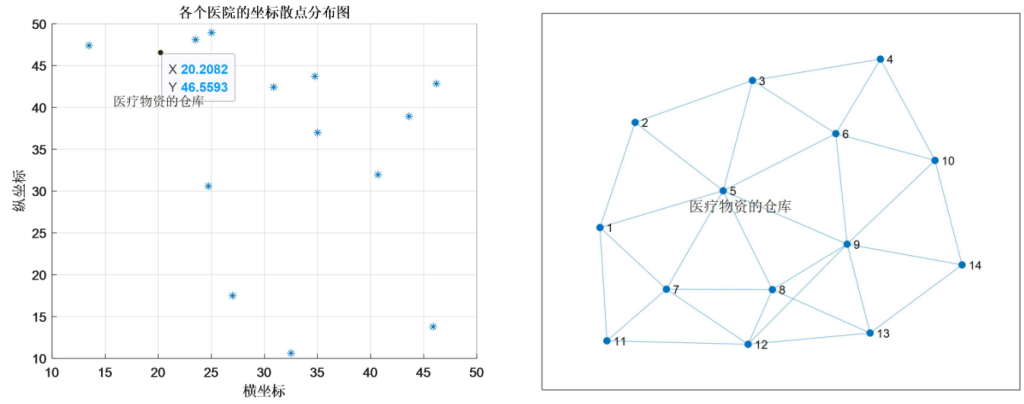
一、问题重述

1.1 问题背景

新冠肺炎疫情发生以来，部分防疫一线重点医疗物资供应告急。坚决遏制疫情蔓延势头，短时间内提升供给保障能力成为当务之急。全国各地都在进行防疫的攻坚战，而在防疫攻坚的过程中有很多医疗物资需要运送。医疗物资保障能力的提升，为彰显中国速度、中国力量，打赢疫情防控阻击战提供了强有力保障。

1.2 问题重述

现在考虑某个城市的 14 个医院，这些医院的坐标位置和日需求量均存放在附件中，且医疗物资存放在节点 5 的仓库中，各个医院和物资仓的散点坐标图和直达道路拓扑关系图如下图所示：



(a) 各个医院和物资仓的散点坐标图　　(b) 各个医院的直达道路拓扑关系图

图 1: 总图标题

我们通过分析相关数据，运用数学思想，建立数学模型来研究医疗物资的配送有关的下列问题：

问题一：假设任意两个节点之间均可以直线到达，请设计一个路径规划方案，使得能够运载 800 吨货物的货车从仓库出发能够一次性到达所有的医院后返回仓库且所经过的总路程最小。

问题二：在问题一的假设下，如果有两辆能够运载 500 吨货物的货车，要求同时从仓库出发最后回到仓库且一次性到达所有的医院，请设计一个路径规划方案，使得两辆货车的总路程最小。

问题三：在现实中，由于交通道路和城市布局的限制，医院与医院之间不可能都直达，有直达道路的节点对是有限的。根据附件所给的医院之间的直达道路存在关系以及图二所给的拓扑关系。在问题三的前提下重新设计一个路径规划方案，使得两辆能够运载 500 吨货物的货车同时从仓库出发最后都回到仓库且两辆货车的总路程最小。

问题四：在问题三的前提下，可以选择哪一家医院作为医疗物资的第二仓库，使得两辆能够运载 500 吨货物的货车分别从两个仓库驶出再回到各自起点且遍历所有医院的总路程能够最短。

二、问题分析

根据对题目的理解，我们知道问题的求解是在满足每题要求以及医院实际所需医疗物资的情况下，要设计一个最优的路径规划方案，使得货车所经过的总路程最小，下面我们对每一个问题进行具体分析。

2.1 问题一的分析

针对问题一，根据附件中各个医院的医疗物资日需求量，可以确定运载 800 吨货物的货车可以满足所有目标医院的总需求量。可以采用旅行商问题的数学建模方法，我们以任意两个医院之间的距离为权值，构建一个完备图，从而使用贪心算法中的最邻近法和 matlab 软件，确定出一条总路程最短的路径。

2.2 问题二的分析

针对问题二，货车的数量从一辆增加为两辆，在满足医院医疗物资日需求量的前提下可以将其转化为 MTSP 问题。之后，我们结合单仓库多旅行商解决方法建立目标模型，在求解时先忽略医疗物资这个影响因素，结合遗传算法确定出使得总路程较小的路径集合，再从中挑选出满足物资需求的最优的方案。

2.3 问题三的分析

针对问题三，可以采用带有容量约束的 VRP 的数学建模方法，具体地，可以将医院之间的道路表示为边，将每个医院的需求量视为点权，将货车的容量限制表示为容量约束，问题就可以转化为求解一组满足容量约束的最短路径。

2.4 问题四的分析

针对问题四，结合多仓库车辆路径问题的建模方法，逐层增加约束来筛选最优路径，依次将其余医院作为第二仓库，从计算出的最优路径中取出最小值，这条路径所对应的医院即为目标医院。

三、模型假设

1. 忽略医疗物资在运输过程中的损耗。
2. 假设货车在运输过程中没有意外情况的发生，如：动力不足，车辆故障，交通事故等。
3. 假设医院之间所有的直达道路均为直线道路。
4. 认为附件中的所有数据均真实可靠。

四、符号说明

符号	说明
L	最优路径总路程
M_k	K 号车的载货量
d_{ij}	节点 i 与节点 j 的距离
m_i	节点 i 所对应医院的医疗物资日需求量
q_{ki}	k 号车给节点 i 所对应医院的物资供给量
Q_{ki}	k 号车在到达节点 i 所在医院时已经消耗的物资量

五、模型的建立与求解

5.1 问题一模型的建立与求解

5.1.1 模型的准备

1. 数据可视化

根据附件的数据，我们可以将医院坐标投射到平面坐标轴上，以便更为直观的观察各个医院和物资仓（代表各个节点）的分布特点。

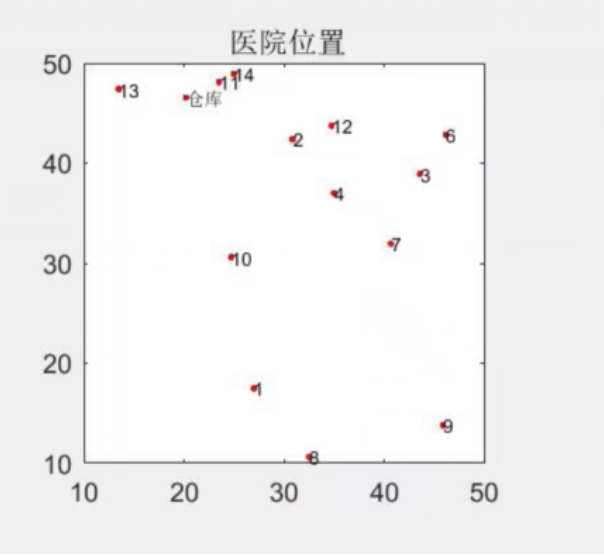


图 2: 各个医院的坐标散点分布图

2. 最优路径的定义

从图论的角度来看，该问题可以转化为是给定一个带权完全无向图（顶点表示医院，边表示直达道路，权重是道路距离），找一个权值最小的 Hamilton 回路。我们将满足这样条件的回路称为“最优路径”。

5.1.2 模型的建立

1. 节点距离 d_{ij}

我们认为，当最优路径中的两个相邻节点之间的距离越小时，货车所经过的总路程越小，两个节点之间的距离是影响最优路径规划下的总路程大小的主要因素。

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{1}$$

式中，(x_i, y_i) 表示节点 i 的坐标，(x_j, y_j) 表示节点 j 的坐标。

2. 决策变量 a_{ij}

为了优化最优路径中的直达道路与其他直达道路的数据处理，从而求出最优路径，我们引入一个 0-1 决策变量。

$$a_{ij} = \begin{cases} 1, \text{节点 i 与节点 j 之间的直达道路在最优路径上} \\ 0, \text{其他} \end{cases} \tag{2}$$

有了上面的一些定义之后，我们可以建立目标规划模型。

3. 目标函数 L 的确定

对于该题而言，我们的目标非常明确，是使货车的总路程 L 最小。

$$L = \min \sum_{i \neq j}^{14} a_{ij} d_{ij} \tag{3}$$

4. 约束条件的确定

• 决策变量的约束

对于某一个固定 i 的节点而言，有且仅有两条与其相连的直达道路在最优路径上。因此，与节点 i 有关的决策变量满足

$$\sum_{j \neq i}^{14} a_{ij} = 2 \tag{4}$$

• 医疗物资的需求量

根据实际需要，在货车运输的过程中，其所运载医疗物资的总量可以满足所有医院的日需求量。

$$\sum_{i=1}^{14} m_i \leq M \tag{5}$$

式中 M 是货车的载货量，m_i 是节点 i 所对应的医院医疗物资的日需求量。

综上，得到最优路径问题的目标优化模型：

目标函数：

$$L = \min \sum_{i \neq j}^{14} a_{ij} d_{ij} \tag{6}$$

$$s.t. \begin{cases} L = \min \sum\limits_{i \neq j}^{14} a_{ij} d_{ij} \\ \sum\limits_{i=1}^{14} m_i \leq M \\ \mathrm{a}_{ij} \in (0,1) \end{cases} , i = 1, 2, ..., 14, j = 1, 2, ..., 14 \tag{7}$$

5.1.3 模型的求解

由于该模型约束条件少，附件中数据量小，所以直接求解难度不大。因此，我们考虑使用贪心算法来简化运算，即从当前节点下遍历所有能到达的下一节点，选择距离最近的节点作为下一节点。基本思路是从物流仓这个节点开始出发遍历所有能到达的下一节点，选择距离最近的节点作为下一节点，然后把当前节点标记已走过，下一节点作为当前节点，重复贪心策略，以此类推直至所有节点都标记为已走节点结束。

为了更加准确的求出最优路径，我们对贪心策略进行改进，对每一个固定节点 i 我们都将其作为起始节点进行上述运算，计算出所对应的最优路径，循环多次，从所有结果中选取最优结果。具体方法如流程图所示：

