基于改进式遗传算法的多目标路径优化方案的分析

摘　要

本文对俄罗斯与乌克兰战争中拟定的军事运输的最优路径进行了分析，建立了蚁群模型、运输接收量数学模型和时间数学模型，对最优运输路径进行制定，最终应用在实际问题中。

针对问题一,去建立物资补给站,要求保证在规定的运输时间条件之下和物资需求点之内去完成任务,首先要建立出每个物资需求点之间的距离坐标,数据由表一可以得到,忽略任何干扰的条件,两点之间直线距离是最小,可以达到最大的程度去节约物资运输的时间,最新供给点的建立,可以被理解成为如何求得费兹马点坐标 P 的一个问题,在此理论基础上,可以直接用这两个模型函数来做互相的验证,使用了 Matlab 软件自 **fminsearch 函数**,在该 fminsearch 函数过程中,用到了**隐匿函数**,先通过将需求点变换为坐标,得到了初始化的数据,所需要的是输入的参数有两个,分别记为目标函数值和目标变量初值,变量的初值其实并不是单是指需求点坐标上的一个横坐标,也有可以认为是一个二维的甚至可能是一个三维的,在此题中可看做先是求一个二维变量,再去求得一个满足最小距离要求的需求点,其次才是 Matlab **粒子群算法**搜索**费马点**,用到了**迭代优化思想**,将所有需求点都看做一个粒子,初始化所有粒子的速度,计算适应度,将当前适应度为最佳群体的粒子作为当前最佳的个体,再重新将当前的这最后一批的粒子作为当前最佳的群体,更新各个粒子的速度以及最新的位置,重新的计算出每个粒子当前的最佳适应度,与最佳个体进行比较更换,直到满足优化目标或迭代次数达到上限得到最优费马点位置,最后若是在满足题目的条件下,两种方法得到同一个费马点 P,那么这个点即是新补给点的修建地点。

针对问题二，已知新的补给站建立在坐标（10,30）上，基于原始条件的限制下，通过对飞行方案的制定从而得到最优的运输方案，由此分析，首先，为了解决此类多条件多变量问题，要明确限制条件：一为飞机出发到某一个运输地点并成功运输完成之后，必须返回补给点，才能再次出发，二为 A、B 型飞机的数量、速度、载重量，三为部队接收物资的数量，四为时间。其次再建立**运输接收量数学模型**和**时间数学模型**确定飞机的调用方案，基于条件限制，进行**优化问题分析**，使飞机实现在最短时间、最少来回次数、全部完成任务的最优解时的方案是最好的调用方案，基于目标函数，引入**遗传算法**，进行**最优**来回次数的判断，选择最优的的来回次数，对于 A 和 B 型飞机进行分别进行分类讨论，最后基于以上条件进行组合，形成最终的调用方案。

针对问题三，基于问题二的基础之上，在各个部队接到上级要求需要再收到补给后继续移动前往下一个地点执行任务，但是题干中并未表述转移的速度条件，所以**首先**要对所有可能的情况进行分析根据题目所给出的关系和第二问得到的结论，将所有肯分为两类，其中第一类为不同部队之间无影响关系，即运输完第一个点后，此点的部队进行的转移，对于之后要去的部队无任何影响，两者无任何关联，而第二类就是运输到第二个点后，第一支部队同时来到了第二个点位，相互之间可以有运输物质的关联，但是以上两种情况都要建立在既定路线下，对于未知路线是无法实现数据分析，其次建立了**蚁群模型**区队，对所有路径情况进行实例化，确定一个最优路径方案的建立，形成一个最短距离的循环，基于距离，最后基于第二问所建立的数学模型，加上新的数量和损耗限制，形成新的数学模型，再对于**最优方案**进行分析，从而形成最优转运方案。

关 键 词：粒子群算法 迭代思想 fminsearch 函数 蚁群模型 遗传算法

一、问题重述

1.1 问题背景

近日俄罗斯和乌克兰的冲突日益加剧，不难发现现代战争仍然是以人为主的战争，并未进入到全机械化的战争状态。那么说明在军事战争环境中军给事物资中的后勤补给便显得更加十分之重要,如中国的古代军事兵法书中有讲到的"兵马未动,粮草先行"就是对这一点重要性的体现。现代战争中，士兵往往不会一次性带上大量的食物和弹药，如果陷入了较为持久的战斗状态则需要调动补给站中的后勤部队快速补充物资。而分析其中的问题对战场资源进行合理分配会大大加快支援速度。

1.2 问题的数据条件

在某地的补给站附近，同时有 4 支部队需要补充物资，而补给站有 A,B 两种类型的飞机，A 类飞机有 6 架,最大载重 13 吨，飞行速度 260km/h；B 类飞机有 10 架,最大载重 20 吨,飞行速度 50km/h。每支部队可以接受多于所需要的物资，但接受的物资数量不能超过所需数量的 150%。飞机装卸也需要时间，A 类飞机装卸均为 20min；B 类飞机装载货物需要 30min，卸载货物需要 40 分钟。 4 支部队的坐标、请求物资的数量最晚的需求时间见表 1。

1.3 问题的提出

（1）假设旧补给站被敌军炸毁，需要紧急建设新补给站。不考虑其他因素，仅仅考虑能最大程度给各个部队提供支援，请建立合适的数学模型确定新补给站的修建地点。
（2）由于种种原因，新补给站被修建在坐标为(10,30)的位置，请建立数学模型确定飞机的调用方案。
（3）各个部队接到上级要求需要再收到补给后继续移动前往下一个地点执行任务，因此不能携带超过所需数量 110%的物资，且由于搬运过程中 A 类飞机会有 5%-10%的损耗，B 类飞机有 3%-7%的损耗。请建立数学模型确定转运方案。

二、问题分析

2.1 问题一的分析

在原有的补给站被被摧毁的情况下，需要重新建立一个补给站，要求在能够在规定的时间条件和物资需求之内去完成任务区，首先建立每个需求点的坐标，数据由表一得到，忽略干扰条件，将问题一简化为用最短的时间运输每个需求点所需要的物资，换而言之，需要得到最短的时间，就需要得到新补给点到四个需求点的距离之和最小，可以最大程度节约运输时间，想要在仅仅考虑最大程度给以支援的条件下建立新的供给点，可以理解为求得费马点 P 的问题，在平面内有 n（n>=3）个点 $N_1(x_1,y_1),N_2(x_2,y_2),\ldots,N_n(x_n,y_n)$，现求一点 P(x,y)，使得 P 到各点直线距离之和最

小，当 n=3 时，就是著名的三角形费马问题，题目要求仅仅考虑最大程度提供支援，可以将题简化成用最少的时间将所需物资送达，也可以看做费马问题，此题中设 n=4。在此基础上，可以用两个模型来互相验证，首先是 Matlab 自带的 fminsearch 函数，求得满足最小距离的点，其次是 Matlab 粒子群算法搜索费马点，用到了迭代思想，最后若是在满足题目的条件下，两种方法得到同一个费马点 P，那么这个点即是新补给点的修建地点。

2.2 问题二的分析

对于问题二，是基于已知坐标点（10,30）开始，对于其他点的四支部队进行物资的运输，从而基于一些限制条件进行一个最优飞机调度方案，由此分析可知为多条件变量的最优解问题，限制条件一飞机出发到某一个运输地点并成功运输完成之后，必须返回补给点，才能再次出发，限制条件二为 A、B 型飞机的数量、速度、载重量，限制条件三为部队接收物资的数量，限制条件四为时间。可以在条件限制的情况下，建立数学模型确定飞机的调用方案，基于条件限制，飞机实现在最短时间、最少来回次数、全部完成任务的最优解时的方案是最好的调用方案，因此可以基于已知条件进行函数关系式的构建，形成运输接收量数学模型和时间数学模型，基于目标函数，引入遗传算法，进行最优来回次数的判断，选择最优的的来回次数，对于 A 和 B 型飞机进行分别进行分类讨论，最后基于以上条件进行组合，形成最终的调用方案。

2.3 问题三的分析

各个部队接到上级要求，在收到物资后进行任务转移，但是题干中并未表述转移的速度条件，所以要对所有可能的情况进行分析，分为两类，其中第一类为不同部队之间无影响关系，即运输完第一个点后，此点的部队进行的转移，对于之后要去的部队无任何影响，两者无任何关联，第二类就是运输到第二个点后，第一支部队也来到了第二个点位，相互之间可以有运输物质的关联，但是以上两种情况都要建立在既定路线下，对于未知路线是无法实现数据分析的，所以，首先要利用蚁群模型，进行一个最优路径方案的建立，形成一个最短距离的循环，基于距离，基于第二问所建立的数学模型，加上新的数量和损耗限制，形成新的数学模型，再对于最优方案进行分析，从而形成最优转运方案。

三、模型假设

1. 假设题目所给的数据真实可靠；
2. 假设在运输过程中无意外因素干扰；
3. 假设基于条件所设假设真实可靠；
4. 假设基于多要素的最优模型不再受其他条件干扰。

四、定义与符号说明

符号	说明
C_A、C_B	飞机来回次数
L_{A_P}	A 型飞机基于 p 地的运输量
L_{B_P}	B 型飞机基于 p 地的运输量
L_j	J 地总货量
T_{A_j}	A 型飞机来回一次的时间
T_{B_j}	B 型飞机来回一次的时间
T'_{A_i}	全部 A 型飞机完成任务所需要的时间
T'_{B_i}	全部 B 型飞机完成任务所需要的时间
Q	百分比含量
distance	最短路径之和
x_n	横坐标
c_1，c_2	学习因子
y_n	纵坐标
evolution_max	最大进化次数
fitness_best	最优适应度

