智能机器人大作业

学生姓名：吴程锴

学　　号：18029100040

班　　级：1802015

授课教师：于　昕

目录

一、研究背景

路径规划是机器人领域最重要的研究课题之一，他相当于机器人的大脑，只有知道了怎么走最好，然后才能控制无人机等机器人沿着规划好的路径运动。

二、国内外研究现状与发展趋势

路径规划算法包括基于搜索和基于采样的规划算法。

基于搜索的路径规划算法已经较为成熟且得到了广泛应用，常常被用于游戏中人物和移动机器人的路径规划。其中包括最佳路径优先搜索算法、Dijkstra 算法、A *搜索算法、实时学习 A * 搜索（LRTA*）算法、实时适应性 A * 搜索（RTAA*）算法、动态 A * 搜索（D*）算法等。

与基于搜索不同，基于采样的路径规划算法不需要显式构建整个配置空间和边界，并且在高维度的规划问题中得到广泛应用。其中包括快速随机搜索树（RRT）算法、目标偏好 RRT 算法、双向快速扩展随机树（RRT_CONNECT）算法、动态 RRT 算法、快速行进树（FMT*）算法、Batch Informed 树（BIT*）算法等。

三、研究方案

本文主要研究 Weighted A*算法在不同权值下的表现。

3.1 地图

本文采用二维的占用的栅格地图，如图 1 所示。

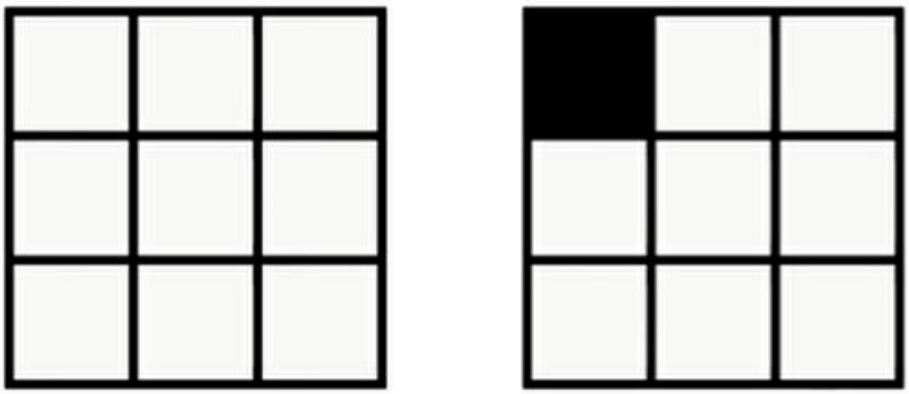

图 1 二维的占用的栅格地图

3.2 节点间连接方法

本文采用八连接的方法，如图 2 所示。

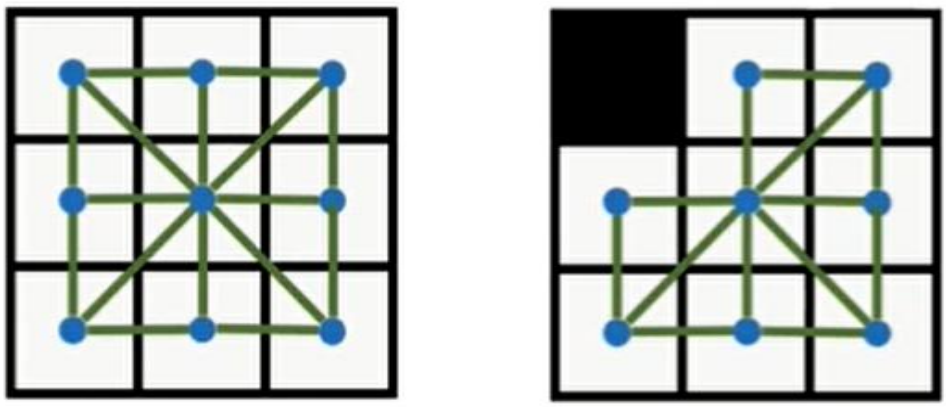

图 2 八连接

3.3 距离定义

本文采用欧式距离作为各点间的距离，各点的位置坐标即为矩阵的索引。

3.4 Weighted A*算法

Weighted A*是在 A*算法的基础上发展而来的。先介绍 A*算法。

3.4.1 A*算法

A*算法基于贪婪算法，从起点出发，逐步找到到达终点的最短距离。由于我使用了八连接的方法，所以使用简单的曼哈顿距离作为启发式函数可能会使结果不是最优解。为了使启发式函数 admissible，即估计的节点到终点的距离要小于实际的距离，所以我采用欧式距离作为启发式函数。A*算法的流程图如图 3 所示。