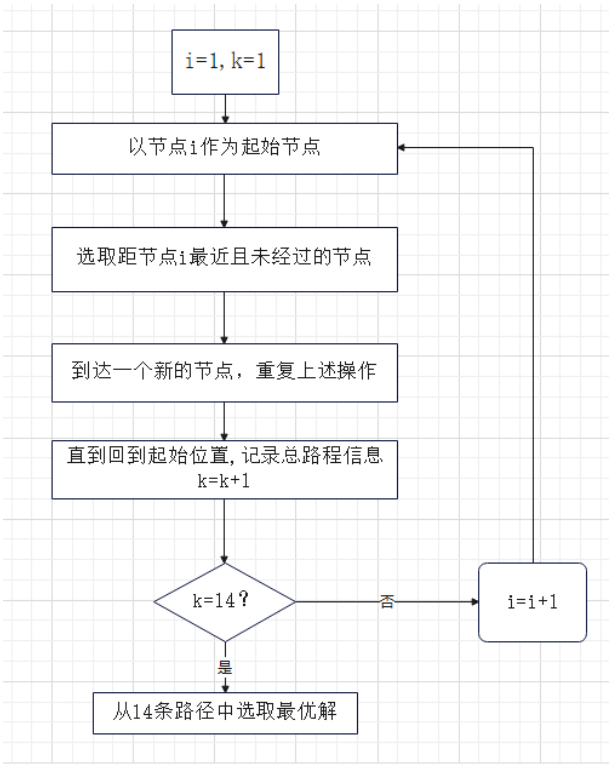


图 3: 问题一改进最邻近点算法过程

通过多次实验，我们得到的最优结果如下图所示：

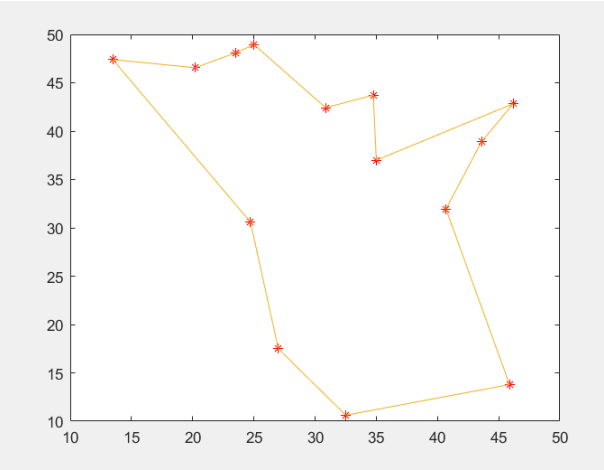


图 4: 问题一 matlab 最优路径图示

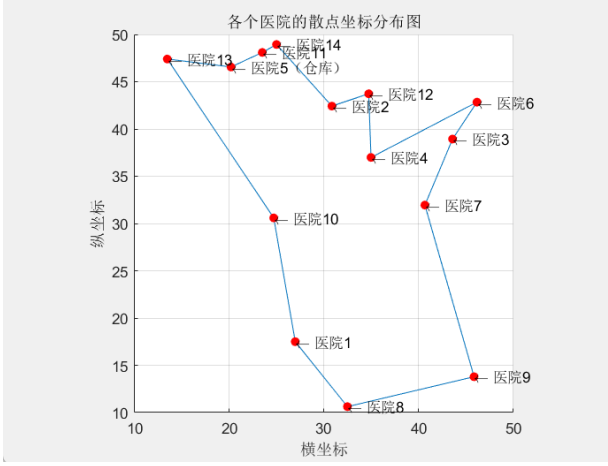


图 5: 问题一 matlab 最优路径示意图

通过分析与计算，最优路径的节点编号依次为:

- 5→ 13 → 10 → 1 → 8 → 9 → 7 → 3 → 6 → 4 → 12 → 2 → 14 → 11 → 5
- 5→ 11 → 14 → 2 → 12 → 4 → 6 → 3 → 7 → 9 → 8 → 1 → 10 → 13 → 5

在该方案下计算所得的最小总路程为 131.5864。

5.2 问题二模型的建立与求解

5.2.1 模型的准备

首先将两辆货车分别编号为 1 号车和 2 号车。其次需要指出的是，我们不考虑节点 5 所对应医院的医疗物资分配，这是因为两辆货车均会回到节点 5 且其医疗物资剩余总量大于节点五所对应医院的日需求量，可以认为其最后分配。最后需要注意的是，在最优路径中，1 号车与 2 号车除节点 5 以外没有共同经过的节点。下面我们来证明这一结论。

假设存在最优路径使得 1 号车与 2 号车存在除节点 5 以外的另一共同经过节点 i。由于两辆车均经过节点 i 且每辆车的路线可以看作一个闭环，不妨设两辆车均是最后到达节点 i，可以推出两辆车在到达节点 i 时其剩余运载物资量均小于节点 i 所对应医院的日需求量。否则其中剩余运载物资量小的车可以跳过节点 i，直接前往对应闭环的下一点，根据三角形两边之和大于第三边，可以得到更优的路径，这与假设矛盾！推出：

$$M_1 - N_1 < m_i \tag{8}$$

$$M_2 - N_2 < m_i \tag{9}$$

式中 M_i 表示 i 号车载货量，N_i 表示闭环 i 上所有医院的总日需求量，m_i 表示节点 i 所对应医院的日需求量。

将（8）（9）式相加：

$$M_1 + M_2 - (N_1 + N_2) < 2m_i \tag{10}$$

结合问题本身的数据与实际，不难发现 M_1=M_2=500 且：

$$N_1 + N_2 = \sum_{j \neq i}^{14} m_j - m_i \tag{11}$$

式中 m_j 表示节点 j 所对应医院的日需求量, 结合附件可知 $\sum_{j \neq i}^{14} m_j$=762。

最终推出存在节点 i，其对应医院物资的日需求量：

$$m_i > 238 \tag{12}$$

这与附件中的实际数据不符，故最初的假设不成立。基于在最优路径中，1 号车与 2 号车除节点 5 以外没有共同经过的节点这一事实，认为在最优路径下两辆货车分别的路径可以看作两个仅有一个公共节点 5 的闭环，并成为闭环 1 与闭环 2（节点与道路的集合）。下面我们开始对问题二进行建模。

5.2.2 模型的建立

1. 指标函数

为了明确各个医院与货车之间的对应运输关系，我们引入一个指标函数来判断医院的物资来源：

$$x_{ki} = \begin{cases} 1, \text{节点 i 所对应医院的物资来自 k 号车} \\ 0, \text{其他} \end{cases} \tag{13}$$

上式展现了每个节点与货车之间的对应关系，我们还同样可以引入另一个指标函数用来表明每一条直达道路与货车之间的关系，用于判断货车是否经过该道路：

$$y_{ij}^{k} = \begin{cases} 1, \text{k 号车从节点 i 通过直达道路到达节点 j} \\ 0, \text{其他} \end{cases} \tag{14}$$

2. 目标函数 L 的确定

最优路径要求 1 号车与 2 号车的总路程最小，直观的来说是要求两个闭环所包含的直达道路距离总和最小。注意到，对任意的 i，j$\in N, k \in K$

$$L_{ij} = y_{ij}^{k} d_{ij} \tag{15}$$

的值可以反应出节点 i 与节点 j 之间的直达道路为最优路径 L 的贡献。

其中，N={1,2,3,4,5,6,7,8,9,10,11,12,13,14} ,K={1,2}。

从而确定目标函数：

$$L = \min \sum_{i \in N} \sum_{j \in N} \sum_{k \in K} y_{ij}^{k} d_{ij} \tag{16}$$

4. 约束条件的确定

• 医疗物资的供给量

由于每辆货车所在闭环的节点所对应的医院物资均由该货车单独供给，且根据前文分析，不存在供给不足的情况且不考虑节点 5，可以断定闭环中所有的节点所对应医院的物资总日需求量不大于 k 号车的载货量，即：

$$\sum_{i \in N_0} m_i x_{ki} \leq M_k \tag{17}$$

式中 M_k 表示 k 号车的载货量,$N_0 = N \setminus \{5\}$.

• 指标函数的关系

我们设置的两个指标函数分别表示节点与货车，道路与货车之间的关系，实际上指标函数之间也存在一定数量关系，可以发现如果节点 h 所在医院的物资由 k 号车运送，那么 k 号车已经到达的上一个节点 i 和即将到达的下一个节点 j 在闭环 k 上，同时节点 h 与这两个节点之间的直达道路均在最优路径上。

$$\sum_{i \in N} y_{ih}^{k} = \sum_{j \in N} y_{hj}^{k} = x_{kh} \tag{18}$$

对任意的 k$\in K, h \in$$N_0$ 成立。

• 闭环的约束

由于我们知道最优路径必然是由两个仅有一个公共点节点 5 的闭环组成，所以对于 N_0 集合中的所有节点是不存在闭环的。对任意的子集 S$\subset$$N_0$，均有

$$\sum_{i \in S} \sum_{j \in S} y_{ij}^{k} \leq |S| - 1 \tag{19}$$

式中 $|S|$ 表示集合元素个数。

综上，我们得到最优路径问题的优化模型：

目标函数：

$$L = \min \sum_{i \in N} \sum_{j \in N} \sum_{k \in K} y_{ij}^{k} d_{ij} \tag{20}$$

$$s.t.\begin{cases} \sum\limits_{i\in N_0} m_i x_{ki} \le M_k \\ \sum\limits_{i\in N} y_{ih}^k = \sum\limits_{j\in N} y_{hj}^k = x_{kh} \\ \sum\limits_{i\in S}\sum\limits_{j\in S} y_{ij}^k \le |S|-1, S \subset \mathrm{N}_0 \quad ,k=1,2 \\ x_{ki} \in (0.1) \\ y_{ij}^k \in (0,1) \end{cases} \tag{21}$$

5.2.3 模型的求解

由于该模型约束条件较为复杂，直接求解难度较大。我们可以先不考虑医院物资需求量对结果的影响，假设每个医院所需物资为零，再这样的假设下求解出使得最优路径的总路程较短的一部分数据，然后回归实际，逐一验证这些数据是否满足前文所述医疗物资供给量这个要求，这些满足要求的数据中的最小值就是我们所求的结果。

具体思路如下：我们通过遗传算法，并对节点的序号进行二进制编码，每两个节点之间都可以互通，两个节点之间的距离用两点之间的距离直接衡量，最后的总距离为整个路径总长，并以此为适应度值，由此问题转化为求解最小路径总和。，取得所有合适路径，每条路径迭代形成对应的 2 个闭环，并将所有的解进行一个比较：将每条路径的总长度和每辆货车供应量都保存下来，然后利用一个自己构建的 findminsuit 函数，去选择在每辆货车供应量满足条件的情况下的路径长度最小的方案，然后将最优路径输出和绘制出来。