五、模型的建立与求解

5.1 问题一

5.1.1 问题分析

对于问题一,从上面给出的坐标表一开始建立坐标,忽略了运输物资过程中所有的干扰和条件,将问题表一过程简化为需要用一个最短的时间去运输到每个新需求点上所可能需要到的全部物资,换句而言讲之,需要运输得到物资最短的运输时间,就表示需要运输得到每个新物资补给点上到这四个新需求点间的运输距离之大和之最小,所以费马的问题二是很著名的一个几何极值问题,已知是一个三角形,求作一点,使到其顶点与这个三角

形内的另外三个相邻顶点间的两点距离的之和均为一个极小,费尔马点即为到这三个相邻点间距离的之和是最小的一个点,推而广则之,到这 n 个端点距离的之和就是最小的点也因此可以直接叫做费尔斯马点,此题求解在不考虑其他因素,仅仅考虑能最大程度给各个部队提供支援建立新的补给站,也可以看做求解费马点的数学问题。采用两种模型来寻找费马点,一个是 Matlab 自带的 fminsearch 函数,另一个是 Matlab 粒子群算法,如果两种模型寻找到的结果相同,可以认为这个费马点就是最优的新补给点的修建地点。

5.1.2 条件处理及模型建立

(1) Matlab 的 femisearch 函数

首先设置目标函数，并且列出坐标点 A120 坐标为（5，4），A122 坐标为（9，5），B121 坐标为（13，6），B404 坐标为（16，4），处理目标函数步骤如下：

步骤一.data-x

$$data-x=\begin{bmatrix}5 & 4\\ 9 & 5\\ 13 & 6\\ 16 & 4\end{bmatrix}-[m \quad n]=\begin{bmatrix}5-m & 4-n\\ 9-m & 5-n\\ 13-m & 6-n\\ 16-m & 4-n\end{bmatrix}$$

步骤二，（data-x）^2

$$(data-x).^2=\begin{bmatrix}5-m & 4-n\\ 9-m & 5-n\\ 13-m & 6-n\\ 16-m & 4-n\end{bmatrix}.*\begin{bmatrix}5-m & 4-n\\ 9-m & 5-n\\ 13-m & 6-n\\ 16-m & 4-n\end{bmatrix}=\begin{bmatrix}(5-m)^2 & (4-n)^2\\ (9-m)^2 & (5-n)^2\\ (13-m)^2 & (6-n)^2\\ (16-m)^2 & (4-n)^2\end{bmatrix}$$

步骤三.sum((data-x)^2,2)

$$sum((data-x).^2,2)=\begin{bmatrix}(5-m)^2+(4-n)^2\\ (9-m)^2+(5-n)^2\\ (13-m)^2+(6-n)^2\\ (16-m)^2+(4-n)^2\end{bmatrix}$$

步骤四.sqart(sum((data-x)^2,2))

$$sqart(sum((data-x).^2,2))=\begin{bmatrix}\sqrt{(5-m)^2+(4-n)^2}\\ \sqrt{(9-m)^2+(5-n)^2}\\ \sqrt{(13-m)^2+(6-n)^2}\\ \sqrt{(16-m)^2+(4-n)^2}\end{bmatrix}$$

步骤五.sum(sqart(sum((data-x)^2,2)))

$$(sum(sqart(sum((data-x).^2,2)))=\sqrt{(5-m)^2+(4-n)^2}+\sqrt{(9-m)^2+(5-n)^2}+\sqrt{(13-m)^2+(6-n)^2}+\sqrt{(16-m)^2+(4-n)^2}$$

在已搜索给出的全部 4 个需求点中各找到某一个的初值位置 P,若从这个点到其他这三个需求点间的距离之大和最小,最后搜索结果得到的费马点值 Q 点必然是落点在 P 点的附近,这样才可以最大提高搜索结果效率,避免了盲目的搜索。程序如下：

```
for i = 1:n
distance(i) = fun(data(i,:));
end
[~,index] = min(distance);
```

可以得到一个最优点初值为（9,5），发现与已知点重合，再运行全部的程序可以得到：

最短路径之和为 15.3173，横坐标为 9，纵坐标为 5，坐标图如图显示:

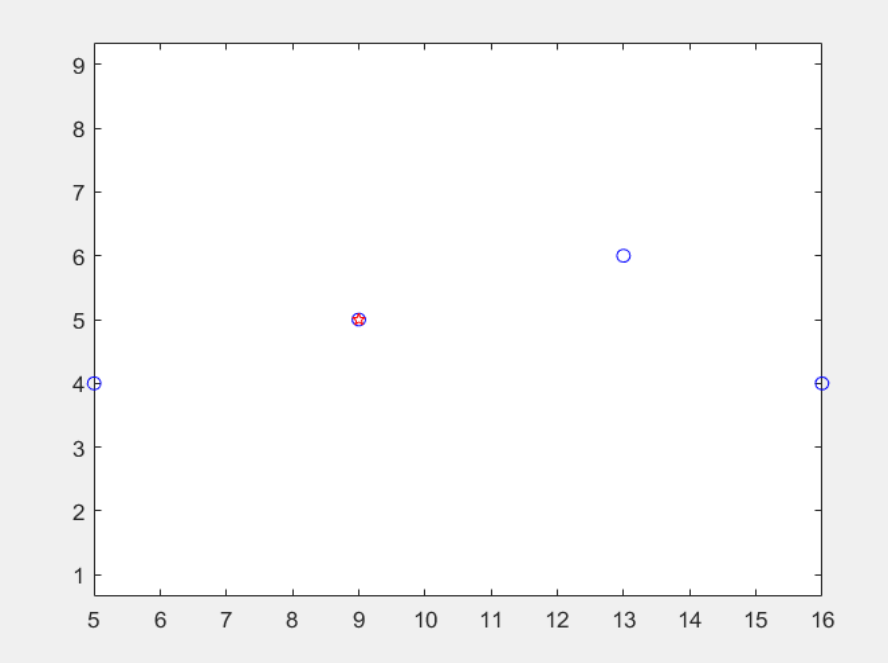

图 5-1-1 根据 feminsearch 函数得到的费马点

(2) Matlab 粒子群算法

首先粒子群算法中的适应度函数就是目标函数，即对于费马点 P（x，y），其到其与点$\{(x_1,y_1),(x_2,y_2)\dots(x_n,y_n)\}$的距离之和最小。

$$Minf(x,y)=\sum_{i=1}^{n}\sqrt{(x_i-x)^{2+}+(y_i-y)^2}$$

学习因子c_1和c_2，最大进化次数 evolution_max，$\{(x_1,y_1),(x_2,y_2)\dots(x_n,y_n)\}$中横坐标的最大最小值$x_{min}$，$y_{min}$，$x_{max}$，$y_{max}$，x 和 y 方向的最大速度和最小速度。计算了所有粒子的适应度，现在需要从中选出最优的适应度 fitness_best 和其索引 index_best。matlab 提供了 min 函数，可以搜索一个向量中的最小设置值及其位置。

接着把有最优适应度的粒子的位置保存到 I，把当前的这群粒子暂时作为最佳群体，保存到 P，最后分别保存 I 和 P 的适应度 I_fitness_best 和 P_fitness_best，用以之后的对比。绘制每一次迭代的最有适应度 fitness_evolution,输出最后求解得到的费马点 P 和最小距离和 P_fitness_best。得到迭代次数与最优适应度图像如下所示：

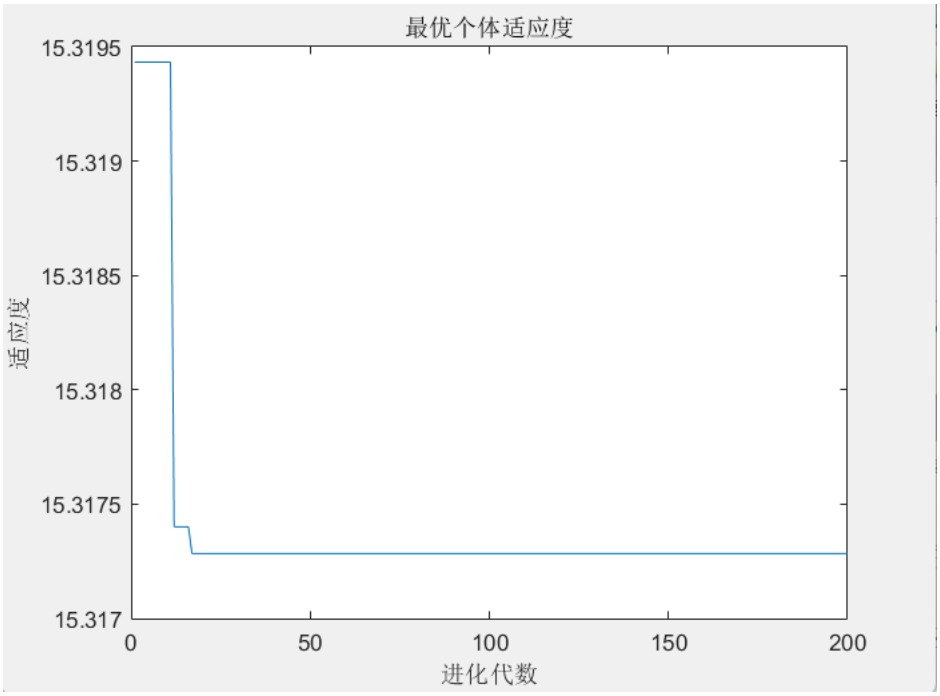

图 5-1-2 迭代次数与最优适应度关系

对于费马点的搜索，费马点的横坐标必定不会小于中的任何点。$\{(x_1,y_1),(x_2,y_2)\dots(x_n,y_n)\}$的横坐标，必定不会大于任何点的横坐标，既