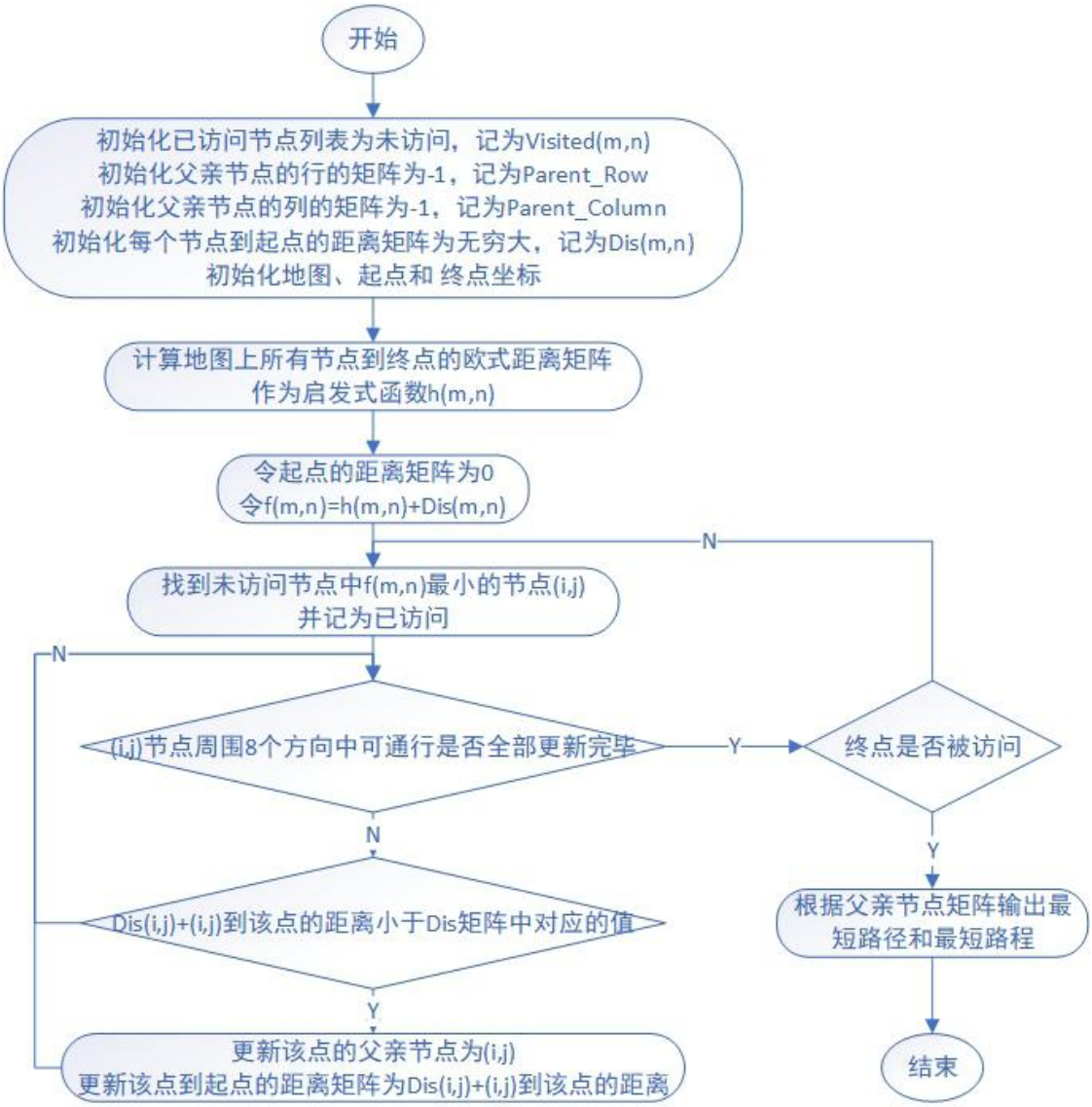

图 3 A*算法流程图

3.4.2 Weighted A*算法

在 A*算法的基础上，修改选择下一个访问节点的依据函数。增加一个参数 $w \in [0,\infty)$，新的函数为

$$f(m,n) = h(m,n) + w \times Dis(m,n)$$

算法可以通过调整 w 的取值来调节算法的贪婪性。当 $w=0$，Weighted A*算法退化为 Dijkstra 算法；当 $w=1$，即为 A*算法；当 $w \to \infty$，Weighted A*算法退化为贪心算法

四、仿真结果及分析

本文使用 MATLAB 进行仿真。代码见附录。

4.1 地图构建

在程序中，把地图抽象成一个矩阵，矩阵的值为 0 或 1，0 代表未占用，1 代表占用。创建一个简单的地图矩阵，使用 MATLAB 中的 *pcolor* 函数进行可视化如图 4 所示，其中黄色表示被占用，蓝色表示未占用。