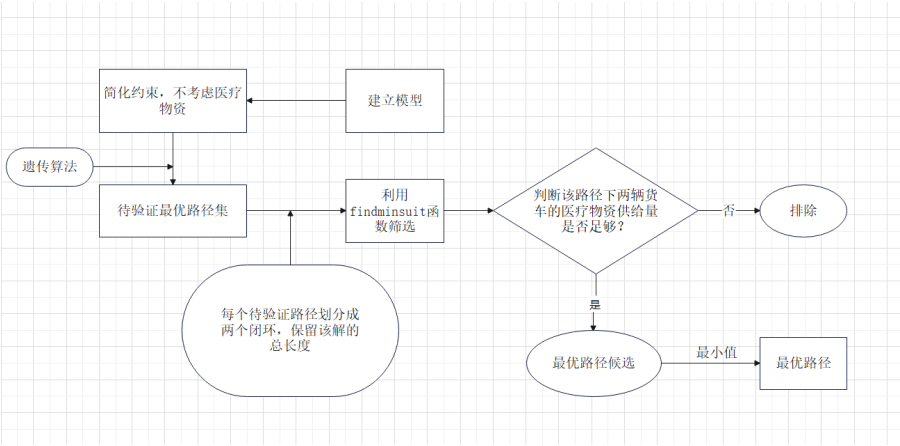


图 6: 问题二求解算法过程

在使用 matlab 多次调整待验证路径后，总计迭代 240 次，从所有结果中选取最优结果，此时最优路径的总路程为 158.9719。最优路径如下图所示：

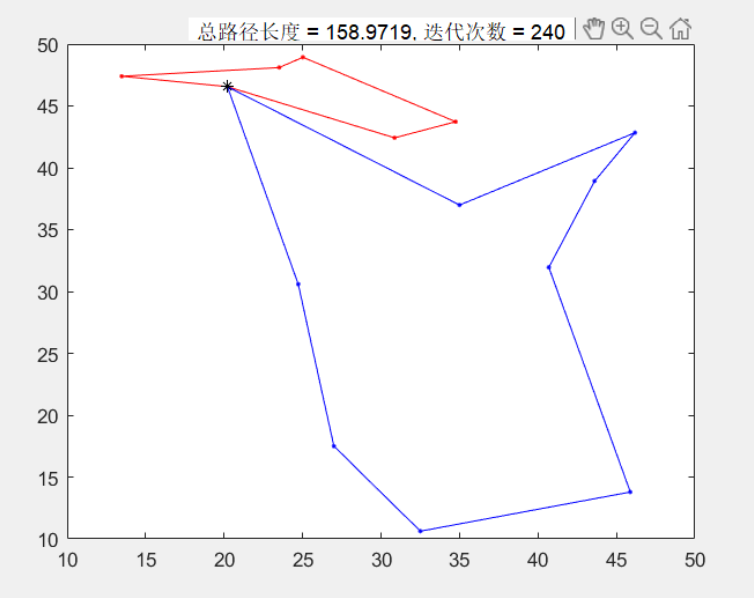


图 7: 问题二 matlab 最优路径图示

由图示可以看出，问题二最优路径如下（顺序可逆）:

1 号车：5→ 4 → 6 → 3 → 7 → 9 → 8 → 1 → 10 → 5

2 号车：5→ 13 → 11 → 14 → 12 → 2 → 5

5.3 问题三模型的建立与求解

5.3.1 模型的准备

1. 指示变量 c_{ij}

节点与节点之间可能存在直达道路，也有可能不存在直达道路，所以我们引入一个指示变量来判断这件事情

$$c_{ij} = \begin{cases} 1, \text{节点 i 与节点 j 有直达道路} \\ 0, \text{节点 i 与节点 j 没有直达道路} \end{cases} \tag{22}$$

根据附件中的数据，我们可以确定所有的使得 c_{ij}=1 的无序对 (i，j)，设它们组成的集合为 P。

2. 无内环结构

我们断言：在最优路径中不存在内环结构，即一辆货车在运输过程中除节点 5 以外不会经过同一个节点两次。下面证明这一结论:

根据题目所给的各个医院的直达道路拓扑关系图:

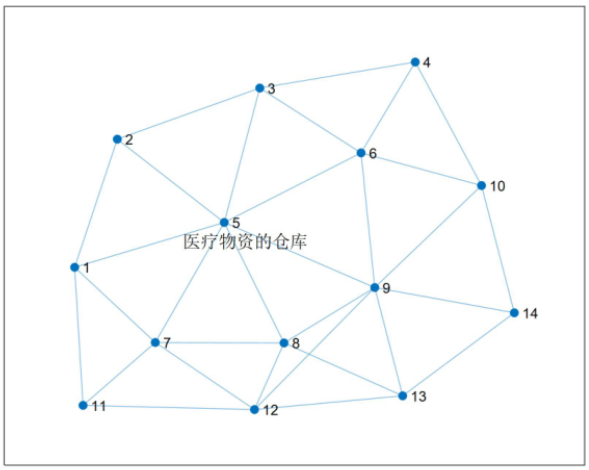


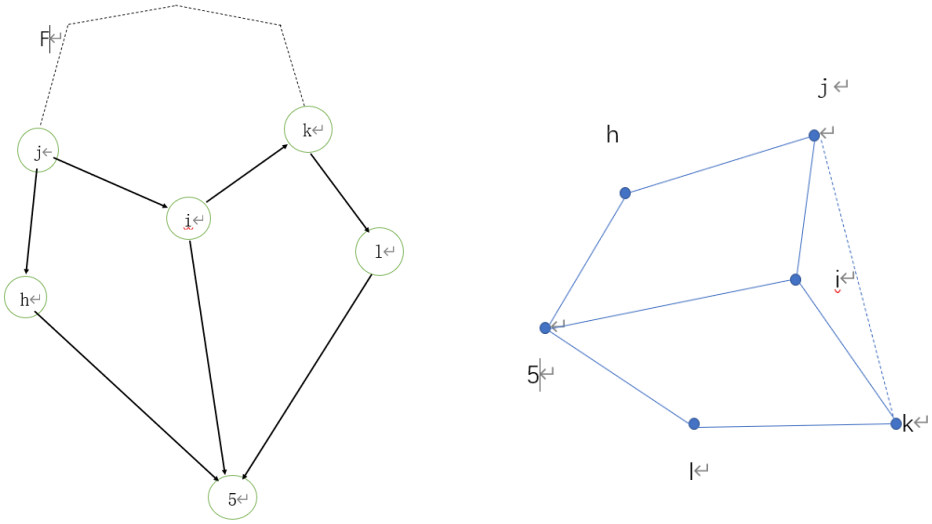
图 8: 各个医院直达道路拓扑关系图

我们将与节点 5 有直达道路的节点称为一级点，其余点称为二级点，假设现在存在节点 i，货车及经过节点 i 两次，那么此时会有一个闭环的结构，我们将闭环上与节点 i 相邻的两个节点设为节点 j 和节点 k。不妨设货车的行驶方向为节点 j 到节点 i 到节点 k，货车在节点 j 返回仓库的过程中，此时节点 i 已经运送过物资，所以从节点 j 再次经过节点 i 的影响因素是距离，即从节点 j 经过节点 i 返回节点 5 这条路径比从节点 j 返回节点 5 另外的路径更短。接下来两种情况讨论：

节点 i 是一级点：若此时节点 j 也是一级点，有

$$d_{j5} \leq d_{ji} + d_{i5} \tag{23}$$

与上述矛盾，所以可以确定节点 j 是二级点，同理，节点 k 也是二级点。那么这些点之间的拓扑关系图可以如下图表示；



(a) 环结构图示　　(b) 局部节点的直达道路拓扑关系图

图 9: 总图标题

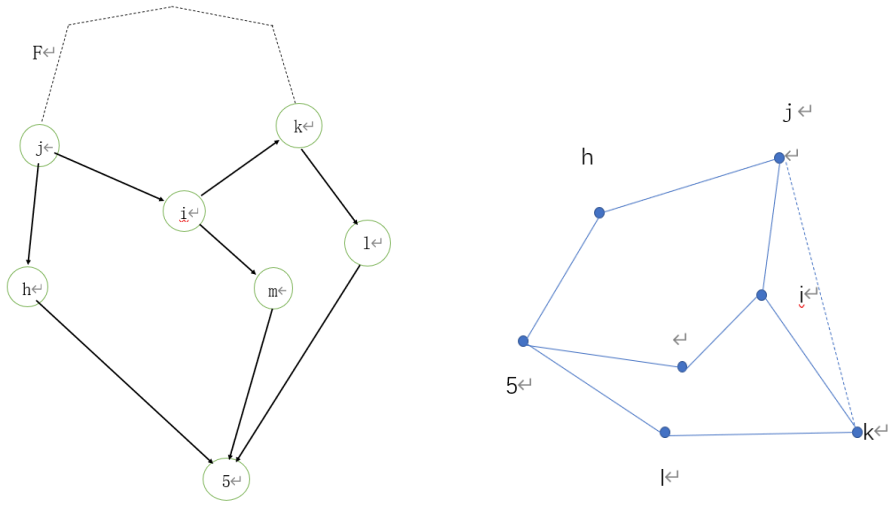
图中节点 h 和节点 m 均为一级点, 在结合整体的拓扑关系图可以找到 i 只能是 6，7，8，9。结合附件中的数据，我们计算以下距离：

表 1: 二级点与仓库的距离

节点顺序	距离大小
5 → 1 → 11	60.65
5 → 3 → 4	33.49
5 → 6 → 4	54.25
5 → 6 → 10	66.34
5 → 7 → 11	48.74
5 → 7 → 12	38.35
5 → 8 → 12	71.16
5 → 8 → 13	79.39
5 → 9 → 10	68.65
5 → 9 → 12	73.57
5 → 9 → 13	88.32
5 → 9 → 14	82.51

在逐一排除下，我们发现只有当 i=7 时可能有内环结构的存在。而在这种情况下，为了简化模型，我们可以认为是得不到最优路径的。

节点 i 是二级点：同理分析，可以得到下图所示的局部拓扑图：



(a) 环结构图示　　(b) 局部节点的直达道路拓扑关系图

图 10: 总图标题

结合整体与拓扑关系图验证后发现这样的局部拓扑关系是不存在的，综上所述，我们可以认为在最优路径中是没有内环结构的存在。

5.3.2 模型的建立

1. 目标函数 L 的确定

相比于问题二，问题三仅多了交通道路的限制，所以目标函数可以表示为：

$$L = \min \sum_{(i,j)\in P} \sum_{k\in K} y_{ij}^{k} d_{ij} \tag{24}$$

2. 约束条件的确定

• 所有医院的遍历

由于每一个医院都要求获得医疗物资，对某给固定的节点 i 而言，至少有一辆货车到达该节点

$$\sum_{j\in N_i} \sum_{k\in K} y_{ij}^{k} \geq 1 \tag{25}$$

式中 N_i=N\{i}.