$$\min（x_1，x_2,\dots,x_n）\leq x \leq \max(x_1，x_2,\dots,x_n)$$

纵坐标必定不会小于中的任何点$\{(x_1,y_1),(x_2,y_2)\dots(x_n,y_n)\}$的纵坐标，必定不会大于任何点的纵坐标，既

$$\min（y_1，y_2,\dots,y_n）\leq y \leq \max(y_1，y_2,\dots,y_n)$$

找到横纵坐标的极值，组合后得到四个点，并得到矩形，那么费马点必须在这个矩形内，根据这个逻辑对超出边界的粒子进行修正即可，得到的图像如下所示：

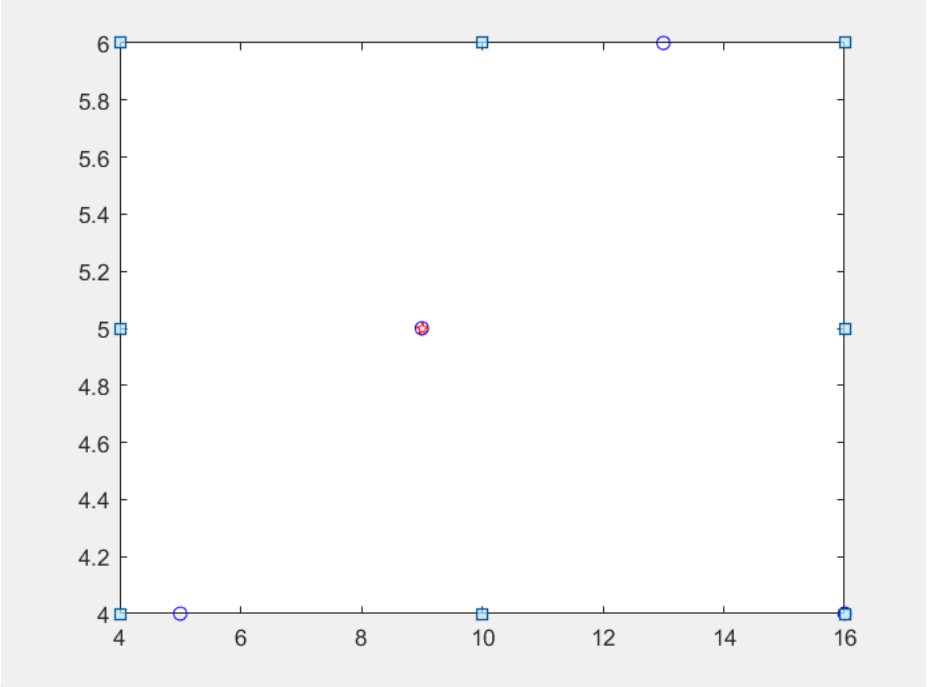

图 5-1-3 四个包含极值的坐标构成的矩形

代入完整代码，可以得到一个包含费马点的图像，如图所示：

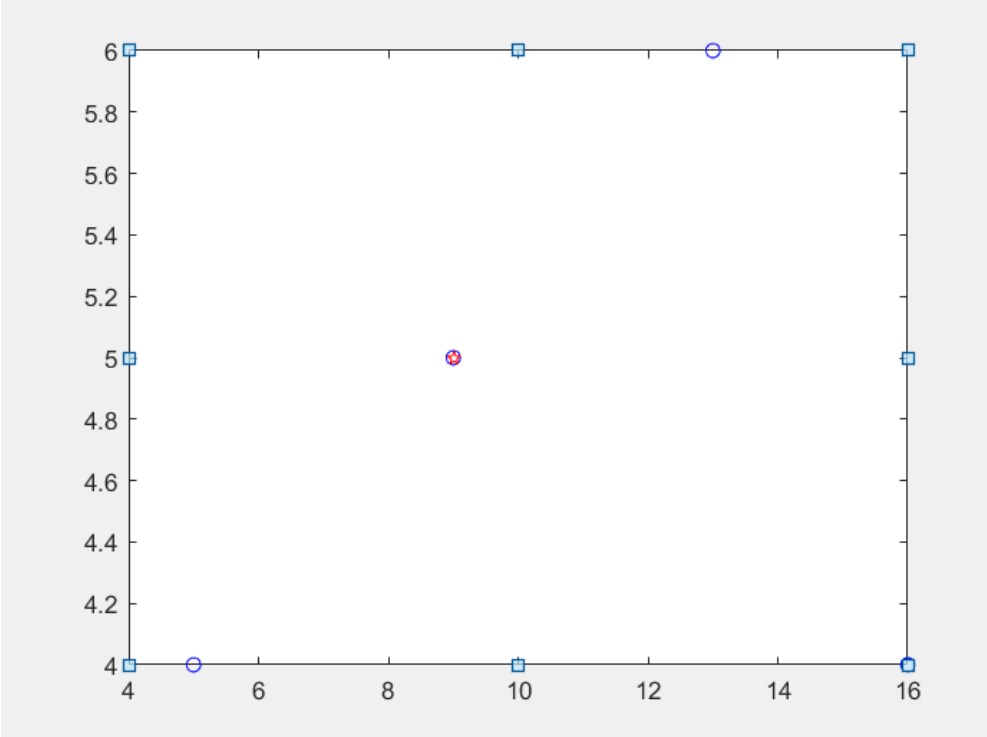

图 5-1-4 根据粒子群算法得到的费马点

可以发现由粒子群得到的费马点与 feminsearch 函数得到的费马点坐标一致。

5.1.3 问题解决

问题一可以简化为寻找最短时间，而最短时间同时对应了最短路径，而题目要求每架飞机的路线都需要在供给点和需求点来回，可以简化为需要寻找新的供给点到其他四个需求点的距离之和最小，根据两种数学模型可以找到新的供给点修建地点。

5.1.4 结果及分析

基于 fminsearch 函数模型和粒子群算法模型来对系统进行优化构建的求解数学模型,可以用来事先就确定计算出在需要时进行求解时所用的求解的目标函数,在这 fminsearch 函数建模系统中,将任何一个需求点坐标变换为坐标,得到的一组初始化的输入是数据,所需要初始化或者需要处理的一组初始化时输入到的参数就必须至少有这两个变量,一个函数值即为解的目标函数值而另一个目标数值则为所求该求解变量值的一个初值,变量中的那个初值实际上并不一定总是单纯是指这个解需求点的二维坐标的一个横坐标,也因此即也可以简单认为这是个二坐标维变量甚至更可能认为是指一个三维坐标的一个多维变量,在解答此题过程中可单纯看做这是解需求点坐标的其中一个二维横坐标,再可通过利用由 Matlab 程序所自带出来的函数值进行计算后即可以简单计算并得到解费马点坐标为(9,5),由随机优化算法实现的粒子群算法,将任何一个需求点坐标都看做是一群粒子,初始化时计算出所有的粒子群的粒子运动的速度,计算粒子适应度,将当适应度达到当前最佳状态时的某一群的粒子群作为一个当前状态最佳的个体,再重新将当前状态最佳时这一批群的各个粒子群作为当前最佳的群体,更新计算了当前各个群体粒子运动的最新速度以及最新粒子的位置,重新计算算出每个粒子之间的适应度,与最佳个体进行比较更换,直到满足优化目标或迭代次数达到上限得到最优费马点位置(9,5),两种模型得到的费马点位置相同,可以看做(9,5)为新的供给点的修建地点。fminsearch 函数可以得到最优费马点坐标为(9,5),粒子群算法这一随机优化算法进行求解同样得到最有费马点坐标为(9,5),根据经验可以确定需要寻找的新的供给点的修建地点为(9,5)。

5.2 问题二

5.2.1 问题分析

对于问题二，要从已知坐标点（10,30）出发，对于不同的部队进行物资运输，其中限制条件包括运输次数、运输数量、运输速度、飞机数量、飞机类型、时间、需求数量等限制条件，可以利用倒推法进行处理，基于所需数量和时间进行优化分析，在满足最终需求的同时实现其余消耗最小化，同时达到在最短时间内完成运输任务，但是还有一个条件就是要实现组合，对于 A 型和 B 型飞机进行最优的组合的同时，再在以上条件下进行调用，此时的方案便是最优的方案。

5.2.2 条件处理及模型建立

首先对于所有的飞机进行一个分类处理，并且对于位置点进行一个指代，令 A120

位置为位置 1，A122 位置为位置 2，B121 位置为位置 3，B404 位置 1 位置 4，基于飞行来回次数和 A 和 B 不同的机型进行一个每一架飞机的拆分，建立飞机来回次数矩阵如下：

$$C_{A_{ij}} = \begin{pmatrix} C_{A_{11}} & \cdots & C_{A_{14}} \\ \vdots & \ddots & \vdots \\ C_{A_{61}} & \cdots & C_{A_{64}} \end{pmatrix}$$

$$C_{B_{ij}} = \begin{pmatrix} C_{B_{11}} & \cdots & C_{B_{10\,4}} \\ \vdots & \ddots & \vdots \\ C_{B_{10\,1}} & \cdots & C_{B_{10\,4}} \end{pmatrix}$$