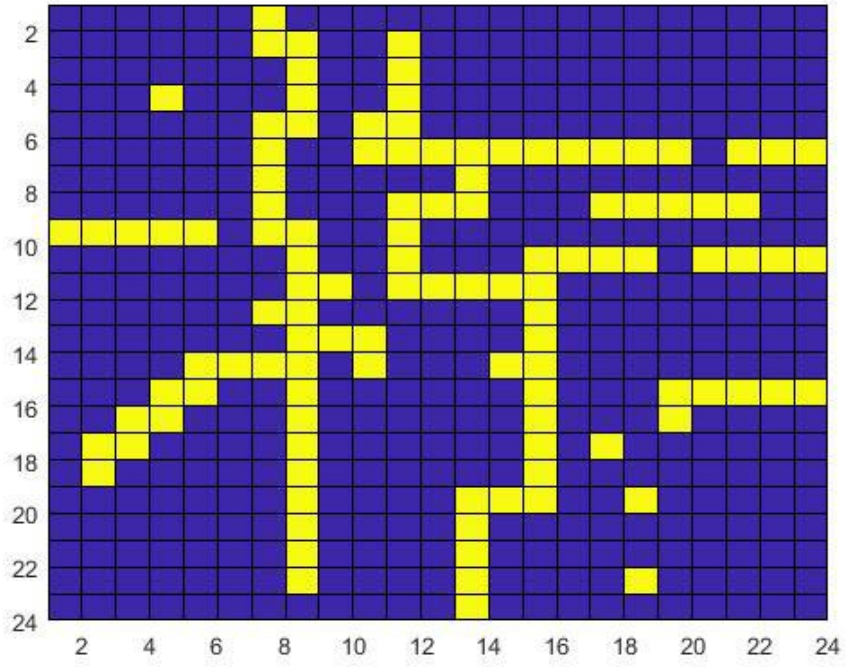

图 4 栅格地图

4.2 起点和终点选取

选取地图左上角和右下角作为起点和终点，如图 5 中绿色点所示。

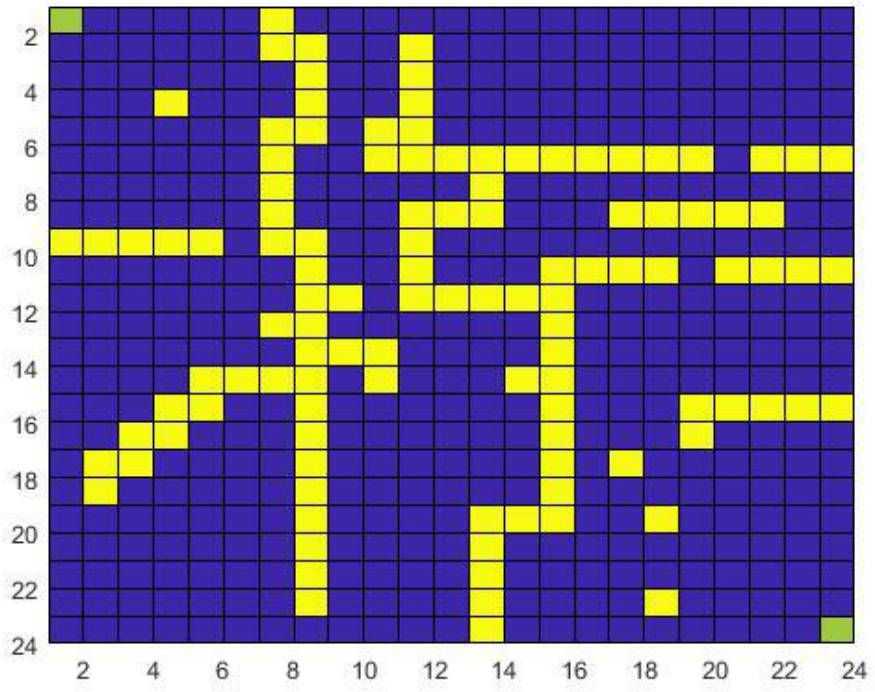

图 5 起点终点

4.3 Weighted A*算法仿真分析

4.3.1 设置权重 $w=1$

