

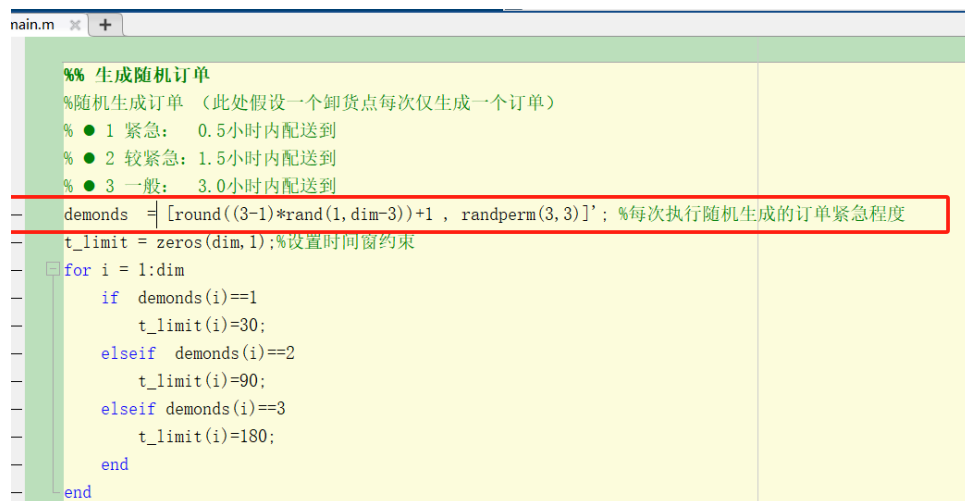
无人机配送路径规划作业报告

2023202210148 魏康丽

1 引言

本文采用了基于遗传算法的路径规划方法，优化无人机的配送路径，在满足订单优先级约束的前提下，总配送路径最短。下文将详细介绍对于该无人机配送路径规划问题我们设计的遗传算法，包括随机生成数据、配送中心聚类设计、遗传算法中的适应度函数设计、数据可视化与路径实验结果可视化。

本文使用 Matlab 编码实现，程序随机生成优先级不同的数据订单，具体订单生成指令如下图 1.1 所示：



```
%% 生成随机订单
%随机生成订单 （此处假设一个卸货点每次仅生成一个订单）
% ● 1 紧急： 0.5小时内配送到
% ● 2 较紧急： 1.5小时内配送到
% ● 3 一般： 3.0小时内配送到
demonds = [round((3-1)*rand(1,dim-3))+1, randperm(3,3)]'; %每次执行随机生成的订单紧急程度
t_limit = zeros(dim,1);%设置时间窗约束
for i = 1:dim
    if demonds(i)==1
        t_limit(i)=30;
    elseif demonds(i)==2
        t_limit(i)=90;
    elseif demonds(i)==3
        t_limit(i)=180;
    end
end
```

图 1.1 随机优先级语句

这里给出本研究的输入数据，如表 1.1 所示。表中数据在变量 `data_tem` 里存储，如图 1.2 所示，双击打开 `data_tem` 即可获取当前的优先级及其他信息。



图1.2 数据变量

表1.1 输入数据

配送中心	x	y	卸货点	优先级	距离
1	4.2	9.7	1	2	0.960469
1	2.5	9.9	8	3	1.241974
1	0.9	8	15	3	2.777139
1	6.2	8.8	19	2	2.766315
2	7.6	1.9	2	1	1.424001
2	5	0.5	21	2	1.553848
2	6.2	1.8	27	3	0.405518
3	9.8	3.2	3	3	2.477678
3	13.7	0.2	25	2	3.764158
3	13.2	7.6	28	3	4.050377
4	9.7	12.5	4	3	2.544562
4	14	9.1	7	2	3.183662
4	11.3	11.8	9	1	0.813116
4	9.9	10.3	14	1	2.583562
4	13.2	12.5	17	3	1.387369
4	11.4	11	22	1	0.958347
4	13.9	13.1	23	3	2.308536
4	12.2	10.3	24	2	1.358232
4	13.3	10.5	26	2	1.665225
4	9.6	14.1	29	1	3.497177
4	14.6	13	30	2	2.839058
5	6	14.1	5	2	0.806226
5	6.4	13.1	11	2	1.5
5	6.6	14.6	16	3	1.523155
5	1.8	14.2	20	3	3.405877
6	5.5	3.9	6	3	0.940213
6	7.3	5.3	10	2	2.333238
6	6	5.5	12	1	1.236932
6	4	5	13	1	1.077033
6	2.4	3.9	18	2	2.764417