• 确保无环结构

由前文的分析可知，问题三的最优路径中也是没有环结构的存在，与问题二的约束类似。对任意的子集 S⊂N_0，均有

$$\sum_{i\in S} \sum_{j\in S} y_{ij}^{k} \leq |S| - 1 \tag{26}$$

式中 $|S|$ 表示集合元素个数。

• 物资的供给关系

根据题目的要求，货车的供给量需要满足医院的日需求量：

$$\sum_{k\in K} q_{ki} = m_i \tag{27}$$

式中 q_{ki} 表示 k 号车给节点 i 所对应医院的物资供给量。

• 物资的容量限制

在货车运输的过程中，我们要确保货车的剩余物资量可以满足还未到达的医院的总需求量

$$Q_{ki} + y_{ij}^{k} q_{ki} - M_k(1 - y_{ij}^{k}) \leq Q_{kj} \tag{28}$$

$$q_{ki} \leq Q_{ki} \leq M_k \tag{29}$$

式中 Q_{ki} 表示 k 号车在到达节点 i 所在医院时已经消耗的物资量。

• 仓库的约束

我们认为，在最优的情况下，一辆货车不会重复经过仓库，只在开始运送与结束运送任务时到达

$$\sum_{k\in K} \sum_{i\in N_0} y_{5i}^{k} \leq 2 \tag{30}$$

综上所述，在增加道路约束的条件下我们建立的模型如下：

目标函数：

$$L = \min \sum_{(i,j)\in P} \sum_{k\in K} y_{ij}^{k} d_{ij} \tag{31}$$

$$s.t. \begin{cases} \sum\limits_{j\in N_i} \sum\limits_{k\in K} y_{ij}^{k} \geq 1 \\ \sum\limits_{i\in S} \sum\limits_{j\in S} y_{ij}^{k} \leq |S| - 1, S \subset \mathrm{N}_0 \\ \sum\limits_{k\in K} q_{ki} = m_i \\ Q_{ki} + y_{ij}^{k} q_{ki} - M_k (1 - y_{ij}^{k}) \leq Q_{kj} \quad , k = 1, 2 \\ \sum\limits_{k\in K} \sum\limits_{i\in N_0} y_{5i}^{k} \leq 2 \\ q_{ki} \leq Q_{ki} \leq M_k \\ y_{ij}^{k} \in (0, 1) \end{cases} \tag{32}$$

5.3.3 模型的求解

问题三的约束条件比较复杂，但仅仅只是在问题二的基础上增加了道路约束，原本可以走的最优路径可能会不完整，存在两个医院之间没有直达道路。为了优化模型，我们假设，在最优路径中，货车只可能为到达下一个目标医院走一次弯路，即通过一个别的过渡医院到达目标医院，这样我就可以认为两个不连通的医院的距离是过渡医院连接两者的总距离并记录下来，储存在 path 矩阵中。如果两个不连通的医院需要货车走多次弯路才可以到达，那么我们认为这两个医院的距离非常大，在 path 矩阵它们之间的距离当作一个很大的数值处理，在最优路径中不考虑。然后我们沿用问题二的求解思路：

1. 先假设每个医院之间具有直达道路，利用遗传算法求解出距离较短的路径集合。

2. 对于这些路径中不连通的节点距离，我们利用 path 矩阵中的数值进行替换，得到新的路径集合。

3. 在新的路径集合中验证医院的总需求量不大于货车的运载量，得到待确认最优路径集。

4. 在该集合中选出数值最小的路径，即为最优路径。

在 matlab 的多次模拟中，最后的得出的最短路径总路程为：281.7508.

如下图所示：

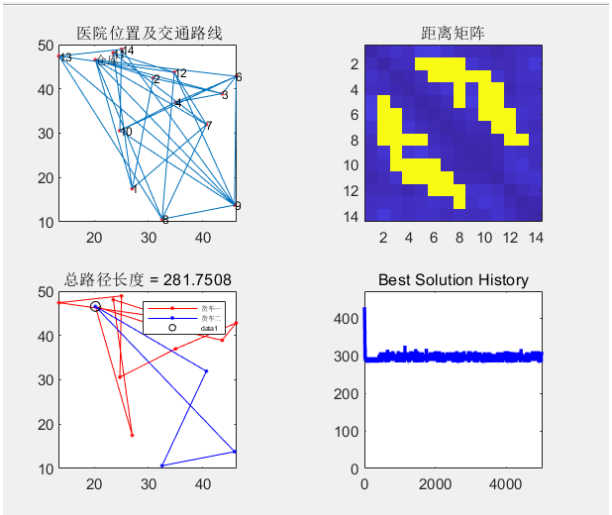


图 11: 问题三 matlab 最优路径示意图

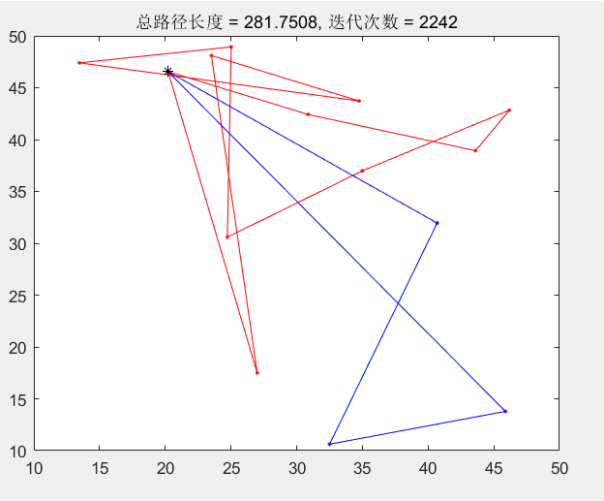


图 12: 问题三 matlab 最优路径图示

由图示可以看出，问题二最优路径如下（顺序可逆）：

1 号车：5→ 7 → 8 → 9 → 5 → 9 → 8 → 1 → 10 → 5

2 号车：5→ 1 → 11 → 12 → 13 → 14 → 10 → 4 → 6 → 3 → 2 → 5

5.4 问题四模型的建立与求解

模型的建立问题四与问题三的数学模型极其相似，可以认为，问题三是问题四的一种特殊情形，我们假设现在以某个固定的节点 h 所在医院作为第二仓库，1 号车从节点 5 出发，2 号车从节点 h 出发，记 N_t=$N\backslash\{5,h\}$.

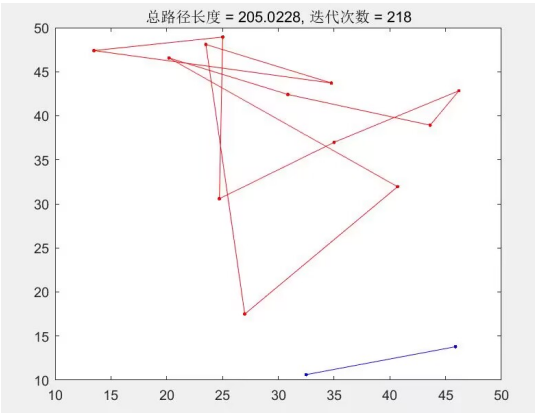
目标函数：

$$L = \min \sum_{(i,j)\in P} \sum_{k\in K} y_{ij}^{k} d_{ij} \tag{33}$$

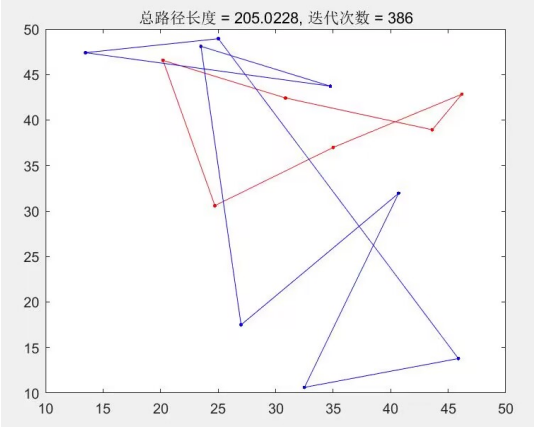
$$
s.t.\begin{cases}
\sum\limits_{j\in N_i}\sum\limits_{k\in K} y_{ij}^{k} \geq 1 \\
\sum\limits_{i\in S}\sum\limits_{j\in S} y_{ij}^{k} \leq |S|-1, S \subset \mathrm{N}_t \\
\sum\limits_{k\in K} q_{ki} = m_i \\
Q_{ki} + y_{ij}^{k} q_{ki} - M_k(1-y_{ij}^{k}) \leq Q_{kj} \quad , k = 1,2 \\
\sum\limits_{i\in N_t} y_{5i}^{1} \leq 1 \\
\sum\limits_{i\in N_t} y_{hi}^{2} \leq 1 \\
q_{ki} \leq Q_{ki} \leq M_k \\
y_{ij}^{k} \in (0,1)
\end{cases} \tag{34}
$$

模型的求解

在问题三的基础上，把 path 矩阵更新为允许有两家过渡医院的情况，在最优路径中，货车只可能为到达下一个目标医院走两次弯路，即通过两个别的过渡医院到达目标医院，这样我就可以认为两个不连通的医院的距离是过渡医院相继连接两者的总距离并记录下来，这时候 path 矩阵较小的值就代表是可以通过 0-2 家过渡医院到达的总路程，这时候我用了两个 path 矩阵进行记录，一个 path1 矩阵，一个 path2 矩阵。如果两个不连通的医院需要货车走多次弯路才可以到达，那么我们认为这两个医院的距离非常大，在 path 矩阵它们之间的距离当作一个很大的数值处理，在最优路径中不考虑。然后我们沿用问题三的求解思路，在对每个节点经过上百次循环计算后，最终得出以某个固定的节点作为第二仓库时的最优路径，如下图所示：



(a) 节点 9 作为第二仓库的最优路径 1



(b) 节点 9 作为第二仓库的最优路径 2

我们将所得的图示中的数据进行汇总，如下表所示：

表 2: 各个方案下的最短距离

第二仓库	最短总路程
节点1	218.2253
节点2	235.2258
节点3	230.2297
节点4	221.4509
节点6	225.5997
节点7	222.3721
节点8	212.2193
节点9	205.0228
节点10	226.1182
节点11	211.4820
节点12	223.5488
节点13	213.9642
节点14	213.6965

通过上表我们发现当以节点 9 所对应医院作为第二仓库时，最优路径的总路程最小，总路程为 205.0028。同时我们发现其对应的最优路径有两条。所以我们认为问题四的最优解为节点 9。

六、模型的评价与推广

6.1 模型的优点

1. 结合 TSP 模型和 VPR 模型进行建模，实用性强，具有良好的推广性。
2. 在建模过程中对某些结论进行严谨的证明与合理的假设。
3. 四个问题的模型联系紧密，层层递进，可解释较好。

6.2 模型的缺点

1. 算法上主要采用遗传算法，代码上存在缺陷，随机样例不够大，有时无法生成最优解，需多次运行。

2. 模型上的无内环假设以及需求量约束简化导致结果准确度降低。

3. 将所有医院分配给两辆货车进行周游，并且要求每辆货车的总路程最小，属于 NP-hard 问题，大型实例不能用精确算法求解，必须寻求这类问题的有效的近似算法。

6.3 模型的推广

本文建立的最优路径模型有效的找出在一定数量的车辆下，物流调度中的最短路径。除了车辆数量约束、载重量约束、运行路线约束，需求量约束以外，我们还可以在更复杂的背景下进行调整参数进行类似建模。在实际生活中，规划出最优路径，大大的降低了物流调度的实时性和运输成本。

参考文献

[1] 赖志柱，张云艳，戈冬梅. 求解 TSP 问题的改进最邻近法 [J] . 毕节学院报告, 2016（034）001.