设置权重参数 $w=1$，算法即退化为 A*算法。运行程序，得到最短路程为 90.0833，最短路径如图 6 中淡蓝色所示。用时 0.0010 秒。

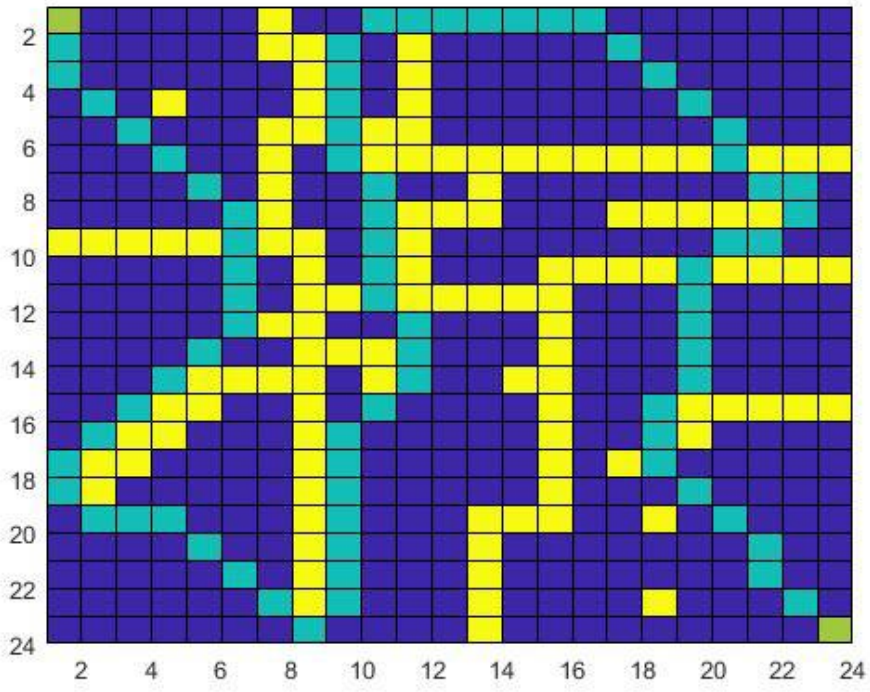

图 6 A*算法

4.3.2 设置权重 $w = 0.001$

设置权重参数 $w = 0.001$，算法退化为 Dijkstra 算法。运行程序，得到最短路程为 90.08，最短路径如图 7 中淡蓝色所示。用时 0.0020 秒。