基于上表的订单优先级顺序生成的配送路线方案与可视化分析于第四章实验结果部分中展示。

2 相关工作介绍

近两年，无人机路径规划问题掀起了研究热潮。主要研究方向包括静态路径规划和动态路径规划，前者适用于配送环境和订单信息固定的条件，后者则更适用于订单和环境动态变化的实际场景。常见的方法有：

1. **静态规划算法：**如经典的旅行商问题（TSP）算法，最近的研究通常针对有 d -relaxed 优先级的 TSP 问题，利用图论与优化算法来解决路径优化目标。
2. **动态规划算法：**如实时路径规划算法（RRT）、蒙特卡洛树搜索（MCTS）等，适应动态变化的配送环境。
3. **智能优化算法：**如遗传算法（GA）、粒子群优化（PSO）、蚁群算法（ACO）等，模拟自然界的进化或群体行为来寻找优化解。

本文主要基于遗传算法，结合 K-means 聚类算法，对配送路径进行优化，考虑到优先级订单的约束条件，优先指派高优先级任务。

3 遗传算法设计

3.1 初始化

首先载入数据坐标，该部分数据如第一章中的表 1.1 所示。由于数据中卸货点个数为 30，这里设置遗传算法的基因个数参数 $\text{dim} = 30$ 。本文设计的遗传算法可任意调整配送中心点参数，并能够适应后续算法。这里选择配送中心的数量 $\text{cnum} = 6$ ，种群大小 $\text{popsize} = 10$ 。根据上述参数生成初始种群矩阵，大小为 $\text{dim} * \text{popsize}$ 。

根据后续运行实验，调整该遗传算法参数最大迭代次数 = 150、变异概率 = 0.01、交叉概率 = 0.85 为性能最佳。该部分代码如图 3.1 所示。

```

%% 遗传算法 初始化
cnum= 6;           %调整配送中心的数量 【可调参数】
dim = 30;          % 个体包含30位基因 【这里的基因长度=卸货点（卸货点）个数】
popsize = 10;       % 种群大小 【可调参数】
genmax = 150;       % 最大搜索次数 【可调参数】
Pm = 0.01;          % 变异概率
Pc = 0.85;          % 交叉概率

```

图 3.1 初始化遗传参数

3.2 适应度计算

设计初代种群的适应度值，首先使用 K-means 聚类算法，将现有的配送点进行聚类，选择各聚类中心作为配送中心的坐标。由于该问题中的直线距离相对较近，两点间的直线距离与球面距离的误差较小，本文选择计算配送中心与卸货点间的欧拉距离作为适应度值进行优化，以最邻近（NN）作为启发式规则。该部分代码如图 3.2，图 3.3 所示。

```

%% 计算初代适应度值
bb0 = data;
[fitness,distance] =ga_fitness(bb0,popsize,Parents,a1,cnum); %调用适应度值函数
%% 每一代的个体平均适应度值
every_generation_fitness = zeros(1, genmax+1);
repeat_num_all = zeros(1, genmax); % 每一代种群中重复个体的数量
current_generation_fitness = fitness;
every_generation_fitness(1,1) = sum(fitness(1,:))/popsize;% 保存每一代的个体平均适应度值

```

图3.2 计算初代适应度值

```

for jj = 1:popsizexxxx
% 初始化V0
V0 = zeros(cnum, 2);
% 获取当前父母的染色体
Parents_01 = Parents(jj,:);
data_tem = [Parents_01, a1, bb0(:,1)];
data_tem = sortrows(data_tem, 1);
% 找出分类边界
[x1, x2] = unique(data_tem(:, 1));
% 确保有足够的起降点数量
if length(x1) < cnum
    Parents(jj,:)=[round((cnum-1)*rand(1,30-cnum))+1 , randperm(cnum,cnum)];
    Parents_01 = Parents(jj,:);
    data_tem = [Parents_01, a1, bb0(:,1)];
    data_tem = sortrows(data_tem, 1);
    % 找出分类边界
    [x1, x2] = unique(data_tem(:, 1));
end
% 计算聚类中心
for i = 1:cnum
% 确定当前类别的数据点
if i < length(x1)
    y1_tem = data_tem(x2(i):x2(i+1)-1, 2:3);
else
    y1_tem = data_tem(x2(i):end, 2:3);
end
% 使用fminsearch找到初始聚类中心
points = y1_tem;
objectiveFunction = @(x) sum(sqrt(sum((points - x(ones(size(points,1),1),:)).^2, 2))));
initialGuess = [mean(points(:,1)), mean(points(:,2))];
[V0(i,:), dis1] = fminsearch(objectiveFunction, initialGuess);
end

% 计算每个点到聚类中心的距离
bb01=[bb0(:,1:3),Parents_01];
for ii = 1:size(bb0, 1)
    for j = 1:cnum
        if bb01(ii, 4) == j
            distance(ii, jj) = calculateDistance(bb01(ii, 2), bb01(ii, 3), V0(j, 1), V0(j, 2));
        end
    end
end
% 计算fitness (即所有距离的总和)
fitness(jj) = sum(distance(:, jj));
end

[B,C,CC] = k_means(2,cnum,a1);
V0_fit = CC;
ind1= B';
end