其中，i，j 分别代表第几架飞机和目的地位置。

紧接着，对 A 型和 B 型飞机进行一个运输数量的函数关系构建，形成 A、B 型飞机运输数量函数模型如下：

$$L_{A_P} = \sum_{i=1}^{i=6} C_{A_{iP}*13}$$

$$L_{B_P} = \sum_{i=1}^{i=10} C_{B_{iP}*20}$$

由 A、B 型运输数量便可得出某一部队位置所受到的数量函数模型如下:

$$L_j = \left(L_{A_j} + L_{B_j}\right)$$

$$1 \le \frac{L_j}{L_{jmin}} \le 1.5$$

因为要进行时间计算，所以要进行一个距离的计算，利用 Matlab 进行信息处理，得到如下距离信息表：

0	26.4764	25.01999	24.18677	26.68333
26.4764	0	4.123106	8.246211	11
25.01999	4.123106	0	4.123106	7.071068
24.18677	8.246211	4.123106	0	3.605551
26.68333	11	7.071068	3.605551	0

表 5-2-1 原始位置对于其余部队点距离表

再进行时间函数的建立，分别建立 A 型飞机和 B 型飞机一次来回的函数式，在进行 A 型和 B 型飞机每一种飞机的最终执行任务的运行次数，最后确立时间函数模型如下：

$$T_{A_j} = 40 + 2\frac{S_{A_j}}{V_A}$$

$$T_{B_j} = 70 + 2\frac{S_{B_j}}{V_B}$$

$$T'_{A_i} = \sum_{j=1}^{j=4} C_{A_{ij}} * T_{A_j} - \frac{S_{A_j}}{V_A} < 400$$

$$T'_{B_i} = \sum_{j=1}^{j=4} C_{B_{ij}} * T_{B_j} - \frac{S_{B_j}}{V_B} < 400$$

$$minT' = \begin{cases} T'_{A_{imax}}, T'_{A_{imax}} > T'_{B_{imax}} \\ T'_{B_{imax}}, T'_{B_{imax}} > T'_{A_{imax}} \end{cases}$$

基于时间和数量模型，可以求出目标函数，再引入遗传算法，进行最优次数的求解，进行组合分析，最终可以实现 A 型与 B 型飞机的最优分配方案，对于遗传算法，其原理图如下:

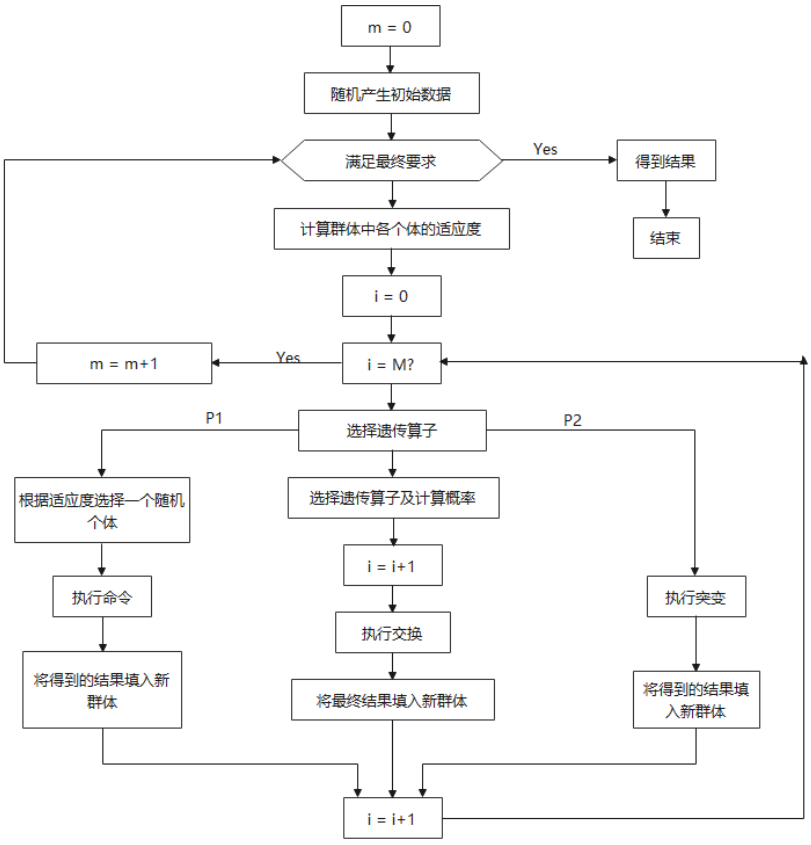

图 5-2-1：遗传算法原理图

对于二进制编码与格雷码之间的相互转换：

$$\begin{cases} g_m = b_m \\ g_i = b_{i+1} \oplus b_i, \quad i = m-1, m-2, \dots, 1 \end{cases}$$

$$\begin{cases} b_m = g_m \\ b_i = g_{i+1} \oplus g_i, \quad i = m-1, m-2, \dots, 1 \end{cases}$$

假设有n个目的地分别记为$C_1, C_2, \dots, C_n$，则$[C_1, C_2, \dots, C_n]$就可构成一个表示整体个体。之后对种群和适应度函数基于所需条件进行设定，实现目标函数与适应度函数的转换：

最大值问题：F（X）= $\begin{cases} f(X)+C_{min}, & if\ f(X)+C_{min}>0 \\ 0, & if\ f(x)+C_{min}\le 0 \end{cases}$

最小值问题：F（X）= $\begin{cases} C_{max}-f(X), & if\ f(X)<C_{max} \\ 0, & if\ f(X)\ge C_{max} \end{cases}$

进行适应度尺度的变换，利用轮盘赌选择法：

$$P_i=\frac{F_i}{\sum_{i=1}^{M}F_i}$$

进行交叉和变异：

$$a_i'=\begin{cases} 1-a_i & 若x_i\le p_m \\ a_i & 若x_i>p_m \end{cases} \qquad i\in\{1,2,3,\dots,n\}$$

最终基于停止和约束条件约束条件：

$$F'(X)=\begin{cases} F(X) & X\ 满足约束条件 \\ F(X)-P(X) & X\ 不满足约束条件 \end{cases}$$

确定最后的所需要的最终结果。

5.2.3 问题解决

首先基于所要实现的目的进行一个梳理，形成最优方案流程图如下：

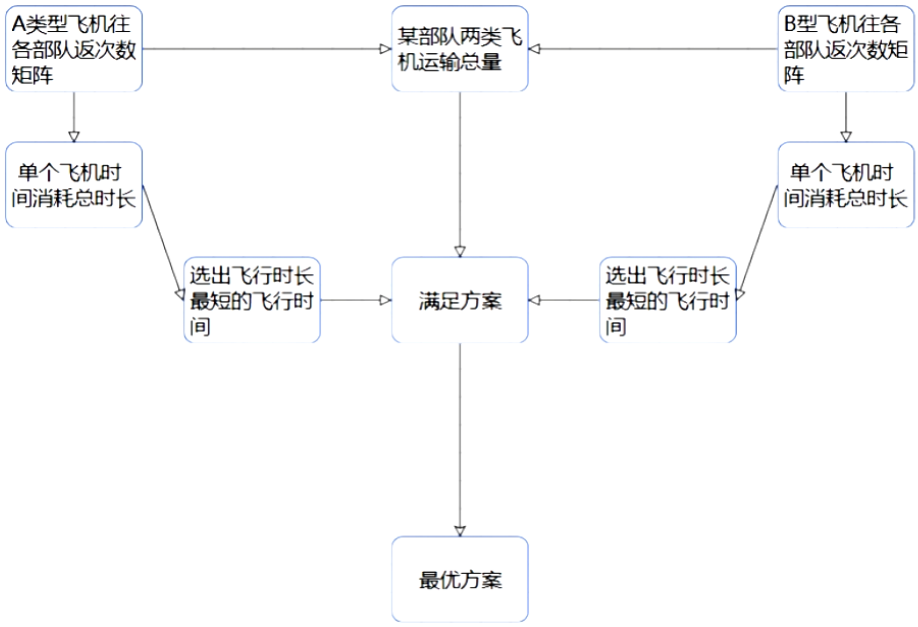

图 5-2-2 最优方案流程图

基于流程图利用运输数量数学模型和时间数学模型进行目标函数的确定，最后利用遗传算法进行一个最优来回次数的确定，运行结果如下：

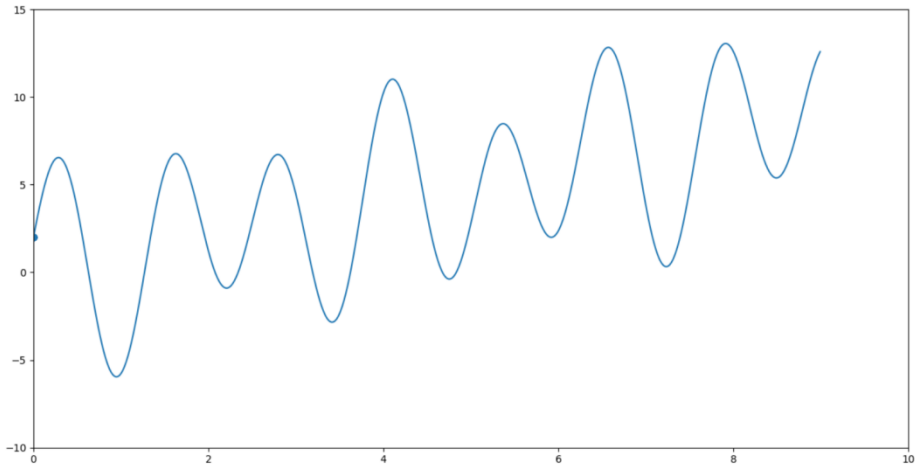

图 5-2-3 遗传算法最优次数

得出最优次数，利用以构建数学模型，进行处理，形成最优方案分析。

5.2.4 结果及分析

基于所构建数量数学函数模型和时间数学函数模型，可以确定目标函数，对于未知变量来回次数可利用遗传算法进行一个处理，得到次数结果，最后再把次数结果带回到所构建的函数模型中，基于时间和运输数量，再结合和 A 型 B 型速度之比可得到的最优方案如下：