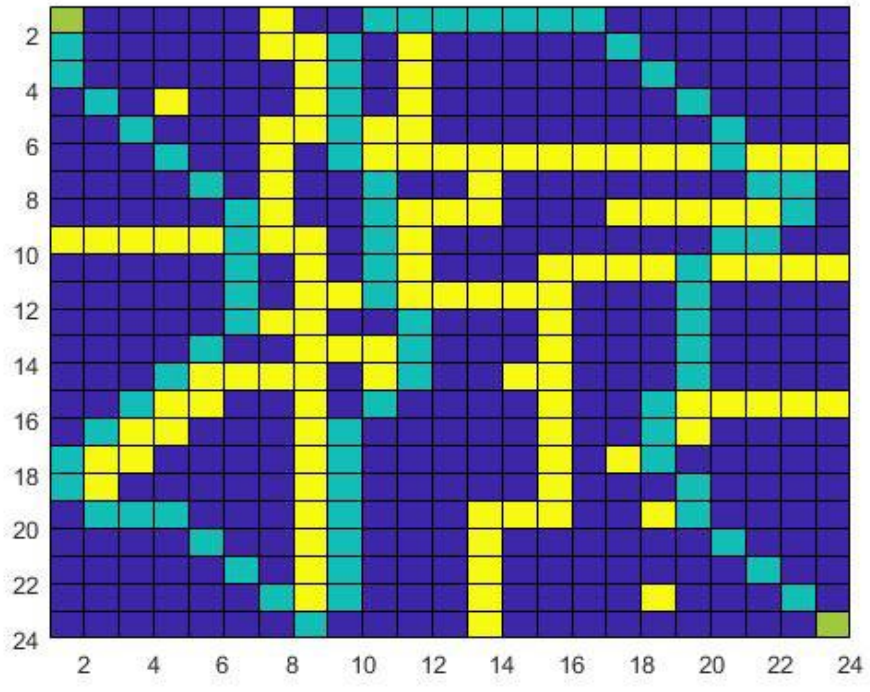

图 7Dijkstra 算法

4.3.3 设置权重 $w=100$

设置权重参数 $w=100$，算法退化为贪心算法。运行程序，得到最短路程为 95.9828，最短路径如图 7 中淡蓝色所示。用时 0.0010 秒。

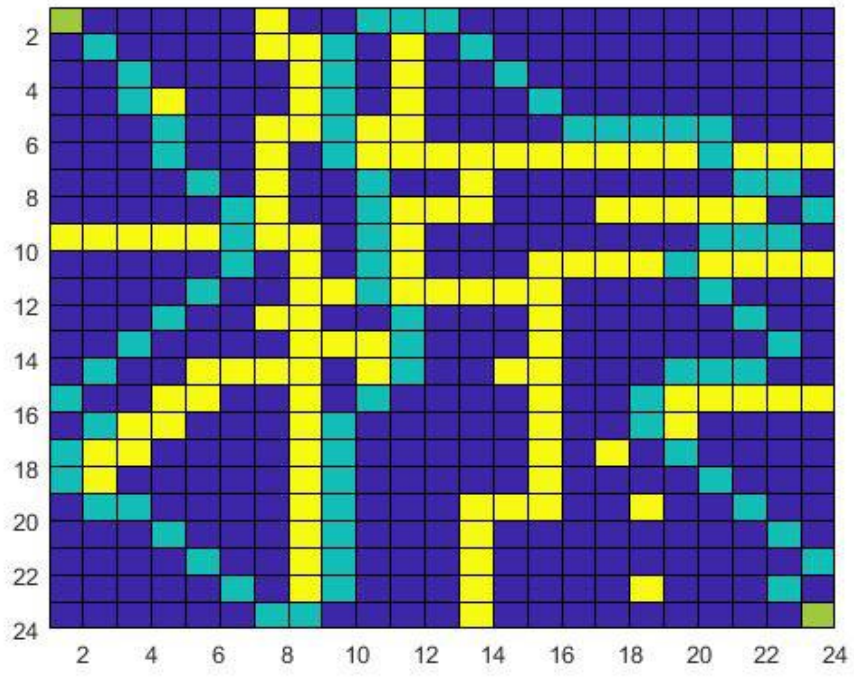

图 8 贪心算法

4.3.4 遍历权重

通过上面的分析，我们可以发现，权重越小，找到的最短路径越短，但是运行时间却长，接下来进行进一步的分析。把权重设置为从 0.001 到 30 步长为 0.01 遍历，分析在不同权重下运行时间和最短路程的关系。仿真结果如图 9 与图 10 所示