```

图 3.3 适应度函数设计

3.3 遗传操作

构造交叉函数和变异函数，考虑到无人机配送路径的特性，本文算法随机选择交叉点，并设置变异有 1/6 的概率保持不变。对种群进行交叉、变异操作，生成新的子代，保留适应度值低的个体。合并子代和父代，迭代遗传算法进行种群筛选。迭代过程概括如下：

- ① 交叉
- ② 变异
- ③ 合并子代和父代
- ④ 通过 kmeans 聚类算法计算配送中心的坐标
- ⑤ 计算配送中心与卸货点之间的距离，将距离之和作为适应度值进行优化
- ⑥ 保留适应度值低的个体，继续下一次迭代

遗传算法迭代设计代码如图 3.4 所示，其中交叉变异部分代码如图 3.5 所示。

```
%% 搜索开始
for t = 1 : genmax
    [population]=ga_cross(Parents,Pc); % 交叉
    [Children]=ga_mut(population, Pm,cnum); % 变异
    Combination = [Parents;Children]; % 合并子代和父代
    [Parents,fit_parent]=ga_TourSel(bb0,popsize,Combination,a1,cnum); % 锦标赛选择,适应度计算在优选内完成
    every_generation_fitness(1,t+1) = sum(fit_parent(1,:))/popsize; % 父代种群适应度和:种群大小=个体平均适应度值
    data1=sortrows(Parents); % 种群中相同的个体
    [~,r]=unique(data1,'rows');
    [dataUnique,r1]=unique(data1,'rows','last');
    repeat_num_all(1,t) = max(r1-r+1);
end
```

图 3.4 遗传迭代流程

```
function [child]=ga_cross(parent,pcross)
%--
[popSize,dim]=size(parent);
t=randperm(popSize);
parent=parent(t,:); %随机打乱个体位置
child=parent;

temp1=ones(dim,1);
for i=1:popSize/2
    if rand() > pcross % 1
        continue;
    end
    pos=round(rand()*(dim-1))+1; % 随机选择交叉点
    temp1(1:pos)=child(2*i,1:pos);
    child(2*i,1:pos)=child(2*i-1,1:pos);
    child(2*i-1,1:pos)=temp1(1:pos);
end
```

```

function [population]=ga_mut(population, pm,cnum)
%pm=0.01;
[popSize,dim]=size(population);
for i=1:popSize
    for j=1:dim
        r=rand();
        %%对个体某变量进行变异
        if r<=pm
            population(i,j)=round((cnum-1)*rand()+1;%变异有1/6的概率会保持不变
        end
    end
end
end

```

图 3.5 交叉变异设计

3.4 路径规划与评估

如在本文第一章中介绍，订单的优先级由随机函数随机生成，每次运行都会产生不同优先级的订单，针对订单进行遗传迭代。

本文根据同一配送中心距离其辖区内卸货点的距离之和、无人机的续航里程、订单的优先级三个因素，估算需要出动的无人机架次。对于同一配送中心，**优先级高**的卸货点将会优先配送，相同优先级下，优先配送**距离最近**的卸货点。对于超出无人机往返里程的辖区，根据其辖区内订单的**累计距离**，派出最少的无人机架次。

对于每架无人机的实际配送路径，计算其到达每个卸货点的运输时间，判断是否满足所有卸货点的**时间窗约束**。若不满足，转至遗传算法部分继续迭代；若满足约束，则输出适应度值最佳的个体，对应的配送中心坐标和辖区内的卸货点，并绘制相应的结果图。

该部分代码如图 3.6，图 3.7 所示。

```

%% 得到最优个体
[fit_final,distance_02,V0_fit, ind1,C2C] =ga_fitness(bb0,popsize,Parents,a1,cnum); %调用适应度值函数

[min_p,pos] = min(fit_final);
V0=V0_fit; %聚类中心的坐标
best_ind=ind1;
%计算中心坐标更新后的距离
bb01=[bb0(:,1:3),best_ind];
for ii=1 : dim %dim
    for j=1:size(V0,1) % cnum
        if bb01(ii,4)==j
            distance_fit(ii,:)= calculateDistance(bb01(ii,2), bb01(ii,3), V0(j,1), V0(j,2));
        else
            continue
        end
    end
end
end
Trans_time_fit = distance_fit./60*60; %计算运输时间 运输速度60km/h 再转化为分钟min