[2] 浅谈旅行商（TSP）的七种整数规划模型

https://blog.csdn.net/weixin$_5$3463894/*article*/*details*/1282300172023325.

[3] 基于 matlab 遗传算法求解多车场带时间窗的车辆路径规划问题

https://blog.csdn.net/TIQCmatlab/article/details/118070129 2023 年 3 月 25 日.

[4] 姜启源，谢金星，叶俊. 数学模型（第四版）[M] . 北京：高等教育出版社，2011.

附录

注：为了编写代码的简便性，将仓库第五节点与第一节点的医院的位置进行了调换，所以在代码保存路径时，要将节点 1 看成仓库，节点 5 号看成节点 1 所对应的医院。

问题一的代码

```
clc,clear
%文件读取
T=readtable('data\B题附件.xlsx',"VariableNamingRule","preserve");
% 医院的坐标
Hos_Loc = [T{:,2},T{:,3}];
%初始化将各个节点作为出发点的路径信息和总路径长度
Path_total=zeros(size(Hos_Loc,1),size(Hos_Loc,1)+1);
distance_total=zeros(size(Hos_Loc,1),1);
%以第i个节点作为出发点
for k=1:size(Hos_Loc)
%此处将节点1与节点k坐标信息进行调换
m = k; n = 1;
Hos_Loc = [T{:,2},T{:,3}];
Hos_Loc([m n],:) = Hos_Loc([n m],:);
% 计算医院之间的距离矩阵
distances = pdist2(Hos_Loc, Hos_Loc);
% 设置起点
start_Hos = 1;
% 初始化路径和总路程
path = start_Hos;
total_distance = 0;
% 迭代找到最短路径
while length(path) < size(Hos_Loc, 1)
    % 获取当前路径的最后一所医院
    current_Hos = path(end);
    min=1000;
    % 找到距离最短的下一所医院
    for i=1:size(Hos_Loc,1)
    if distances(current_Hos,i)<min&&distances(current_Hos,i)~=0&&~ismember(i,path)
    min =distances(current_Hos,i);
    next_Hos=i;
    end
    end
```

```
    % 如果下一所医院不在当前路径中，则将其加入路径，并记录该路径
    if ~ismember(next_Hos, path)
        path = [path, next_Hos];
        % 更新总路程
        total_distance = total_distance + distances(current_Hos, next_Hos);
    end
end
% 最后回到起点形成闭环
total_distance = total_distance + distances(path(end), start_Hos);
path = [path, start_Hos];
%将此次路径信息和总路程信息保存
for t=1:size((Hos_Loc),1)+1
    Path_total(k,t)=path(1,t);
end
distance_total(k,1)=total_distance;
end
%寻找最短总路程的回路
Mindist=distance_total(1,1);
index=1;
for i=2:size(Hos_Loc)
    if distance_total(i,1)<Mindist
        Mindist=distance_total(i,1);
        index=i;
    end
end
%更新路径信息（还原）
X=T{:,2};
Y=T{:,3};
for i=1:size((Hos_Loc),1)+1
    if Path_total(index,i)==index
            Path_total(index,i)=1;
    end
end
Path_total(index,1)=index;
Path_total(index,size((Hos_Loc),1)+1)=index;
%在散点图中连接路径
for i=1:size((Hos_Loc),1)
   line([X(Path_total(index,i),1),X(Path_total(index,i+1),1)],[Y(Path_total(index,i)
end
```

```
axis square
hold on
grid on
%绘制散点图
scatter(X,Y,'filled','r');xlabel('横坐标');ylabel('纵坐标');title('各个医院的散点坐标
    text(X(1,1),Y(1,1),'\leftarrow␣医院1');
    text(X(2,1),Y(2,1),'\leftarrow␣医院2');
    text(X(3,1),Y(3,1),'\leftarrow␣医院3');
    text(X(4,1),Y(4,1),'\leftarrow␣医院4');
    text(X(5,1),Y(5,1),'\leftarrow␣医院5（仓库）');
    text(X(6,1),Y(6,1),'\leftarrow␣医院6');
    text(X(7,1),Y(7,1),'\leftarrow␣医院7');
    text(X(8,1),Y(8,1),'\leftarrow␣医院8');
    text(X(9,1),Y(9,1),'\leftarrow␣医院9');
    text(X(10,1),Y(10,1),'\leftarrow␣医院10');
    text(X(11,1),Y(11,1),'\leftarrow␣医院11');
    text(X(12,1),Y(12,1),'\leftarrow␣医院12');
    text(X(13,1),Y(13,1),'\leftarrow␣医院13');
    text(X(14,1),Y(14,1),'\leftarrow␣医院14');
% 打印结果
fprintf('最短路径： ');
disp(Path_total(index,:));
fprintf('总路程： %.2f\n', Mindist);
```

问题二的代码

```
%% 获得参数
% n = 14;
% T=readtable('data\B题附件.xlsx',"VariableNamingRule","preserve");
% Need_Load=xlsread('data\B题附件.xlsx','各个节点物资的日需求量');
% t=Need_Load(2,1);
% Need_Load(2,1)=Need_Load(2,5);
% Need_Load(2,5)=t;
% Hos_Loc= [T{:,2},T{:,3}];
% Hos_Loc([5 1],:) = Hos_Loc([1 5],:);
% Hos_dist = pdist2(Hos_Loc, Hos_Loc);%计算距离矩阵。
% nVehicle = 2;
% min_tour = 1;
% pop_size = 160;
```

```
% num_iter = 5e3;
% a = meshgrid(1:n);
% [opt_route,opt_break,min_dist] = mtspf_ga_2(Hos_Loc,Hos_dist,nVehicle,min_tour, Ne
%%
function varargout = mtspf_ga_2(Hos_Loc,Hos_dist,nVehicle,min_tour,Need_Load,pop_siz
%参数扫描
[N,~] = size(Hos_Loc);
[nr,nc] = size(Hos_dist);
if N ~= nr || N ~= nc
    error('ERROR!');
end
n = N - 1;
% 数据检查
nVehicle = max(1,min(n,round(real(nVehicle(1)))));
min_tour = max(1,min(floor(n/nVehicle),round(real(min_tour(1)))));
pop_size = max(8,8*ceil(pop_size(1)/8));
num_iter = max(1,round(real(num_iter(1))));
show_prog = logical(show_prog(1));
show_res = logical(show_res(1));

% 初始化路径分隔点
num_breaks = nVehicle-1;
dof = n - min_tour*nVehicle;  % 自由医院数目
addto = ones(1,dof+1);
for k = 2:num_breaks
    addto = cumsum(addto);%累加
end
cum_prob = cumsum(addto)/sum(addto);

%初始化种群
pop_route = zeros(pop_size,n);           % 路径
pop_break = zeros(pop_size,num_breaks);   %分割点
for k = 1:pop_size
    pop_route(k,:) = randperm(n)+1;
    pop_break(k,:) = randbreaks();
end

% 选择两条路径的颜色
clr = [1 0 0; 0 0 1];
```

```
% 运行GA遗传算法
global_min = Inf;
totalLoad=zeros(2,pop_size);
total_dist = zeros(1,pop_size);
dist_history = zeros(1,num_iter);
tmp_pop_route = zeros(8,n);
tmp_pop_break = zeros(8,num_breaks);
new_pop_route = zeros(pop_size,n);
new_pop_break = zeros(pop_size,num_breaks);
if show_prog
    pfig = figure('Name','MTSPF_GA | Current Best Solution','Numbertitle','off');
end
for iter = 1:num_iter
    % 计算每一个货车的总行驶距离
    for p = 1:pop_size
        d = 0;
        p_route = pop_route(p,:);
        p_break = pop_break(p,:);
        rng = [[1 p_break+1];[p_break n]]';%代表每个旅行商的起点和终点
        for s = 1:nVehicle%计算这一种群的总距离
            d = d + Hos_dist(1,p_route(rng(s,1)));
            load=0; %记录货车补给消耗量
            for k = rng(s,1):rng(s,2)-1%计算旅行商中间的距离
                d = d + Hos_dist(p_route(k),p_route(k+1));
                 load=load+Need_Load(2,p_route(k));
            end
            load=load+Need_Load(2,p_route(rng(s,2)));%添加最后一次补给量
            d = d + Hos_dist(p_route(rng(s,2)),1); %添加返回的最后一段距离
            totalLoad(s,p)=load;
        end
        total_dist(p) = d;%每个货车的路径总长度
    end
    % 在种群中找到最优路径，若存在则画出
 [min_dist,index] = find_min_suit(total_dist,totalLoad);%调用函数
    dist_history(iter) = min_dist;
    if min_dist < global_min
        global_min = min_dist;
        opt_route = pop_route(index,:);
```

```
    opt_break = pop_break(index,:);
    rng = [[1 opt_break+1];[opt_break n]]';
    if show_prog
        %绘制最优路线
        figure(pfig);
        for s = 1:nVehicle
            route = [1 opt_route(rng(s,1):rng(s,2)) 1];
            plot(Hos_Loc(route,1),Hos_Loc(route,2),'.-','Color',clr(s,:));
            title(sprintf('总路径长度=%1.4f,迭代次数=%d',min_dist,iter));
            hold on
        end
        plot(Hos_Loc(1,1),Hos_Loc(1,2),'k*');
        hold off
    end
end
% 采用锦标赛方法进行选择，随机选择8个个体选择出最优个体作为父代
rand_grouping = randperm(pop_size);
for p = 8:8:pop_size
    routes = pop_route(rand_grouping(p-7:p),:);
    breaks = pop_break(rand_grouping(p-7:p),:);
    distances = total_dist(rand_grouping(p-7:p));
    [~,idx] = min(distances);
    best_of_8_route = routes(idx,:);
    best_of_8_break = breaks(idx,:);
    route_ins_pts = sort(ceil(n*rand(1,2)));
    I = route_ins_pts(1);
    J = route_ins_pts(2);
    for k = 1:8
        tmp_pop_route(k,:) = best_of_8_route;
        tmp_pop_break(k,:) = best_of_8_break;
        switch k
            case 2 % 翻转
                tmp_pop_route(k,I:J) = fliplr(tmp_pop_route(k,I:J));
            case 3 % 交换
                tmp_pop_route(k,[I J]) = tmp_pop_route(k,[J I]);
            case 4 % 迁移
                tmp_pop_route(k,I:J) = tmp_pop_route(k,[I+1:J I]);
            case 5 % 修改间隔点
                tmp_pop_break(k,:) = randbreaks();
```

```
                case 6
                    tmp_pop_route(k,I:J) = fliplr(tmp_pop_route(k,I:J));
                    tmp_pop_break(k,:) = randbreaks();
                case 7
                    tmp_pop_route(k,[I J]) = tmp_pop_route(k,[J I]);
                    tmp_pop_break(k,:) = randbreaks();
                case 8
                    tmp_pop_route(k,I:J) = tmp_pop_route(k,[I+1:J I]);
                    tmp_pop_break(k,:) = randbreaks();
                otherwise
            end
        end
        new_pop_route(p-7:p,:) = tmp_pop_route;
        new_pop_break(p-7:p,:) = tmp_pop_break;
    end
    pop_route = new_pop_route;
    pop_break = new_pop_break;
end
if show_res
    % 绘制多图像
    figure('Name','MTSPF_GA␣|␣Results','Numbertitle','off');
    subplot(2,2,1);
    plot(Hos_Loc(:,1),Hos_Loc(:,2),'r.');
    title('␣医院位置');
    text(Hos_Loc(1,1),Hos_Loc(1,2),'仓库','FontSize',7);
    text(Hos_Loc(2,1),Hos_Loc(2,2),'2','FontSize',7);
    text(Hos_Loc(3,1),Hos_Loc(3,2),'3','FontSize',7);
    text(Hos_Loc(4,1),Hos_Loc(4,2),'4','FontSize',7);
    text(Hos_Loc(5,1),Hos_Loc(5,2),'1','FontSize',7);
    text(Hos_Loc(6,1),Hos_Loc(6,2),'6','FontSize',7);
    text(Hos_Loc(7,1),Hos_Loc(7,2),'7','FontSize',7);
    text(Hos_Loc(8,1),Hos_Loc(8,2),'8','FontSize',7);
    text(Hos_Loc(9,1),Hos_Loc(9,2),'9','FontSize',7);
    text(Hos_Loc(10,1),Hos_Loc(10,2),'10','FontSize',7);
    text(Hos_Loc(11,1),Hos_Loc(11,2),'11','FontSize',7);
    text(Hos_Loc(12,1),Hos_Loc(12,2),'12','FontSize',7);
    text(Hos_Loc(13,1),Hos_Loc(13,2),'13','FontSize',7);
    text(Hos_Loc(14,1),Hos_Loc(14,2),'14','FontSize',7);
    axis square
```

```
    subplot(2,2,2);
    imagesc(Hos_dist([1 opt_route],[1 opt_route]));
    title('距离矩阵');
    axis square
    subplot(2,2,3);
    rng = [[1 opt_break+1];[opt_break n]]';
    for s = 1:nVehicle
        route = [1 opt_route(rng(s,1):rng(s,2)) 1];
        plot(Hos_Loc(route,1),Hos_Loc(route,2),'.-','Color',clr(s,:));
        legend('货车一','货车二','FontSize',4);
        title(sprintf('总路径长度 = %1.4f',global_min));
        hold on;
    end
    axis square
    plot(Hos_Loc(1,1),Hos_Loc(1,2),'ko');
    subplot(2,2,4);
plot(dist_history,'b','LineWidth',2);
    title('Best Solution History');
    set(gca,'XLim',[0 num_iter+1],'YLim',[0 1.1*max([1 dist_history])]);
    axis square
end
% Return Outputs
if nargout
    varargout{1} = opt_route;
    varargout{2} = opt_break;
    varargout{3} = min_dist;
end
%若对路程距离有要求，则在染色体前面位置选择分隔点的概率会偏低一点
%没有要求则随机生成间隔点
    function breaks = randbreaks()
        if min_tour == 1
            tmp_breaks = randperm(n-1);
            breaks = sort(tmp_breaks(1:num_breaks));
        else
            num_adjust = find(rand < cum_prob,1)-1;
            spaces = ceil(num_breaks*rand(1,num_adjust));
            adjust = zeros(1,num_breaks);
            for kk = 1:num_breaks
                adjust(kk) = sum(spaces == kk);
```

```
            end
            breaks = min_tour*(1:num_breaks) + cumsum(adjust);
        end
    end
end

function [min_dist,index]=find_min_suit(total_dist,totalLoad)
[x,y]=min(total_dist);
while totalLoad(1,y)<192||totalLoad(1,y)>500
    total_dist(y)=1000;
    [x,y]=min(total_dist);
end
    min_dist=x;
    index=y;
end
```