A 型飞机首次运行装载满货物，全体出动，对于最远距离的 A102、B404 进行货物的运输，由于遗传算法结果显示约为 2 次，表明 A 型飞机运输到目的地要经过两次才能完成，其中，首次要进行货物的满载，第二次，只需要有一架飞机进行剩余缺少物资的运输便可。B 型飞机全员对于最近距离 B121 和 A122，可以进行 B 型飞机的拆分，其中二分之一带物资前往 B121，剩下前往 A122 位置进行运输，物资载满，只运输一次便可。

经检验分析得到，此方案满足最短时间内以最少来回次数，最少派出次数，完成了运输需求的满足，因此此方案满足条件，且比较完备。

5.3 问题三

5.3.1 问题分析

对于第三题，各个部队接到上级要求，在收到物资后进行任务转移，但是题干中并未表述转移的速度条件，所以要进行假设法分析，对于可能的情况进行分类假设，对于前往下一个部队点进行任务这一条件来说，不可以随机进行，否则会产生 256 种可能，所以我们可以限定条件，利用蚁群算法，进行最优路线的求解，从而确定路线规划，实现部队转移的确定，在紧接着，我们进行部队间相关条件的假设，这里可以分为两种情况，第一种就是部队间无相关联系，即送往第一个点后，前一个个部队点的

移动与后一个部队点的物资接收无关，各个部队点之间是相互独立的关系，第二种是部队间有相关性，前一支部队可以携带物资运往后一支部队，对于后面的部队进行物资的补充，从而影响飞机所携带的物资，和调用的数量。再结合最新的数量限制条件和损耗条件，进行新的方案的设计，进行多变量优化，形成最新的优化模型，进行求解。

5.3.2 条件处理及模型建立

首先基于问题二的距离路径和蚁群算法进行一个最优路径规划问题，形成一个最优的部队转移路径：

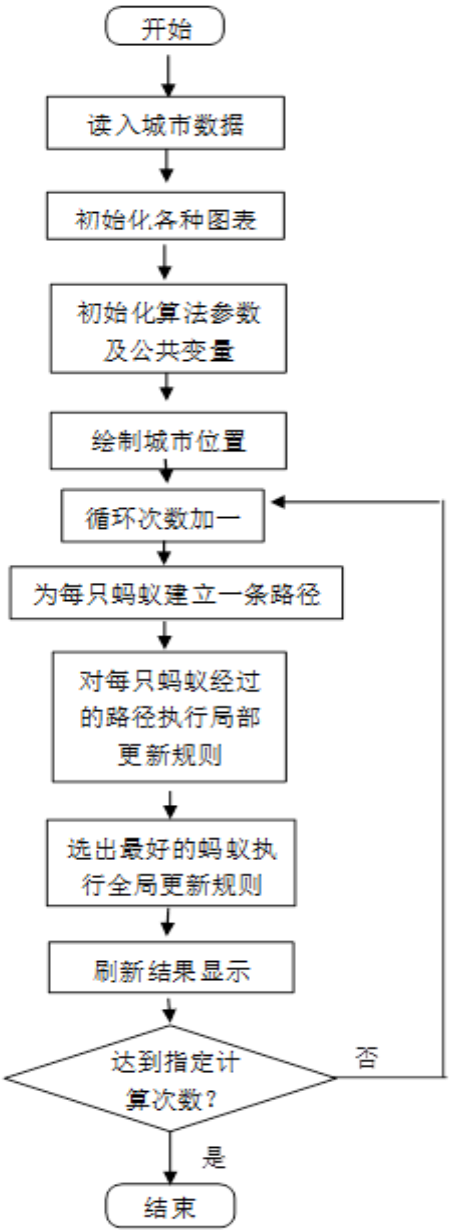

图 5-3-1 蚁群算法原理图

基于蚁群算法，首先得到从给定起始点形成的一个循环路径（图中因为变量原因，坐标点

进行了十倍增加），并且得到了最优的距离数值：

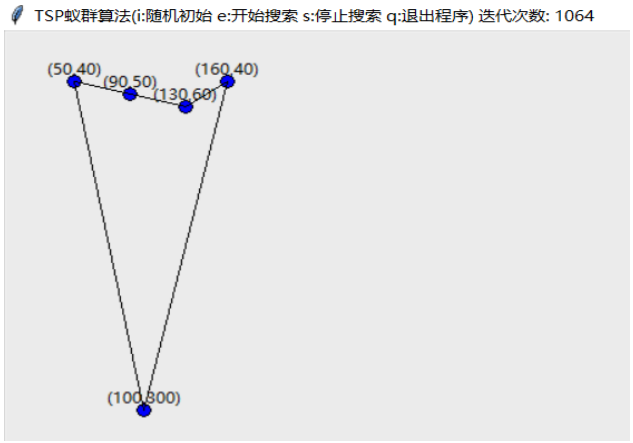

图 5-3-2 基于补给站的最优路线图

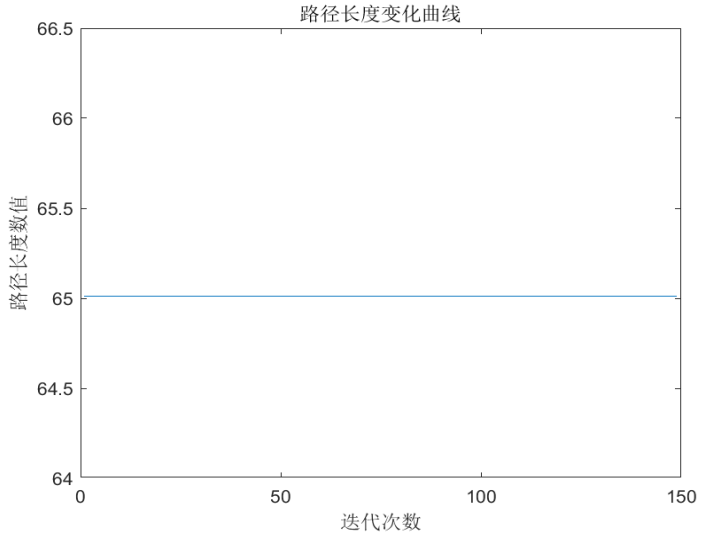

图 5-3-3 基于补给站的最优路线距离图

但是要进行转移，那么要实现一个部队间的最有路线求解，模型构建如下：

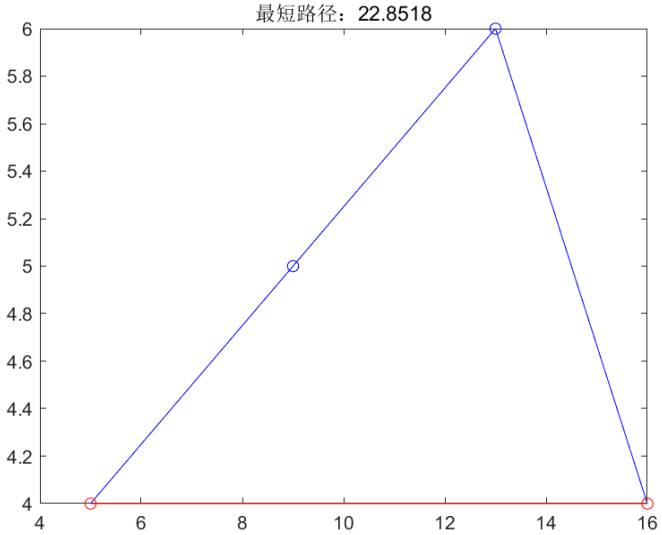

图 5-3-4 部队转移的最优路线图

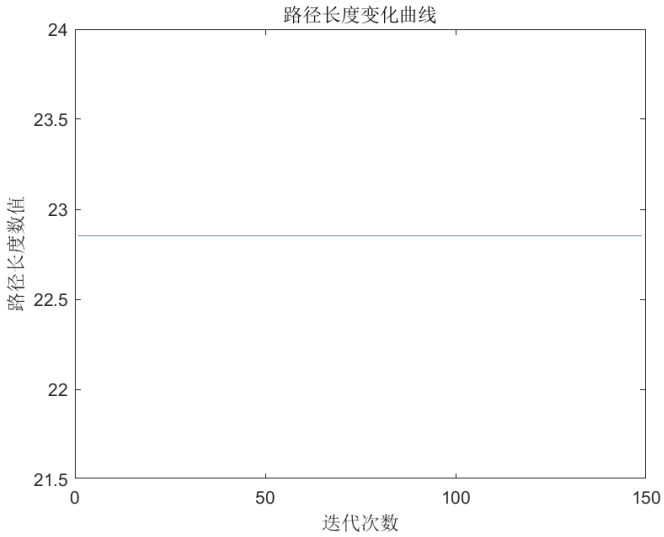

图 5-3-5 部队转移的最优路线距离图

由此可知，最优路线解是按照 A102-A122-B121-B404-A102 的路线进行转移的，所得到的结果也是最短路径。

最优转移路径已得到，下面就要进行新的距离模型和时间模型的构建，基于第二问，对于第一类各部队不相关的情况，所构建的模型如下：

$$C_{A_{ij}} = \begin{pmatrix} C_{A_{11}} & \cdots & C_{A_{14}} \\ \vdots & \ddots & \vdots \\ C_{A_{61}} & \cdots & C_{A_{64}} \end{pmatrix}$$

$$C_{B_{ij}} = \begin{pmatrix} C_{B_{11}} & \cdots & C_{B_{10\,4}} \\ \vdots & \ddots & \vdots \\ C_{B_{10\,1}} & \cdots & C_{B_{10\,4}} \end{pmatrix}$$