```

图 3.6 得到最优个体

```

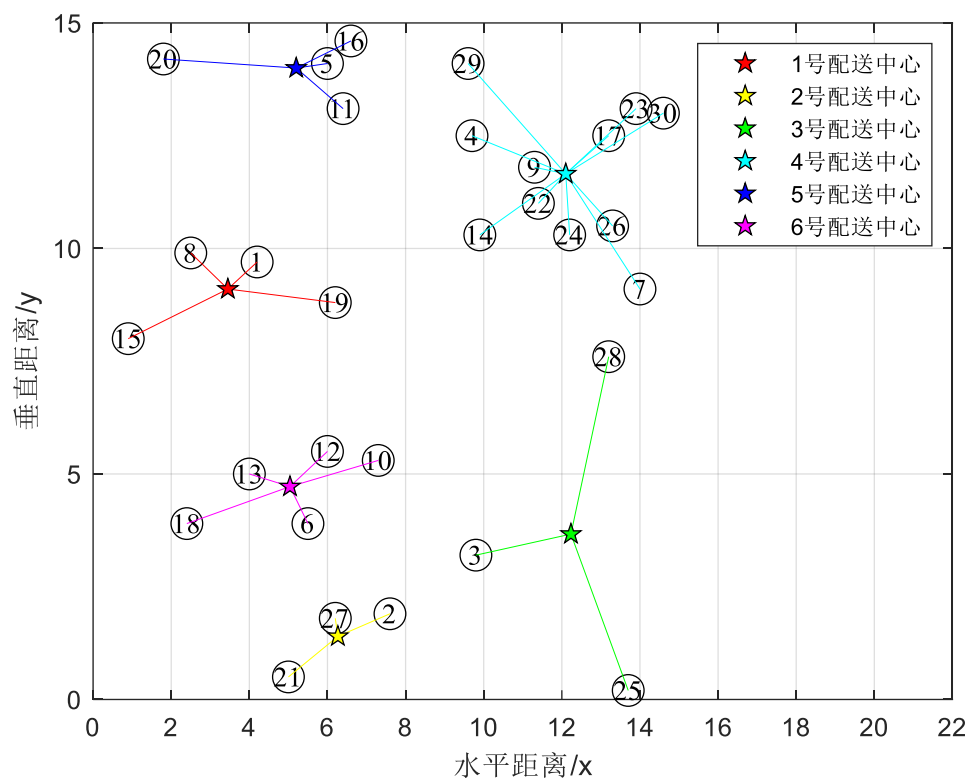
%% 判断配送中心服务的卸货点数量是否超出限制
num_limit = 15; % 一个配送中心 所能服务的卸货点数量上限
num_judge = bb2(:,[1,5]);
tmp=sortrows(num_judge,2);
[p1,p2]= unique(tmp(:, 2));
num_tmp = zeros(1,length(p1));
for i = 1 : length(p1)
    if i ~= length(p1)
        num_tmp(i)=p2(i+1)-p2(i);
    else
        num_tmp(i)=length(num_judge)+1-p2(i);
    end
    if num_tmp(i) > 15
        disp([num2str(i),'号配送中心的负荷不满足要求:',num2str(num_tmp(i))])
    else
        disp([num2str(i),'号配送中心的负荷满足要求'])
    end
end
end