问题三的代码

```
%% 获得参数
% n = 14;
% T=readtable('data\B题附件.xlsx',"VariableNamingRule","preserve");
% Need_Load=xlsread('data\B题附件.xlsx','各个节点物资的日需求量');
% F=xlsread('data\B题附件.xlsx','道路连接关系');
% Link=F(2:end,2:end);
% for i=1:size(Link,1)
% for j=1:size(Link,1)
%     if isnan(Link(i,j))
%         Link(i,j)=0;
%     end
% end
% end
% Link([5 1],2:end)=Link([1 5],2:end);
% Link(2:end,[5 1])=Link(2:end,[1 5]);
% temp=Link(5,1);
% Link(5,5)=0;
% Link(1,1)=0;
% Link(5,1)=temp;
% Link(1,5)=temp;
% t=Need_Load(2,1);
```

```
% Need_Load(2,1)=Need_Load(2,5);
% Need_Load(2,5)=t;
% Hos_Loc= [T{:,2},T{:,3}];
% Hos_Loc([5  1],:) = Hos_Loc([1  5],:);
% [Hos_dist,path]=cal_dist(Hos_Loc,Link);
% nVehicle = 2;
% min_tour = 1;
% pop_size = 160;
% num_iter = 5e3;
% a = meshgrid(1:n);
% [opt_route,opt_break,min_dist] = mtspf_ga_3(Hos_Loc,Hos_dist,nVehicle,min_tour,Nee
%%
function varargout = mtspf_ga_3(Hos_Loc,Hos_dist,nVehicle,min_tour,Need_Load,path,Li
%参数扫描
[N,~] = size(Hos_Loc);
[nr,nc] = size(Hos_dist);
if N ~= nr || N ~= nc
    error('ERROR!');
end
n = N - 1;
% 数据检查
nVehicle = max(1,min(n,round(real(nVehicle(1)))));
min_tour = max(1,min(floor(n/nVehicle),round(real(min_tour(1)))));
pop_size = max(8,8*ceil(pop_size(1)/8));
num_iter = max(1,round(real(num_iter(1))));
show_prog = logical(show_prog(1));
show_res = logical(show_res(1));

% 初始化路径分隔点
num_breaks = nVehicle-1;
dof = n - min_tour*nVehicle;  % 自由医院数目
addto = ones(1,dof+1);
for k = 2:num_breaks
    addto = cumsum(addto);%累加
end
cum_prob = cumsum(addto)/sum(addto);

%初始化种群
pop_route = zeros(pop_size,n);           % 路径
```

```
pop_break = zeros(pop_size,num_breaks);    %分割点
for k = 1:pop_size
    pop_route(k,:) = randperm(n)+1;
    pop_break(k,:) = randbreaks();
end
% 选择两条路径的颜色
clr = [1 0 0; 0 0 1];

% 运行GA遗传算法
global_min = Inf;
totalLoad=zeros(2,pop_size);
total_dist = zeros(1,pop_size);
dist_history = zeros(1,num_iter);
tmp_pop_route = zeros(8,n);
tmp_pop_break = zeros(8,num_breaks);
new_pop_route = zeros(pop_size,n);
new_pop_break = zeros(pop_size,num_breaks);
if show_prog
    pfig = figure('Name','MTSPF_GA | Current Best Solution','Numbertitle','off');
end
for iter = 1:num_iter
    % 计算货车的总行驶距离
    for p = 1:pop_size
        d = 0;
        p_route = pop_route(p,:);
        p_break = pop_break(p,:);
        rng = [[1 p_break+1];[p_break n]]';%代表每个旅行商的起点和终点
        for s = 1:nVehicle%计算这一种群的总距离
            d = d + Hos_dist(1,p_route(rng(s,1)));
            load=0; %记录货车补给消耗量
            for k = rng(s,1):rng(s,2)-1%计算旅行商中间的距离
                d = d + Hos_dist(p_route(k),p_route(k+1));
                load=load+Need_Load(2,p_route(k));
            end
            load=load+Need_Load(2,p_route(rng(s,2)));% 添加最后一次补给量
            d = d + Hos_dist(p_route(rng(s,2)),1); %添加返回的最后一段距离
            totalLoad(s,p)=load;
        end
        total_dist(p) = d;%货车的路径总长度
```

```
␣␣␣␣end
␣␣␣␣%␣在种群中找到最优路径，若存在则画出
␣[min_dist,index]␣=␣find_min_suit(total_dist,totalLoad);%调用函数
␣␣␣␣dist_history(iter)␣=␣min_dist;
␣␣␣␣if␣min_dist␣<␣global_min
␣␣␣␣␣␣␣␣global_min␣=␣min_dist;
␣␣␣␣␣␣␣␣opt_break␣=␣pop_break(index,:);
␣␣␣␣␣␣␣␣[opt_route,add_hos_1,add_hos_2]␣=␣find_route(pop_route,path,index,opt_break)
␣␣␣␣␣␣␣␣rng␣=␣[[1␣opt_break+1+add_hos_1];[opt_break+add_hos_1␣n+add_hos_1+add_hos_2]
        if show_prog
            %绘制最优路线
            figure(pfig);
            for s = 1:nVehicle
                route = [1 opt_route(rng(s,1):rng(s,2)) 1];
                plot(Hos_Loc(route,1),Hos_Loc(route,2),'.-','Color',clr(s,:));
                title(sprintf('总路径长度␣=␣%1.4f,␣迭代次数␣=␣%d',min_dist,iter));
                hold on
            end
            plot(Hos_Loc(1,1),Hos_Loc(1,2),'k*');
            hold off
        end
    end
    % 采用锦标赛方法进行选择，随机选择8个个体选择出最优个体作为父代
    rand_grouping = randperm(pop_size);
    for p = 8:8:pop_size
        routes = pop_route(rand_grouping(p-7:p),:);
        breaks = pop_break(rand_grouping(p-7:p),:);
        distances = total_dist(rand_grouping(p-7:p));
        [~,idx] = min(distances);
        best_of_8_route = routes(idx,:);
        best_of_8_break = breaks(idx,:);
        route_ins_pts = sort(ceil(n*rand(1,2)));
        I = route_ins_pts(1);
        J = route_ins_pts(2);
        for k = 1:8
            tmp_pop_route(k,:) = best_of_8_route;
            tmp_pop_break(k,:) = best_of_8_break;
            switch k
                case 2 % 翻转
```

```
                    tmp_pop_route(k,I:J) = fliplr(tmp_pop_route(k,I:J));
                case 3 % 交换
                    tmp_pop_route(k,[I J]) = tmp_pop_route(k,[J I]);
                case 4 % 迁移
                    tmp_pop_route(k,I:J) = tmp_pop_route(k,[I+1:J I]);
                case 5 % 修改间隔点
                    tmp_pop_break(k,:) = randbreaks();
                case 6
                    tmp_pop_route(k,I:J) = fliplr(tmp_pop_route(k,I:J));
                    tmp_pop_break(k,:) = randbreaks();
                case 7
                    tmp_pop_route(k,[I J]) = tmp_pop_route(k,[J I]);
                    tmp_pop_break(k,:) = randbreaks();
                case 8
                    tmp_pop_route(k,I:J) = tmp_pop_route(k,[I+1:J I]);
                    tmp_pop_break(k,:) = randbreaks();
                otherwise
            end
        end
        new_pop_route(p-7:p,:) = tmp_pop_route;
        new_pop_break(p-7:p,:) = tmp_pop_break;
    end
    pop_route = new_pop_route;
    pop_break = new_pop_break;
end
if show_res
    % 绘制多图像
    figure('Name','MTSPF_GA | Results','Numbertitle','off');
    subplot(2,2,1);
    plot(Hos_Loc(:,1),Hos_Loc(:,2),'r.');
for i=1:size(Hos_Loc,1)-1
for j=i+1:size(Hos_Loc,1)
    if Link(i,j)==1
    line([Hos_Loc(i,1),Hos_Loc(j,1)],[Hos_Loc(i,2),Hos_Loc(j,2)]);
    end
end
end
    title('医院位置及交通路线');
    text(Hos_Loc(1,1),Hos_Loc(1,2),'仓库','FontSize',7);
```

```
    text(Hos_Loc(2,1),Hos_Loc(2,2),'2','FontSize',7);
    text(Hos_Loc(3,1),Hos_Loc(3,2),'3','FontSize',7);
    text(Hos_Loc(4,1),Hos_Loc(4,2),'4','FontSize',7);
    text(Hos_Loc(5,1),Hos_Loc(5,2),'1','FontSize',7);
    text(Hos_Loc(6,1),Hos_Loc(6,2),'6','FontSize',7);
    text(Hos_Loc(7,1),Hos_Loc(7,2),'7','FontSize',7);
    text(Hos_Loc(8,1),Hos_Loc(8,2),'8','FontSize',7);
    text(Hos_Loc(9,1),Hos_Loc(9,2),'9','FontSize',7);
    text(Hos_Loc(10,1),Hos_Loc(10,2),'10','FontSize',7);
    text(Hos_Loc(11,1),Hos_Loc(11,2),'11','FontSize',7);
    text(Hos_Loc(12,1),Hos_Loc(12,2),'12','FontSize',7);
    text(Hos_Loc(13,1),Hos_Loc(13,2),'13','FontSize',7);
    text(Hos_Loc(14,1),Hos_Loc(14,2),'14','FontSize',7);
    axis square
    subplot(2,2,2);
    imagesc(Hos_dist([1 opt_route],[1 opt_route]));
    title('距离矩阵');
    axis square
    subplot(2,2,3);
    rng = [[1 opt_break+1];[opt_break n]]';
    for s = 1:nVehicle
        route = [1 opt_route(rng(s,1):rng(s,2)) 1]
        plot(Hos_Loc(route,1),Hos_Loc(route,2),'.-','Color',clr(s,:));
        legend('货车一','货车二','FontSize',4);
        title(sprintf('总路径长度 = %1.4f',global_min));
        hold on;
    end
    axis square
    plot(Hos_Loc(1,1),Hos_Loc(1,2),'ko');
    subplot(2,2,4);
plot(dist_history,'b','LineWidth',2);
    title('Best Solution History');
    set(gca,'XLim',[0 num_iter+1],'YLim',[0 1.1*max([1 dist_history])]);
    axis square
end
% Return Outputs
if nargout
    varargout{1} = opt_route;
    varargout{2} = opt_break;
```

```
    varargout{3} = min_dist;
end
% 若对路程距离有要求，则在染色体前面位置选择分隔点的概率会偏低一点
% 没有要求则随机生成间隔点
    function breaks = randbreaks()
        if min_tour == 1
            tmp_breaks = randperm(n-1);
            breaks = sort(tmp_breaks(1:num_breaks));
        else
            num_adjust = find(rand < cum_prob,1)-1;
            spaces = ceil(num_breaks*rand(1,num_adjust));
            adjust = zeros(1,num_breaks);
            for kk = 1:num_breaks
                adjust(kk) = sum(spaces == kk);
            end
            breaks = min_tour*(1:num_breaks) + cumsum(adjust);
        end
    end
end


function [Hos_dist,path]=cal_dist(Hos_Loc,Link)
Hos_dist=zeros(size(Hos_Loc,1));
path=zeros(size(Hos_Loc,1));
for i=1:size(Hos_Loc,1)-1
    for j=i+1:size(Hos_Loc,1)
    if Link(i,j)==1
        Hos_dist(i,j)=sqrt((Hos_Loc(i,1)-Hos_Loc(j,1))^2+(Hos_Loc(i,2)-Hos_Loc(j,2))
        Hos_dist(j,i)=Hos_dist(i,j);
    else
        min=1000;
        for k=1:size(Hos_Loc,1)
            if k~=i&&k~=j
                if Link(i,k)==1&&Link(j,k)==1&&min>sqrt((Hos_Loc(i,1)-Hos_Loc(k,1))^
                    sqrt((Hos_Loc(j,1)-Hos_Loc(k,1))^2+(Hos_Loc(j,2)-Hos_Loc(k,2))^2
                    min=sqrt((Hos_Loc(i,1)-Hos_Loc(k,1))^2+(Hos_Loc(i,2)-Hos_Loc(k,2
                    sqrt((Hos_Loc(j,1)-Hos_Loc(k,1))^2+(Hos_Loc(j,2)-Hos_Loc(k,2))^2
                    path(i,j)=k;
                    path(j,i)=k;
```

```
                    end
                end
            end
            Hos_dist(i,j)=min;
            Hos_dist(j,i)=Hos_dist(i,j);
        end
    end
end

function [min_dist,index]=find_min_suit(total_dist,totalLoad)
[x,y]=min(total_dist);
while totalLoad(1,y)<192||totalLoad(1,y)>500
    total_dist(y)=1000;
    [x,y]=min(total_dist);
end
    min_dist=x;
    index=y;
end

function [opt_route,add_hos_1,add_hos_2]=find_route(pop_route,path,index,opt_break)
add_hos_1=0;
add_hos_2=0;
opt_route=pop_route(index,1);
for i=1:12
if path(pop_route(index,i),pop_route(index,i+1))~=0&&i~=opt_break
    opt_route=[opt_route(1,:) path(pop_route(index,i),pop_route(index,i+1)) pop_rout
    if i<opt_break
    add_hos_1=add_hos_1+1;
    end
    if i>opt_break
    add_hos_2=add_hos_2+1;
    end
else
    opt_route=[opt_route(1,:) pop_route(index,i+1)];
end
end
end
```