其中，i，j 分别代表第几架飞机和目的地位置。

紧接着，对 A 型和 B 型飞机进行一个运输数量的函数关系构建，形成 A、B 型飞机运输数量函数模型如下：

$$Q1 * \sum_{i=1}^{i=6} C_{A_{ip}*13} \leq L_{A_p}$$

$$Q2 * \sum_{i=1}^{i=10} C_{B_{ip}*20} \leq L_{B_p}$$

由 A、B 型运输数量便可得出某一部队位置所受到的数量函数模型如下（其中 p 与 j 含义相同）：

$$L_j = \left(L_{A_j} + L_{B_j}\right)$$

$$1 \leq \frac{L_j}{L_{jmin}} \leq 1.1$$

再进行时间函数的建立，分别建立 A 型飞机和 B 型飞机一次来回的函数式，在进行 A 型和 B 型飞机每一种飞机的最终执行任务的运行次数，最后确立时间函数模型如下：

$$T_{A_j} = 40 + 2\frac{S_{A_j}}{V_A}$$

$$T_{B_j} = 70 + 2\frac{S_{B_j}}{V_B}$$

$$T'_{A_i} = \sum_{j=1}^{j=4} C_{A_{ij}} * T_{A_j} - \frac{S_{A_j}}{V_A} < 400$$

$$T'_{B_i} = \sum_{j=1}^{j=4} C_{B_{ij}} * T_{B_j} - \frac{S_{B_j}}{V_B} < 400$$

$$minT' = \begin{cases} T'_{A_{imax}}, T'_{A_{imax}} > T'_{B_{imax}} \\ T'_{B_{imax}}, T'_{B_{imax}} > T'_{A_{imax}} \end{cases}$$

$$1.03 \leq （Q1） \leq 1.07$$

$$1.05 \leq （Q2） \leq 1.1$$

对于第二类部队相关的情况，构建相对应的数量模型和时间模型如下：

$$C_{A_{ij}} = \begin{pmatrix} C_{A_{11}} & \cdots & C_{A_{14}} \\ \vdots & \ddots & \vdots \\ C_{A_{61}} & \cdots & C_{A_{64}} \end{pmatrix}$$

$$C_{B_{ij}} = \begin{pmatrix} C_{B_{11}} & \cdots & C_{B_{10\,4}} \\ \vdots & \ddots & \vdots \\ C_{B_{10\,1}} & \cdots & C_{B_{10\,4}} \end{pmatrix}$$

令 A102、A122、B121、B404 分别用 1、2、3、4 地进行代替，构建的运输量函数公式模型如下：

A 类飞机飞往 1 地的运输货量：

$$L_{A_1} \geq Q1 * \sum_{i=1}^{i=6} C_{A_{i1}*13}$$

B 类飞机飞往 1 地的运输货量：

$$L_{B_1} \geq Q2 * \sum_{i=1}^{i=10} C_{B_{i1}*20}$$

A 类飞机飞往 2 地的运输货量：

$$L_{A_2} \geq Q1 * \sum_{i=1}^{i=6} C_{A_{i2}*13} - L_{A_1} * Q3$$

B 类飞机飞往 2 地的运输货量：

$$L_{B_2} \geq Q2 * \sum_{i=1}^{i=10} C_{B_{i2}*20} - L_{B_1} * Q3$$

A 类飞机飞往 3 地的运输货量：

$$L_{A_3} \geq Q1 * \sum_{i=1}^{i=6} C_{A_{i3}*13} - L_{A_2} * Q3$$

B 类飞机飞往 3 地的运输货量：

$$L_{B_3} \geq Q2 * \sum_{i=1}^{i=10} C_{B_{i3}*20-L_{B_2}*Q3}$$

A 类飞机飞往 4 地的运输货量：

$$L_{A_4} \geq Q1 * \sum_{i=1}^{i=6} C_{A_{i4}*13} - L_{A_3} * Q3$$

B 类飞机飞往 4 地的运输货量：

$$L_{B_4} \geq Q2 * \sum_{i=1}^{i=10} C_{B_{i4}*20} - L_{A_3} * Q3$$

j 地总货量：

$$L_j = \left(L_{A_j} + L_{B_j}\right)$$

$$1 \leq \frac{L_j}{L_{jmin}} \leq 1.1$$

A、B 型飞机进行一次来回的时间：

$$T_{A_j} = 40 + 2\frac{S_{A_j}}{V_A}$$

$$T_{B_j} = 70 + 2\frac{S_{B_j}}{V_B}$$

A 型飞机组合总时间：

$$T'_{A_i} = \sum_{j=1}^{j=4} C_{A_{ij}} * T_{A_j} - \frac{S_{A_j}}{V_A} < 400$$

B 型飞机组合总时间：

$$T'_{B_i} = \sum_{j=1}^{j=4} C_{B_{ij}} * T_{B_j} - \frac{S_{B_j}}{V_B} < 400$$

$$minT' = \begin{cases} T'_{A_{imax}}, T'_{A_{imax}} > T'_{B_{imax}} \\ T'_{B_{imax}}, T'_{B_{imax}} > T'_{A_{imax}} \end{cases}$$

经过遗传算法构建，得到最新的最优来回次数（橙红色区域为最适合）：

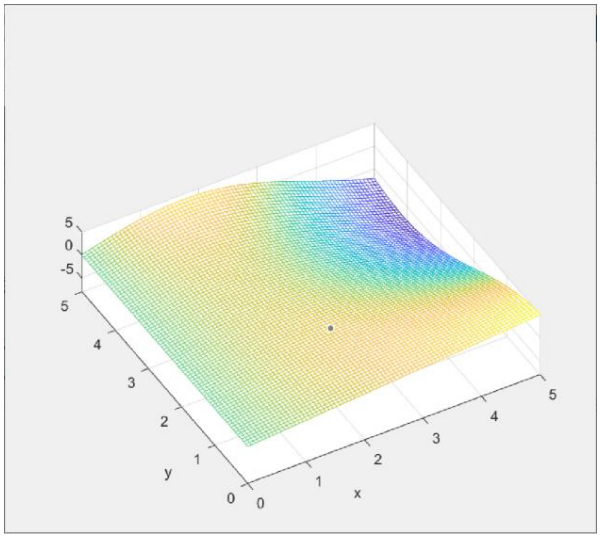

图 5-3-6 最优次数图

5.3.3 问题解决

基于第一类：

由于相互之间互不干扰，所以可以直接利用距离和时间模型，利用遗传算法得到最优次数解，求解思路与第二问类似，但是此时数量的限制条件更大，可以进行二分法思路的划分，兵分两路同时进行，应对与数量模型求解结果进行新的方案调整。

基于第二类：

由于相互之间相关，并且，前一个部队点可以携带多余的 0%-10%的物资支援后一个部队点，所以，要在基于数量和时间限制的前提下，对于多与数量的物资进行最合理化划分，从而进行方案设计。

5.3.4 结果及分析

第一类：

A 型飞机平分为两队，分别和一个 B 型飞机进行组合，飞往 B121、B404，A 型飞机每次均带满物资，B 型飞机所携带物资与 A 两次飞行多携带的物资不多于所需的 10%便可，基于最优次数两次，和所用时间分析可知，A 型飞机运输两次，B 型飞机运输一次，完成此两地的物资运输，剩下的 B 型飞机，携带满物资，平分两队，分别前往 A102、A122 即可实现最优调度。

第二类：

对于 A102 点，直接分 4 支 B 型飞机，载满处理进行物资运输，同时 B121 点分 4 支 B 型飞机载满，B404 点分 2 支 B 型飞机载满，A122 直接将 A 型飞机全部载满前往此地，由 A，B 型飞机所用时间之比可知，A 型飞机已完成后，B 还未到达，此时完成一次运输的 A 型飞机在进行分组，分出 2 支前往 B121 点其中一支载满，另一只载 3 吨，3 支前往 b404 点，其中两支载满，最后一支携带 10 吨便可。

经检验分析可知，应对与两种不同的情况，此两类的调用方案均满足条件，且均符合运输时间少，次数少，运载量不浪费等因素，所以方案设计合理。

六、结果分析

对于问题一,基于求解 fminsearch 的函数模型和粒子群算法的模型来进行重新分类构建一个数学模型,可以被用来直接确定所实际需要到被求解的函数中的原始目标函数,在求解 fminsearch 函数的方法过程中,将原始需求点坐标变换为坐标,得到的一个初始化的输入坐标数据,所实际的需要到的原始目标输入函数的坐标参数通常有以下这样的两个,分别做为分别输入的目标函数值和目标变量的初值,变量的坐标初值其实也并不一定就单只是指在需求点坐标上的那一个横坐标,也有一部分可以单独看成它是个二维变量的甚至是有些可能是一个三维变量的,在此题中也可分别看做它们是哪一个的二维变量,再分别通过使用由 Matlab 等软件中自带出来的函数进行计算后就又可以分别计算得到以费马点坐标为基础(9,5)的,由随机优化算法可实现的粒子群算法,将当前每个粒子需求点群都可看做为一个粒子,初始化当前粒子所有的需求粒子群迭代的速度,计算粒子适应度,将所有粒子适应度都达到一个最佳状态后的每个粒子群体都作为当前一个粒子最佳状态的个体,再将当前粒子的这一个最后的一批需求粒子群体也作为前一个粒子最佳的群体,更新当前各个需求粒子群体迭代的最新速度以及最新的位置,重新计算对目前每个可迭代的粒子群的粒子适应度,与最佳个体进行比较并进行更换,直到满足优化的目标或迭代粒子次数达到上限时得到的最优费马点位置(9,5),两种模型得到的费马点位置相同,可以看做(9,5)为新的供给点的修建地点。