```

图 3.7 判断时间窗负荷

4 实验结果与分析

4.1 聚类数据可视化



第 1 个配送中心辖区内的卸货点有: 1 8 15 19

第 2 个配送中心辖区内的卸货点有: 2 21 27

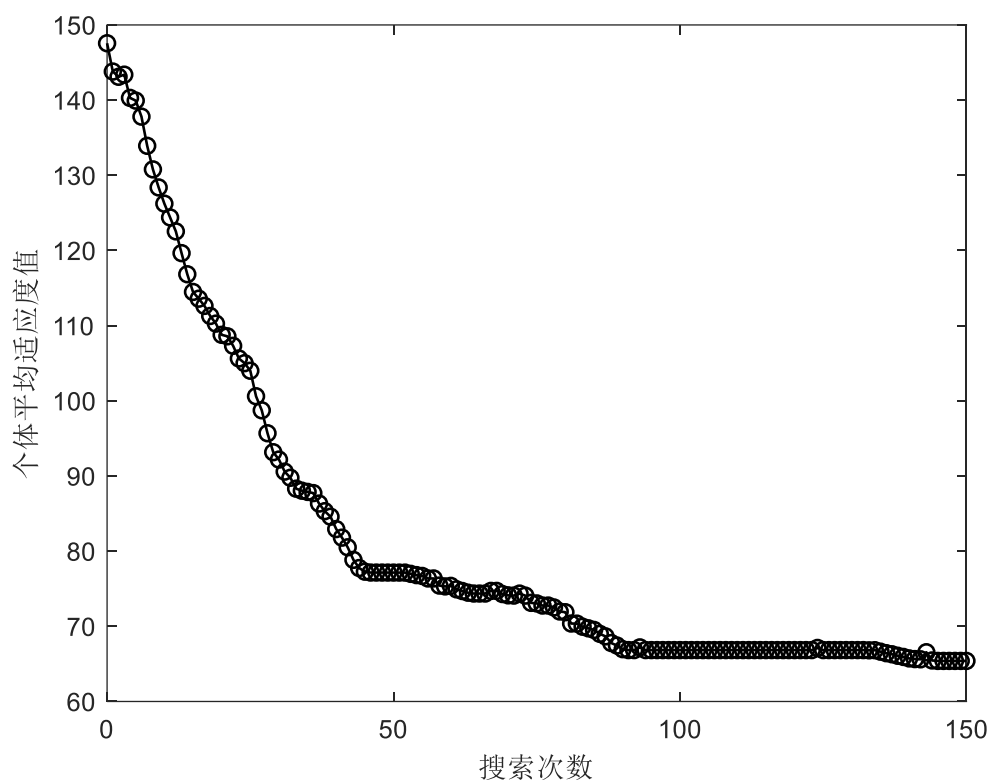
第 3 个配送中心辖区内的卸货点有: 3 25 28

第 4 个配送中心辖区内的卸货点有: 4 7 9 14 17 22 23 24 26
29 30

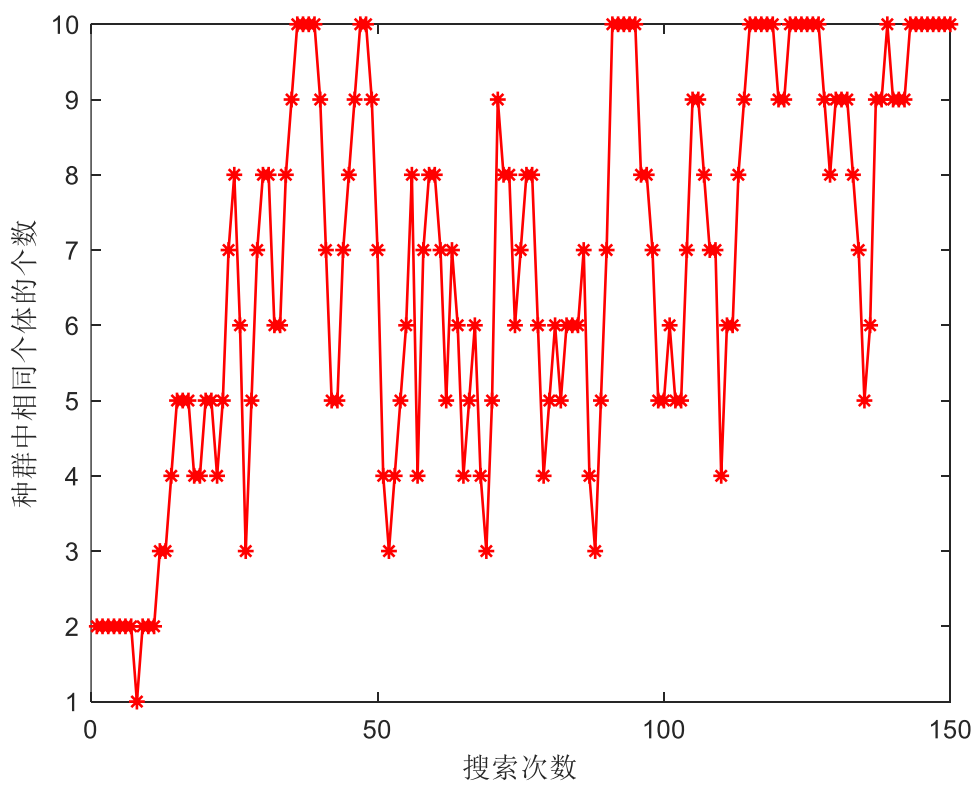
第 5 个配送中心辖区内的卸货点有: 5 11 16 20