问题四的代码

```
%% 获得参数
% n = 14;
% T=readtable('data\B题附件.xlsx',"VariableNamingRule","preserve");
% Need_Load=xlsread('data\B题附件.xlsx','各个节点物资的日需求量');
% F=xlsread('data\B题附件.xlsx','道路连接关系');
% Link=F(2:end,2:end);
% for i=1:size(Link,1)
% for j=1:size(Link,1)
% if isnan(Link(i,j))
% Link(i,j)=0;
% end
% end
% end
% Link([5 1],2:end)=Link([1 5],2:end);
% Link(2:end,[5 1])=Link(2:end,[1 5]);
% temp=Link(5,1);
% Link(5,5)=0;
% Link(1,1)=0;
% Link(5,1)=temp;
% Link(1,5)=temp;
% t=Need_Load(2,1);
% Need_Load(2,1)=Need_Load(2,5);
% Need_Load(2,5)=t;
% Hos_Loc= [T{:,2},T{:,3}];
% Hos_Loc([5 1],:) = Hos_Loc([1 5],:);
% [Hos_dist,path1,path2]=caldist(Hos_Loc,Link);
% second_Hos = i;%本处为第二个作为仓库的节点，请自行修改，注意为第一节点则输入5
% nVehicle = 2;
% min_tour = 1;
% pop_size = 160;
% num_iter = 5e3;
% a = meshgrid(1:n);
% [opt_route,opt_break,min_dist] = mtspf_ga_4(Hos_Loc,Hos_dist,nVehicle,min_tour,Nee
%%
function varargout = mtspf_ga_4(Hos_Loc,Hos_dist,nVehicle,min_tour,Need_Load,path1,p
%参数扫描
[N,~] = size(Hos_Loc);
[nr,nc] = size(Hos_dist);
```

```
if N ~= nr || N ~= nc
error('ERROR!');
end
n = N -1;
% 数据检查
nVehicle = max(1,min(n,round(real(nVehicle(1)))));
min_tour = max(1,min(floor(n/nVehicle),round(real(min_tour(1)))));
pop_size = max(8,8*ceil(pop_size(1)/8));
num_iter = max(1,round(real(num_iter(1))));
show_prog = logical(show_prog(1));
show_res = logical(show_res(1));
% 初始化路径分隔点
num_breaks = nVehicle-1;
dof = n - min_tour*nVehicle; % 自由医院数目
addto = ones(1,dof+1);
for k = 2:num_breaks
addto = cumsum(addto);%累加
end
cum_prob = cumsum(addto)/sum(addto);
%初始化种群
pop_route = zeros(pop_size,n); % 路径
pop_break = zeros(pop_size,num_breaks); %分割点
for k = 1:pop_size
pop_route(k,:) = randperm(n)+1;
pop_break(k,:) = randbreaks();
end
% 选择两条路径的颜色
clr = [1 0 0; 0 0 1];
% 运行GA遗传算法
global_min = Inf;
totalLoad=zeros(2,pop_size);
total_dist = zeros(1,pop_size);
dist_history = zeros(1,num_iter);
tmp_pop_route = zeros(8,n);
tmp_pop_break = zeros(8,num_breaks);
new_pop_route = zeros(pop_size,n);
new_pop_break = zeros(pop_size,num_breaks);
if show_prog
pfig = figure('Name','MTSPF_GA␣|␣Current␣Best␣Solution','Numbertitle','off');
```

```
end

for iter = 1:num_iter
% 计算每一个货车的总行驶距离
for p = 1:pop_size
d=0;
p_route = pop_route(p,:);
p_break = pop_break(p,:);
rng = [[1 p_break+1];[p_break n]]';%代表每个旅行商的起点和终点
for s = 1:nVehicle%计算这一种群的总距离
load=0; %记录货车补给消耗量
for k = rng(s,1):rng(s,2)-1%计算旅行商中间的距离
d = d + Hos_dist(p_route(k),p_route(k+1));
load=load+Need_Load(2,p_route(k+1));
end
if s==1
d = d + Hos_dist(p_route(rng(s,2)),1); %添加返回的最后一段距离
end
if s==nVehicle
d = d + Hos_dist(p_route(rng(s,2)),p_route(1)); %添加返回的最后一段距离
end
totalLoad(s,p)=load;
end
total_dist(p) = d;%每个货车的路径总长度
end
% 在种群中找到最优路径，若存在则画出
[min_dist,index] = find_min_suit(total_dist,totalLoad,Need_Load,second_Hos,pop_break
if min_dist~=0
dist_history(iter) = min_dist;
if min_dist < global_min
global_min = min_dist;
opt_break = pop_break(index,:);
[opt_route,add_hos_1,add_hos_2] = find_route(pop_route,path1,path2,index,opt_break);
rng = [[1 opt_break+1+add_hos_1];[opt_break+add_hos_1 n+add_hos_1+add_hos_2]];
if show_prog
%绘制最优路线
figure(pfig);
for s = 1:nVehicle
    if s==1
```

```
        route = [1 opt_route(rng(s,1):rng(s,2)) 1]
    end
    if s==nVehicle
        route = [opt_route(rng(s,1):rng(s,2)) opt_route(rng(s,1))]
    end
plot(Hos_Loc(route,1),Hos_Loc(route,2),'.-','Color',clr(s,:));
title(sprintf('总路径长度=%1.4f,迭代次数=%d',min_dist,iter));
hold on
end
hold off
end
end
end
% 采用锦标赛方法进行选择，随机选择8个个体选择出最优个体作为父代
rand_grouping = randperm(pop_size);
for p = 8:8:pop_size
routes = pop_route(rand_grouping(p-7:p),:);
breaks = pop_break(rand_grouping(p-7:p),:);
distances = total_dist(rand_grouping(p-7:p));
[~,idx] = min(distances);
best_of_8_route = routes(idx,:);
best_of_8_break = breaks(idx,:);
route_ins_pts = sort(ceil(n*rand(1,2)));
I = route_ins_pts(1);
J = route_ins_pts(2);
for k = 1:8
tmp_pop_route(k,:) = best_of_8_route;
tmp_pop_break(k,:) = best_of_8_break;
switch k
case 2 % 翻转
tmp_pop_route(k,I:J) = fliplr(tmp_pop_route(k,I:J));
case 3 % 交换
tmp_pop_route(k,[I J]) = tmp_pop_route(k,[J I]);
case 4 % 迁移
tmp_pop_route(k,I:J) = tmp_pop_route(k,[I+1:J I]);
case 5 % 修改间隔点
tmp_pop_break(k,:) = randbreaks();
case 6
tmp_pop_route(k,I:J) = fliplr(tmp_pop_route(k,I:J));
```

```
tmp_pop_break(k,:) = randbreaks();
case 7
tmp_pop_route(k,[I J]) = tmp_pop_route(k,[J I]);
tmp_pop_break(k,:) = randbreaks();
case 8
tmp_pop_route(k,I:J) = tmp_pop_route(k,[I+1:J I]);
tmp_pop_break(k,:) = randbreaks();
otherwise
end
end
new_pop_route(p-7:p,:) = tmp_pop_route;
new_pop_break(p-7:p,:) = tmp_pop_break;
end
pop_route = new_pop_route;
pop_break = new_pop_break;
end
if show_res
% 绘制多图像
figure('Name','MTSPF_GA␣|␣Results','Numbertitle','off');
subplot(2,2,1);
plot(Hos_Loc(:,1),Hos_Loc(:,2),'r.');
for i=1:size(Hos_Loc,1)-1
for j=i+1:size(Hos_Loc,1)
if Link(i,j)==1
line([Hos_Loc(i,1),Hos_Loc(j,1)],[Hos_Loc(i,2),Hos_Loc(j,2)]);
end
end
end
title('␣医院位置及交通路线');
text(Hos_Loc(1,1),Hos_Loc(1,2),'5','FontSize',7);