对于问题二，基于所构建数量数学函数模型和时间数学函数模型，可以确定目标函数，对于未知变量来回次数可利用遗传算法进行一个处理，得到次数结果，最后再把次数结果带回到所构建的函数模型中，基于时间和运输数量，再结合和 A 型 B 型速度之比可得到的最优方案如下：

A 型飞机每次运行装载满货物，全体出动，对于最远距离的 A102、A122、B404 进行货物的运输，B 型飞机全员对于最近距离 B121 位置进行运输，最后可在用时 93min 完成全部物资的运输。

经检验分析得到，此方案满足最短时间内以最少来回次数完成了运输需求的满足，因此此方案满足条件，且比较完备。

对于问题三，考虑到两种情况，第一类：A 型飞机平分为两队，分别和一个 B 型飞机进行组合，飞往 B121、B404，A 型飞机每次均带满物资，B 型飞机所携带物资与 A 两次飞行多携带的物资不多于所需的 10%便可，基于最优次数两次，和所用时间分析可知，A 型飞机运输两次，B 型飞机运输一次，完成此两地的物资运输，剩下的 B 型飞机，携带满物资，平分两队，分别前往 A102、A122 即可实现最优调度。

第二类：对于 A102 点，直接分 4 支 B 型飞机，载满处理进行物资运输，同时 B121 点分 4 支 B 型飞机载满，B404 点分 2 支 B 型飞机载满，A122 直接将 A 型飞机全部载满前往此地，由 A，B 型飞机所用时间之比可知，A 型飞机已完成后，B 还未到达，此时完成一次运输的 A 型飞机在进行分组，分出 2 支前往 B121 点其中一支载满，另一只载 3 吨，3 支前往 b404 点，其中两支载满，最后一支携带 10 吨便可。

经检验分析可知，应对与两种不同的情况，此两类的调用方案均满足条件，且均符合运输时间少，次数少，运载量不浪费等因素，所以方案设计合理。

七、模型评价与推广

基于以上优化调度模型中存在的最优调度路径的选择等问题,可以选择同时使用粒子群算法和构造人工蚁群,来最终实现或解决最优化的调度模型问题。首先即是对于全局的最优浮点值的精确的选取方法也就可以直接考虑并通过粒子群算法规则来得到有效解决,它显然也远比于遗传算法规则要来得的更为有效直接而简单,它也就是为什么相比只适用于遗传算法规则中的交叉操作规则和变异操作的规则来说,粒子群算法则能够有效通过追随当前所搜索到全局的局部最优值来有效进行寻找到当前全局的最优,使用此算法其得出的数据精度高,收敛快等优势,对于此类问题更加符合,在解决实际问题中拥有很大优越性。而对于我们来讲目前若要直接应用遗传算法去直接解决 TSP 的问题,主要要同时考虑去解决自然编码问题和算子库的优化改进与设计等两个基本问题.编码的运算处理方式的直接约束往往影响决定了编码运算可解的空间范围的相对大小,好的自然编码的处理计算方式通常就可以用来适当地压缩编码运算可以求解问题的空间,提高自然编码的运算与求解问题效率.常见的自然编码处理运算处理方式一般主要有二进制编码,实值编码,自然编码等本文过程中我们主要侧重在讨论了在自然编码等计算编码方式的约束情况下算子群设计上的算法优化及改进等问题及其可应用与 MATLAB 等软件之上的算法程序群设计及其实现.针对 TSP 问题,提出采用了贪婪交叉变异算子方法和倒位交叉变异算子方法等来能够有效的加快了算法群设计的收敛速度,同时使算法设计又变得较容不易地陷入到了局部的最优,从而能够又可以较好的地实现有效解决解了算法群体结构上的多样性问题和在提高算法收敛速度问题中遇到的各种结构性的予矛盾。所以将两者结合起来,打到的预期效果将会更加准确,在时间上能有很大的提升。

（一）模型的优点

减少大的计算量，通过群体中个体之间的协作和信息共享来寻找最优解。

较好的解决了在路径规划中遇到的时间性限制问题。

使用蚁群算法，对路径进行分布计算，避免重复路径和物资运输中带来的影响。

遗传算法的全局搜索能力比较强,对于交叉可能性高的情况,可以产生更多的个体,能够提全局搜索的范围。

Fmonsearch 函数是一个使用无导数法来计算一个无约束点的多值变量函数的最小值,也这样就使得非常特别地适合于费尔雪马点的搜索。粒子群算法中由于没有任何交叉运算和变异运算,依靠一个粒子速度就可以直接完成信息搜索,并且即使在迭代的进化算法中也会只有一个最优的一个粒子速度就可以保证同时的把所有的信息都同时的传递给另外一个和其他类似的粒子,搜索信息的粒子速度将会是非常之地的快,且由于具有的高的记忆性,记忆群体算法中最好的一个粒子历史和最好的粒子位置就完全可以同时进行记忆并保证直接地传递所有信息给所有其他类型的粒子,需要人工进行调整的粒子参数也都还比较地的少,结构上很简单,易于工程化地加以实现。

（二）模型的缺点

蚁群算法容易收到环境因素的影响，当环境变化很大时会出现偏差。

算法计算量大,得到结果所需时间较长

（三）模型的改进

模型上的一些改进本题也有一些可以尝试的将蚁群模型等和蚁遗传算子模型等模型进行了有效地结合,"蚁群遗传算子模型等主要内容是指通过将每个蚁优化遗传算子函数中的任意一个可能的解表示抽象化成的每一个个体,每个蚁个体再用这其中的一定的编码方式来形成下一个蚁基因,借助这个蚁优化遗传的算子,选择、交叉、变异和操作,对其种群结构进行了动态演化,选择和繁殖创造出的一些可更快的适应新的环境需求变化的种群。在路径的规划的过程中,我们应该首先是将每一条路径规划为每一个种群个体,每个种群中又必须有一个至多的 n 个种群个体,即至少得有一个至少有 n 条的路径,同时,每个种群个体中至少得又得会有一个至少的 m 个染色体,即每个中间的过渡点上的染色体的个数,每个中间过渡点上的路径(染色体)的数量又得会有一个至少有两个的维度(x,y),在代码中也可以用 Genx 和 geny 表示了每一个种群。通过研究每一代种群个体的遗传演化,对各个种群个体进行遗传算子的优化操作,选择匹配出比较合适种群的种群个体(最优路径)。

八、参考文献

[1] 刘紫玉,赵丽霞,薛建越,陈军霞,宋伟.面向车辆路径问题的改进蚁群算法研究[J].河北科技大学学报,2022,43(01):80-89.
[2] 李涛,赵宏生.基于进化蚁群算法的移动机器人路径优化[J/OL].控制与决策:1-9[2022-03-19].DOI:10.13195/j.kzyjc.2021.1324.
[3] 代婷婷,刘秀,虎亚楠.改进的蚁群算法在路径规划问题中的应用[J].佳木斯大学学报(自然科学版),2022,40(01):123-125.
[4] 丁建立, 陈增强, 袁著祉. 遗传算法与蚂蚁算法的融合[J]. 计算机研究与发展, 2003, 40(9):6.
[5] 张文修, 梁怡. 遗传算法的数学基础[J]. 西安交通大学学报, 2000.
[6] 刘勇 刘祥生. 组合数学(面向 21 世纪全国高职高专数学规划教材)[M]. 北京大学出版社, 2006.
[7] 裴卫东. 汽车售后配件配送运输优化方案的研究[D]. 上海交通大学, 2007.
[8] 李晓川, 朱晓敏, 赵乃东. 基于 Lingo 的运输优化系统设计与开发[J]. 物流技术, 2010(3):106-109.

九、附录

fminsearch 函数：

```
    clear all;
    close all;
    clc;
data = [5 4;9 5;13 6;16 4];
n = length(data);
```

```
distance = zeros(1, n);
fun = @(x) sum(sqrt(sum((data-repmat(x,n,1)).^2, 2)));
for i = 1:n
    distance(i) = fun(data(i,:));
end
[~,index] = min(distance);
x0 = data(index,:);
[best_point, min_dis] = fminsearch(fun, x0);
disp([,num2str(min_dis),...
      ,num2str(best_point(1)), ,num2str(best_point(2))]);
figure
plot(data(:,1),data(:,2),);
axis equal
hold on
plot(best_point(1),best_point(2),);
```