第 6 个配送中心辖区内的卸货点有: 12 13 18 10 6

4.2 适应度函数



可以看到，随着迭代次数 100 次后，得到的适应度个体变化差异不明显，说明算法已收敛。

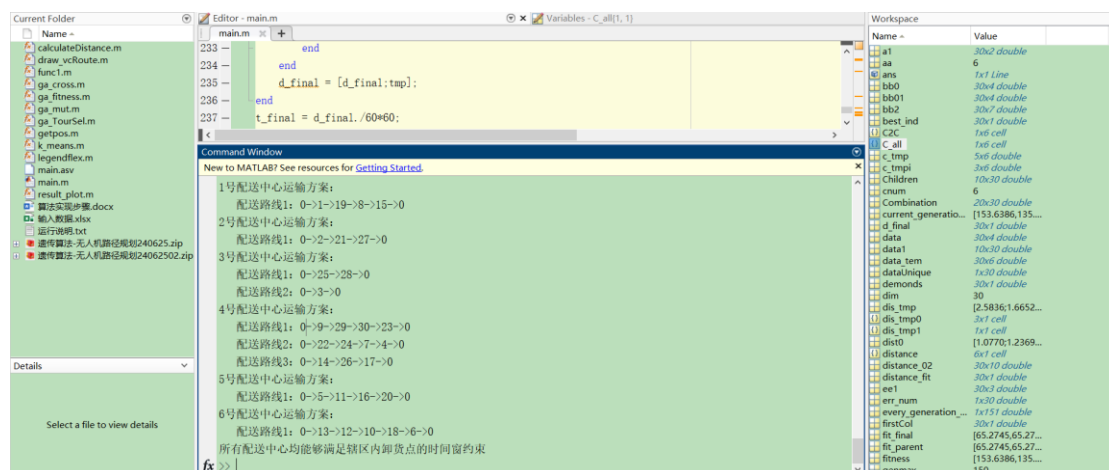


上图说明了种群中相同个体的数量变化，随着搜索次数的增加，相同个体数量保持稳定，同样说明该遗传算法已收敛。

4.3 配送路径规划

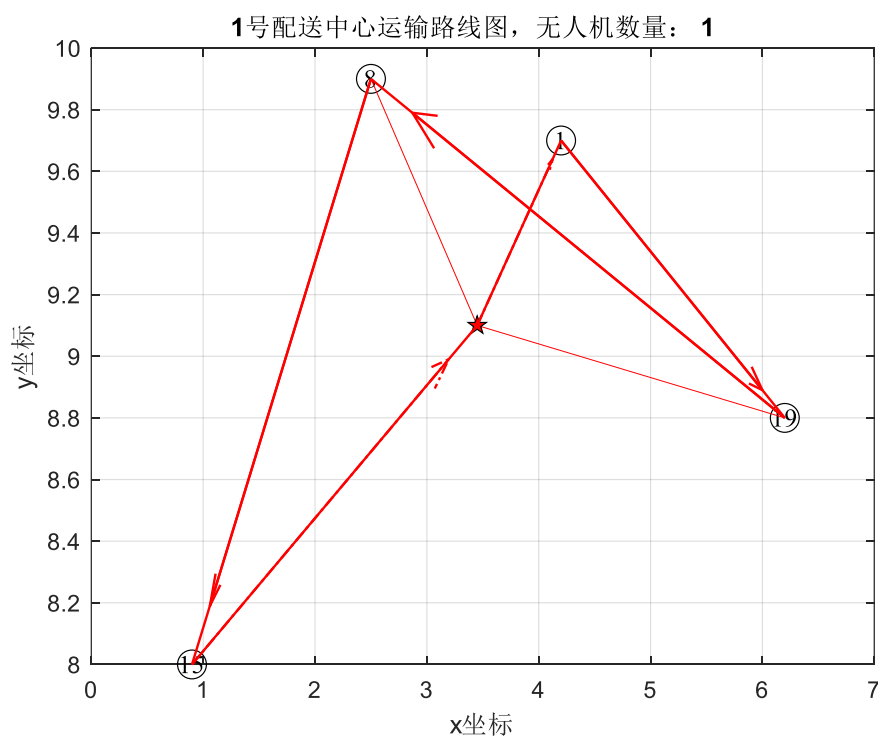
综合上述算法设计，本文考虑优先级约束与时间窗约束，设计了遗传算法得到的无人机路径规划方案与可视化如下，图中包括了每次决策时出动的无人机架数、无人机最优路径规划结果。