text(Hos_Loc(2,1),Hos_Loc(2,2),'2','FontSize',7);
text(Hos_Loc(3,1),Hos_Loc(3,2),'3','FontSize',7);
text(Hos_Loc(4,1),Hos_Loc(4,2),'4','FontSize',7);
text(Hos_Loc(5,1),Hos_Loc(5,2),'1','FontSize',7);
text(Hos_Loc(6,1),Hos_Loc(6,2),'6','FontSize',7);
text(Hos_Loc(7,1),Hos_Loc(7,2),'7','FontSize',7);
text(Hos_Loc(8,1),Hos_Loc(8,2),'8','FontSize',7);
text(Hos_Loc(9,1),Hos_Loc(9,2),'9','FontSize',7);
text(Hos_Loc(10,1),Hos_Loc(10,2),'10','FontSize',7);
```

```
text(Hos_Loc(11,1),Hos_Loc(11,2),'11','FontSize',7);
text(Hos_Loc(12,1),Hos_Loc(12,2),'12','FontSize',7);
text(Hos_Loc(13,1),Hos_Loc(13,2),'13','FontSize',7);
text(Hos_Loc(14,1),Hos_Loc(14,2),'14','FontSize',7);
axis square
subplot(2,2,2);
imagesc(Hos_dist([1 opt_route],[1 opt_route]));
title('距离矩阵');
axis square
subplot(2,2,3);
rng = [[1 opt_break+1];[opt_break n]]';
for s = 1:nVehicle
    if s==1
        route = [1 opt_route(rng(s,1):rng(s,2)) 1];
    end
    if s==nVehicle
        route = [opt_route(rng(s,1):rng(s,2)) opt_route(rng(s,1))];
    end
plot(Hos_Loc(route,1),Hos_Loc(route,2),'.-','Color',clr(s,:));
legend('货车一','货车二','FontSize',4);
title(sprintf('总路径长度 = %1.4f',global_min));
hold on;
end
axis square
subplot(2,2,4);
plot(dist_history,'b','LineWidth',2);
title('Best Solution History');
set(gca,'XLim',[0 num_iter+1],'YLim',[0 1.1*max([1 dist_history])]);
axis square
end
% Return Outputs
if nargout
varargout{1} = opt_route;
varargout{2} = opt_break;
varargout{3} = min_dist;
end
%若对路程距离有要求，则在染色体前面位置选择分隔点的概率会偏低一点
%没有要求则随机生成间隔点
function breaks = randbreaks()
```

```
if min_tour == 1
tmp_breaks = randperm(n-1);
breaks = sort(tmp_breaks(1:num_breaks));
else
num_adjust = find(rand < cum_prob,1)-1;
spaces = ceil(num_breaks*rand(1,num_adjust));
adjust = zeros(1,num_breaks);
for kk = 1:num_breaks
adjust(kk) = sum(spaces == kk);
end
breaks = min_tour*(1:num_breaks) + cumsum(adjust);
end
end
end

function [min_dist,index]=find_min_suit(total_dist,totalLoad,Need_Load,second_Hos,po
Loc=0;
dist=10000;
min_dist=0;
index=0;
for i=1:size(pop_route(:,1))
    if pop_route(i,pop_break(i,:)+1)==second_Hos
        Loc=[Loc i];
        dist=[dist total_dist(i)];
    end
end
[x,y]=min(dist);
k=Loc(1,y);
if k~=0
while totalLoad(1,k)<192-Need_Load(2,pop_route(k,1))||totalLoad(1,k)>500
    dist(1,y)=10000;
    [x,y]=min(dist);
    k=Loc(1,y);
    if k==0
        break;
    end
end
    min_dist=x;
    index=k;
```

```
end
end


function [Hos_dist,path1,path2]=cal_dist(Hos_Loc,Link)%计算允许存在两个过渡医院的距离
Hos_dist=zeros(size(Hos_Loc,1));
path1=zeros(size(Hos_Loc,1));
path2=zeros(size(Hos_Loc,1),size(Hos_Loc,1),2);
for i=1:size(Hos_Loc,1)-1
    for j=i+1:size(Hos_Loc,1)
    if Link(i,j)==1
        Hos_dist(i,j)=sqrt((Hos_Loc(i,1)-Hos_Loc(j,1))^2+(Hos_Loc(i,2)-Hos_Loc(j,2))
        Hos_dist(j,i)=Hos_dist(i,j);
    else
         min=1000;
        for k=1:size(Hos_Loc,1)
            if k~=i&&k~=j
                if Link(i,k)==1&&Link(j,k)==1&&min>sqrt((Hos_Loc(i,1)-Hos_Loc(k,1))^
                    sqrt((Hos_Loc(j,1)-Hos_Loc(k,1))^2+(Hos_Loc(j,2)-Hos_Loc(k,2))^2
                    min=sqrt((Hos_Loc(i,1)-Hos_Loc(k,1))^2+(Hos_Loc(i,2)-Hos_Loc(k,2
                    sqrt((Hos_Loc(j,1)-Hos_Loc(k,1))^2+(Hos_Loc(j,2)-Hos_Loc(k,2))^2
                     path1(i,j)=k;
                     path1(j,i)=k;
                else
                    for w=1:size(Hos_Loc,1)
                        if w~=k&&w~=i&&w~=j
                            if Link(i,k)==1&&Link(k,w)==1&&Link(w,j)==1&&min>sqrt((H
                    sqrt((Hos_Loc(k,1)-Hos_Loc(w,1))^2+(Hos_Loc(k,2)-Hos_Loc(w,2))^2
                                min=sqrt((Hos_Loc(i,1)-Hos_Loc(k,1))^2+(Hos_Loc(i,2)
                    sqrt((Hos_Loc(k,1)-Hos_Loc(w,1))^2+(Hos_Loc(k,2)-Hos_Loc(w,2))^2
                                path2(i,j,1)=k;
                                path2(i,j,2)=w;
                                path2(j,i,1)=w;
                                path2(j,i,2)=k;
                            end
                        end
                    end
                end
            end
```

```
        end
        Hos_dist(i,j)=min;
        Hos_dist(j,i)=Hos_dist(i,j);
    end
    end
end


function [opt_route,add_hos_1,add_hos_2]=find_route(pop_route,path1,path2,index,opt_
add_hos_1=0;
add_hos_2=0;
opt_route=pop_route(index,1);
for i=1:12
if path1(pop_route(index,i),pop_route(index,i+1))~=0&&i~=opt_break
    opt_route=[opt_route(1,:) path1(pop_route(index,i),pop_route(index,i+1)) pop_rou
    if i<opt_break
    add_hos_1=add_hos_1+1;
    end
    if i>opt_break
    add_hos_2=add_hos_2+1;
    end
else
    if path2(pop_route(index,i),pop_route(index,i+1),1)~=0&&i~=opt_break&&i~=opt_bre
    opt_route=[opt_route(1,:) path2(pop_route(index,i),pop_route(index,i+1),1) path2
    if i<opt_break
    add_hos_1=add_hos_1+2;
    end
    if i>opt_break
    add_hos_2=add_hos_2+2;
    end
    else
    opt_route=[opt_route(1,:) pop_route(index,i+1)];
    end
end
end
end
```