粒子群算法：

```
clear, close all
data = [5 4;9 5;13 6;16 4];
fun = @(x) sum(sqrt(sum((data-x).^2, 2)));
c1 = 1.49445;
c2 = 1.49445;
evolution_max = 200;
pop_size = 1000;
x_max = max(data(:,1));
x_min = min(data(:,1));
y_max = max(data(:,2));
y_min = min(data(:,2));
v_x_max = x_max-x_min;
v_x_min = -v_x_max;
v_y_max = y_max-y_min;
v_y_min = -v_y_max;
pop = zeros(pop_size, 2);
v = zeros(pop_size, 2);
fitness = zeros(1, pop_size);
pop(:,1) = x_min + (x_max-x_min) * rand(pop_size,1);
pop(:,2) = y_min + (y_max-y_min) * rand(pop_size,1);
v(:,1) = v_x_min + (v_x_max-v_x_min) * rand(pop_size,1);
v(:,2) = v_y_min + (v_y_max-v_y_min) * rand(pop_size,1);
for i = 1: pop_size
    fitness(i) = fun(pop(i,:));
end
[fitness_best, index_best] = min(fitness);
I = pop(index_best,:);
P = pop;
I_fitness_best = fitness_best;
P_fitness_best = fitness;
fitness_evolution = zeros(evolution_max, 1);
position_evolution = zeros(evolution_max, 2);
for i = 1: evolution_max
```

```
        fitness_evolution(i) = I_fitness_best;
        position_evolution(i,:) = I;
        v(:,1) = v(:,1) + c1*rand*(I(1)-pop(:,1)) + c2*rand*(P(:,1)-pop(:,1));
        v(:,2) = v(:,2) + c1*rand*(I(2)-pop(:,2)) + c2*rand*(P(:,2)-pop(:,2));
        pop(:,1) = pop(:,1) + v(:,1);
        pop(:,2) = pop(:,2) + v(:,2);
        pop(pop(:,1) > x_max, 1) = x_max;
        pop(pop(:,1) < x_min, 1) = x_min;
        pop(pop(:,2) > y_max, 2) = y_max;
        pop(pop(:,2) < y_min, 2) = y_min;
        for j = 1: pop_size
            fitness(j) = fun(pop(j,:));
        end
        index = fitness < P_fitness_best;
        P_fitness_best(index) = fitness(index);
        P(index,:) = pop(index,:);
        [fitness_best, index_best] = min(fitness);
        if fitness_best < I_fitness_best
            I_fitness_best = fitness_best;
            I = pop(index_best,:);
        end
end
figure, plot(fitness_evolution), title()
xlabel(), ylabel();
fprintf(I(1), I(2))
fprintf(', I_fitness_best)
figure
for i = 1: evolution_max
        plot(data(:,1), data(:,2),);
        hold on
        plot(position_evolution(i,1), position_evolution(i,2),);
        hold off
        m(:,i) =getframe;
end
```

遗传算法：

```
    from __future__ import division
    import math
    import numpy as np
    import matplotlib.pyplot as plt
    from matplotlib import animation
    fig = plt.figure(figsize=(15,10))
    ax1 = fig.add_subplot(111)
    ax1.set_xlim([0,10])
    ax1.set_ylim([-10,15])
    def initFunction(x):
        #y = ∑_(i=1)^(i=6)▒X_(X*13)
        #y = ∑_(i=1)^(i=6)▒X_(X*20)
        return y
```

```
x = [i / 100 for i in range(900)]
y = [i%900 for i in range(900)]
for i in range(900):
    y[i] = initFunction(x[i])
ax1.plot(x,y)
population = []
for i in range(20):
    strCode = ''
    for j in range(5):
        strCode += str(np.random.randint(0,2))
    population.append(strCode)
def decode(genCode):
    y = 0 + int(genCode,2)/(2**17-1)*9
    return y

m = [m for m in range(len(population))]
n = [n for n in range(len(population))]
for i in range(len(population)):
    n[i] = initFunction(decode(population[i]))
    m[i] = decode(population[i])
sca1 = ax1.scatter(m,n)
def fitnessfunction(population,initFunction):
    value = []
    for i in range(len(population)):
        value.append(initFunction(decode(population[i])))
        if value[i] < 0:
            value[i] = 0
    return value
def selectChild(population,fitnessResult):
    childPopulation = []
    Sumfitness = []
    indiviSumP = []
    for i in range(len(fitnessResult)):
        if i == 0:
            Sumfitness.append(fitnessResult[i])
        else:
            Sumfitness.append(Sumfitness[i-1] + fitnessResult[i])
    for j in range(len(Sumfitness)):
        indiviSumP.append(Sumfitness[j]/sum(fitnessResult))
    for childNum in range(len(population)):
        x = np.random.uniform(0,1)
        if x <= indiviSumP[0]:
            childPopulation.append(population[0])
        else:
            for m in range(1,len(population)):
                if indiviSumP[m-1] <= x and x <= indiviSumP[m]:
                    childPopulation.append(population[m])
                    break
    return childPopulation
```

```
def crossParent(new_population,pc):
    crossChildPopulation = []

    if len(new_population)%2 != 0:
        randmSingleNum = np.random.randint(0,len(new_population))
        crossChildPopulation.append(new_population[randmSingleNum])
        del(new_population[randmSingleNum])

    half = int(len(new_population)/2)
    father = new_population[:half]
    mother = new_population[half:]
    np.random.shuffle(father)
    np.random.shuffle(mother)

    for i in range(half):
        crossP = np.random.uniform(0,1)
        if pc >= crossP:
            breakPoint = np.random.randint(0,len(new_population[0]))
            son = father[i][:breakPoint] + mother[i][breakPoint:]
            daughter = mother[i][:breakPoint] + father[i][breakPoint:]
        else:
            son = father[i]
            daughter = mother[i]
        crossChildPopulation.append(son)
        crossChildPopulation.append(daughter)
    return crossChildPopulation
def variation(crossChildPopulation,pc):
    variationPopulation = []
    for i in range(len(crossChildPopulation)):
        variationP = np.random.uniform(0,1)
        variationPos = np.random.randint(0,len(crossChildPopulation[0]))
        if variationP <= pc:
            if variationPos == 0:
                if crossChildPopulation[i][0] == '0':
                    crossChildPopulation[i] = '1'+crossChildPopulation[i][1:]
                else:
                    crossChildPopulation[i] = '0'+crossChildPopulation[i][1:]
            else:
                if crossChildPopulation[i][variationPos] == '0':
                    crossChildPopulation[i] =
    crossChildPopulation[i][:variationPos-1] + '1' +
    crossChildPopulation[i][variationPos:]
                else:
                     crossChildPopulation[i] =
    crossChildPopulation[i][:variationPos-1] + '0' +
    crossChildPopulation[i][variationPos:]
        variationPopulation.append(crossChildPopulation[i])
    return variationPopulation
def updata(i):
```

```
        global population
        x1 = [0 for j in range(len(population))]
        y1 = [0 for j in range(len(population))]
        fig.set_size_inches(15,10)#控制再保存图片的时候窗口不要缩

        fitnessValue = fitnessfunction(population,initFunction)
        selectChildes = selectChild(population,fitnessValue)
        crossChild = crossParent(selectChildes,0.7)
        population = variation(crossChild,0.02)
        for i in range(len(population)):
            x1[i] = decode(population[i])
            y1[i] = initFunction(decode(population[i]))
        data = [[x,y] for x,y in zip(x1,y1)]
        sca1.set_offsets(data)
        print(list(zip(x1,y1)))
        return sca1
    ani = animation.FuncAnimation(fig = fig,func=updata,frames=np.arange(1,20),init_func
        = None,interval = 500,blit = False)
plt.show()
```

遗传算法：

```
clear all;
close all;
clc;
C = [10 30;5 4;9 5;13 6;16 4];
% figure(1);
% scatter(C(:,1),C(:,2),'k','d');
[M,N] = size(C);
distance = zeros(M,M);
for m=1:M
    for n=1:M
        distance(m,n) = sqrt(sum((C(m,:)-C(n,:)).^2));
    end
end
m = 50;
alpha = 1;
beta = 5;
rho = 0.25;
G = 150;
Q = 100;
Eta = 1./distance;
Tau = ones(M,M);
Tabu = zeros(m,M);
gen = 1;
R_best = zeros(G,M);
L_best = inf.*ones(G,1);
while gen<G

    random_pos = [];
```

```
for i=1:(ceil(m/M))
    random_pos = [random_pos,randperm(M)];
end
Tabu(:,1) = (random_pos(1,1:m))';
for i=2:M
    for j=1:m
        visited = Tabu(j,1:(i-1));
        % visited=visited(1,:);
        unvisited = zeros(1,(M+1-i));
        visit_P = unvisited;
        count = 1;
        for k=1:M
            if isempty(find(visited==k))
                unvisited(count) = k;
                count = count+1;
            end
        end
        for k=1:length(unvisited)       % Tau(visited(end),unvisited(k))
            visit_P(k) =
((Tau(visited(end),unvisited(k)))^alpha)*(Eta(visited(end),unvisited(k))^beta);
        end
        visit_P = visit_P/sum(visit_P);
        Pcum = cumsum(visit_P);
        selected = find(Pcum>=rand);
        to_visited = unvisited(selected(1));
        Tabu(j,i) = to_visited;
    end
end
if gen>=2
    Tabu(1,:) = R_best(gen-1,:);
end

L = zeros(1,m);
for i=1:m
    R = Tabu(i,:);
    L(i) = distance(R(M),R(1));
    for j=1:(M-1)
        L(i) = L(i)+distance(R(j),R(j+1));
    end
end
L_best(gen) = min(L);
pos = find(L==L_best(gen));
R_best(gen,:) = Tabu(pos(1),:);
Delta_Tau = zeros(M,M);
for i=1:m
    for j=1:(M-1)
        Delta_Tau(Tabu(i,j),Tabu(i,j+1)) = Delta_Tau(Tabu(i,j),Tabu(i,j+1)) +
Q/L(i);
    end
```

```
            Delta_Tau(Tabu(i,M),Tabu(i,1)) = Delta_Tau(Tabu(i,M),Tabu(i,1)) + Q/L(i);
        end
        Tau = (1-rho).*Tau+Delta_Tau;
        Tabu = zeros(m,M);

        for i=1:(M-1)

        plot([C(R_best(gen,i),1),C(R_best(gen,i+1),1)],[C(R_best(gen,i),2),C(R_best(gen,i+
        1),2)],'bo-');
            hold on;
        end

        plot([C(R_best(gen,n),1),C(R_best(gen,1),1)],[C(R_best(gen,n),2),C(R_best(gen,1),
        2)],'ro-');
        title(['最短路径：',num2str(L_best(gen))]);
        hold off;
        pause(0.05);
        gen = gen+1;
    end
    figure(2);
    plot(L_best);
    title('路径长度变化曲线');
    xlabel('迭代次数');
ylabel('路径长度数值');
```