运行结果如下：



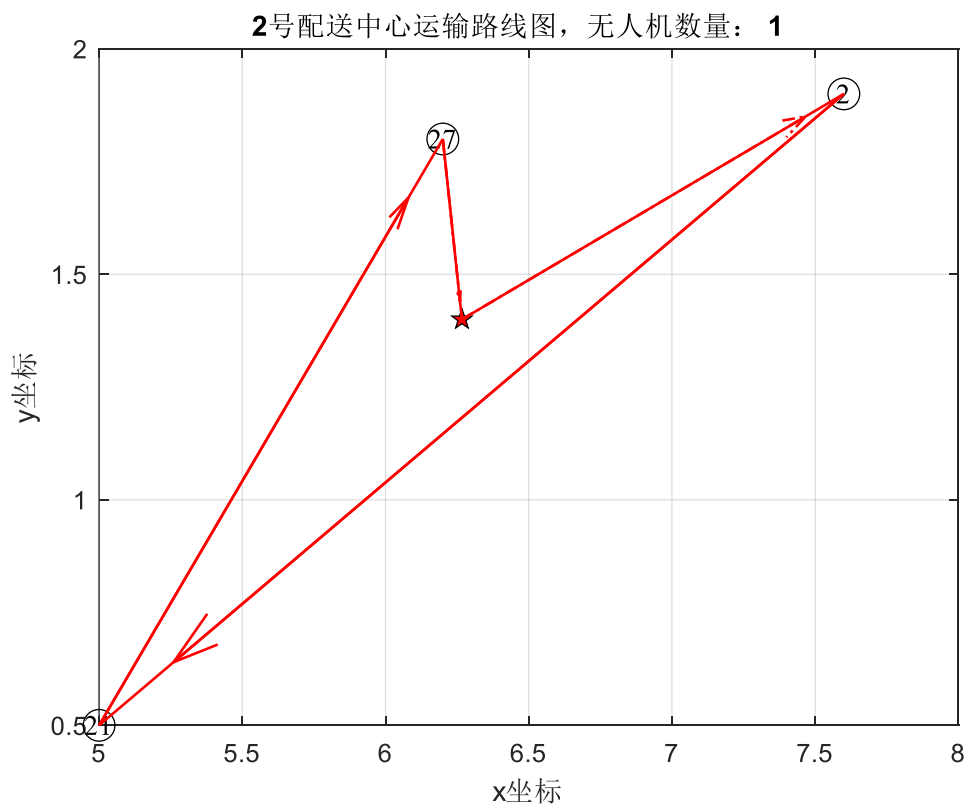
- 1号配送中心运输方案：

配送路线 1：0->1->19->8->15->0



- 2号配送中心运输方案：

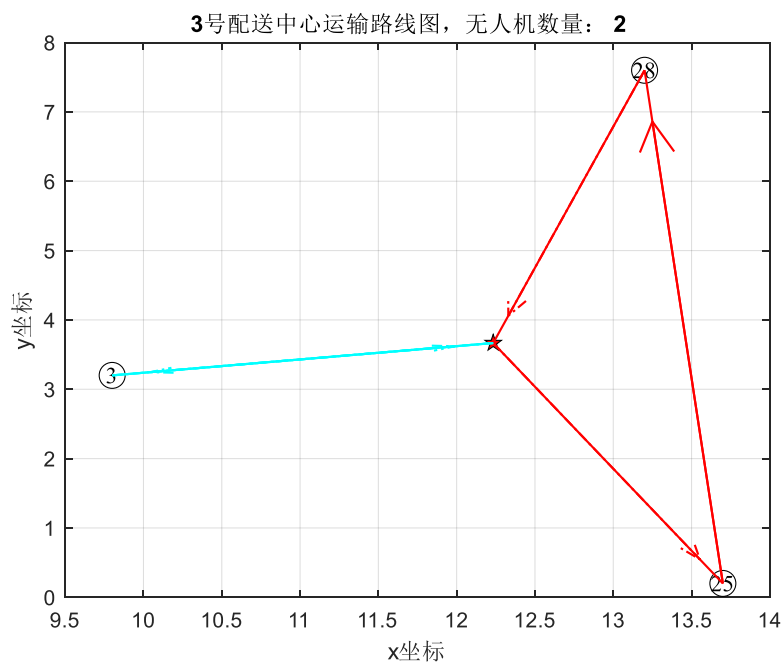
配送路线 1：0->2->21->27->0



• 3号配送中心运输方案：

配送路线 1：0->25->28->0

配送路线 2：0->3->0

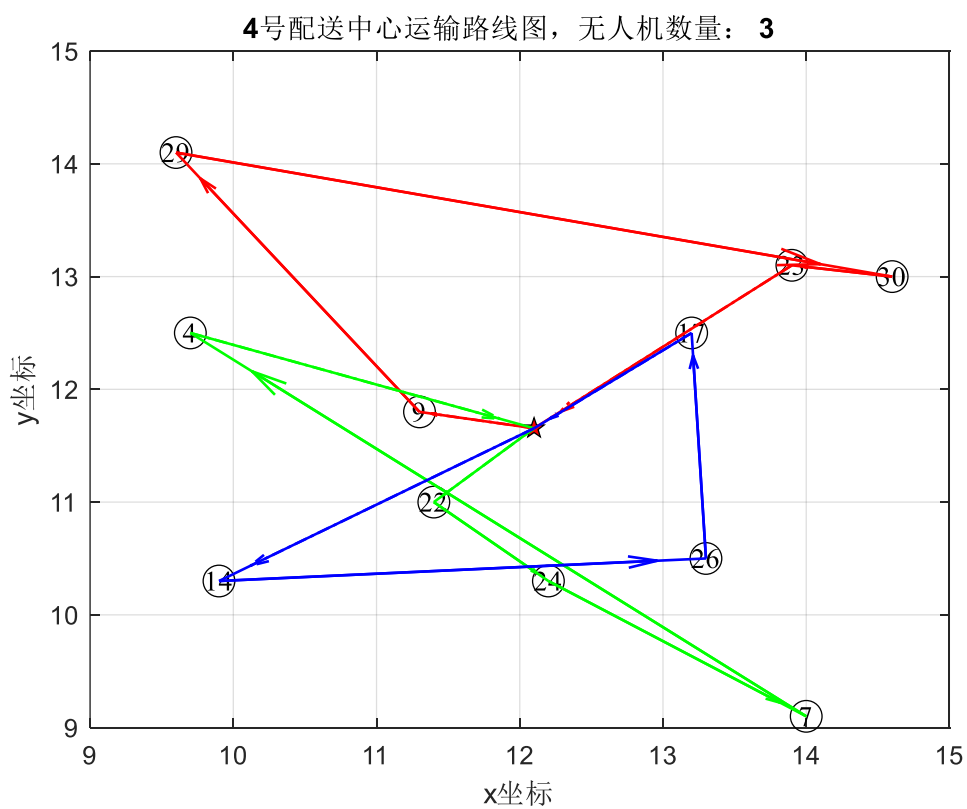


- 4号配送中心运输方案:

配送路线 1: 0->9->29->30->23->0

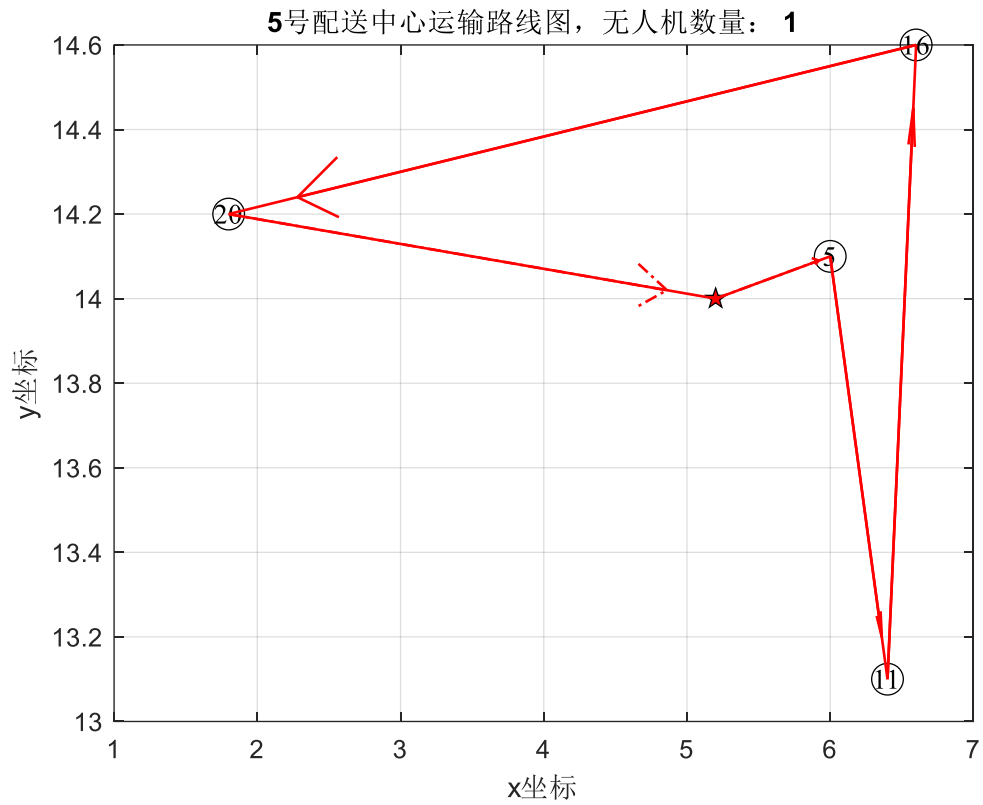
配送路线 2: 0->22->24->7->4->0

配送路线 3: 0->14->26->17->0



- 5号配送中心运输方案:

配送路线 1: 0->5->11->16->20->0



• 6号配送中心运输方案：

配送路线 1：0->13->12->10->18->6->0