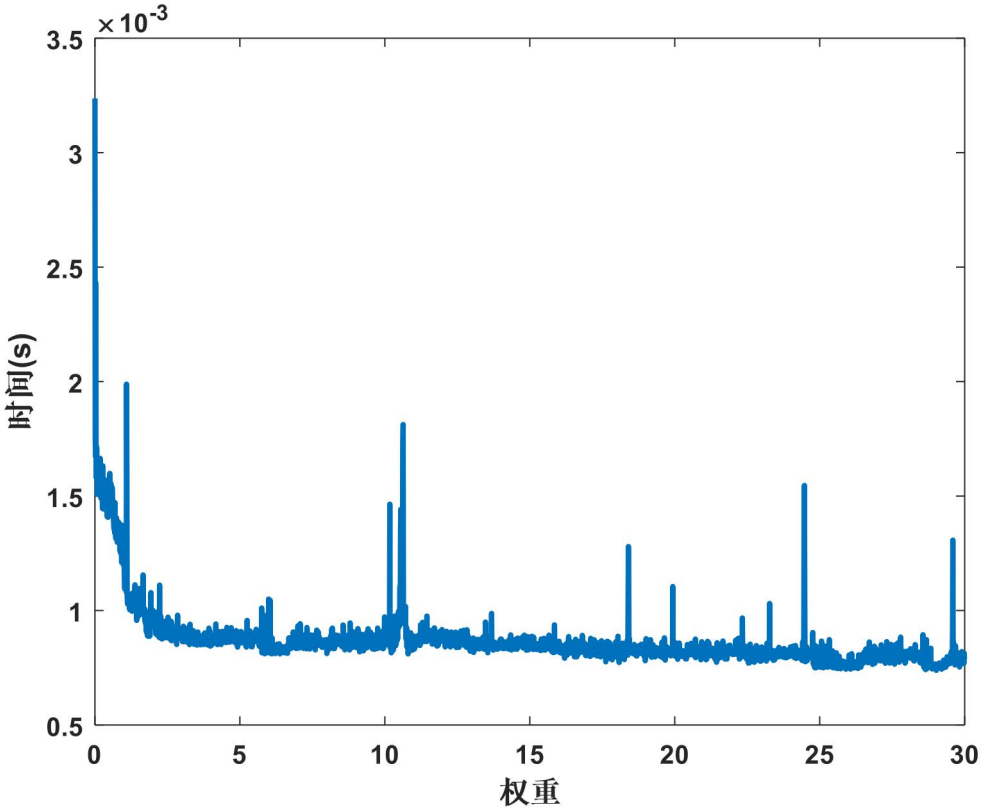

图 9 权重与运行时间的关系

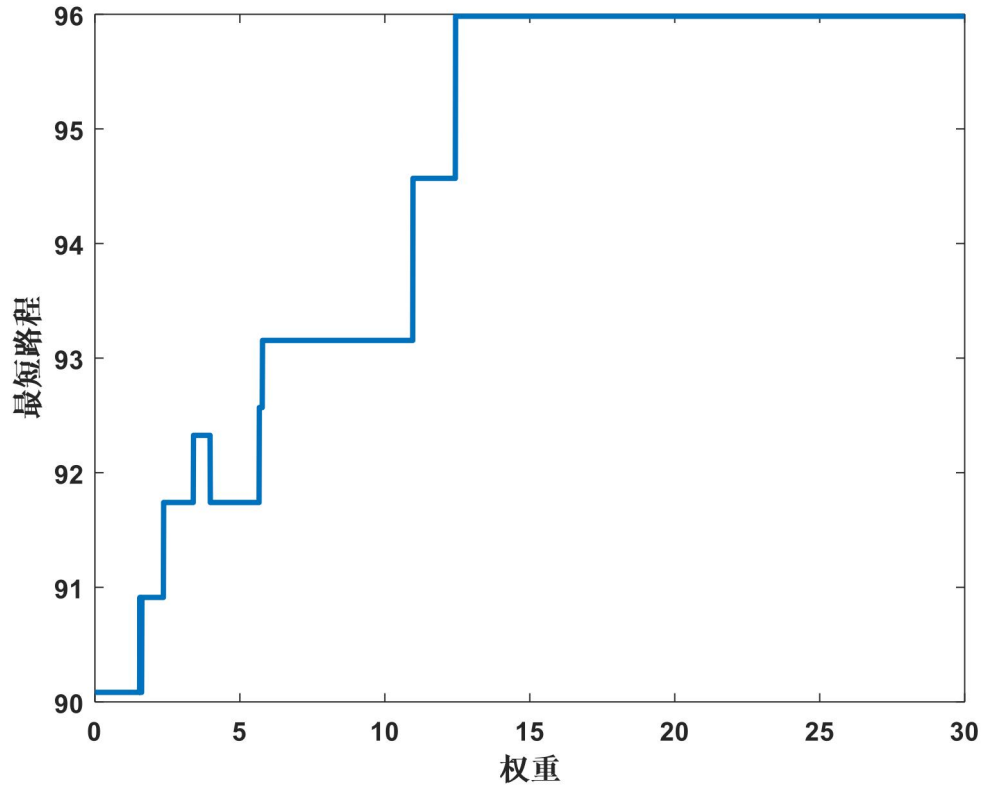

图 10 权重与最短路程的关系

通过仿真的结果可知，权重越小找到的最短路径越好，但计算机运行的时间就越长。当权重在 2~3 左右后，计算机运行时间就基本不变了，此时规划出的路径也足够优秀，所以权重的选择可以选择 2~3。

在实际使用中，可以根据计算路径要求的实时性来确定权重。例如在密集建筑物内快速飞行的无人机，在实时建图后需要极快的路径规划以使无人机不会装上突然出现的障碍物，这时只需要把权值调大即可。

五、总结

本文比较了 Weighted A*算法在不同权值下的表现，当权值越小，得到的最短路径越好，但计算机的开销越大，实际使用中根据实际情况选取适合的权重即可。

六、附录

附录一：A*算法函数

```
function [route,dis,RouteMap,RunTime]=AStar(map,NodeStart,NodeEnd,weight)
%% 初始化
dis=0;
Parent_m=-ones(size(map,1),size(map,1));
Parent_n=-ones(size(map,1),size(map,1));
Parent_m(NodeStart(1),NodeStart(2))=NodeStart(1);
Parent_n(NodeStart(1),NodeStart(2))=NodeStart(2);
Distance=1./zeros(size(map,1),size(map,1));
Distance(NodeStart(1),NodeStart(2))=0;
Visited=ones(size(map,1),size(map,1));
DisPre=map;
for i=1:size(map,1)
    for j=1:size(map,1)
        if(DisPre(i,j)==1)
            DisPre(i,j)=inf;
        else
            DisPre(i,j)=sqrt(abs(NodeEnd(1)-i)^2+abs(NodeEnd(2)-j)^2);
        end
    end
end
DisAStar=DisPre+Distance;

%% A*
t1=clock;
tic
while(Visited(NodeEnd(1),NodeEnd(2))~=9999)
    temp=Visited.*DisAStar;
    [m,n]=find(temp==min(min(temp)));
    m=m(1);
    n=n(1);
    Visited(m,n)=9999;
    %更新八个方向
    if(m-1~=0 && map(m-1,n)==0)
        if(Distance(m-1,n)>Distance(m,n)+1)
            Distance(m-1,n)=Distance(m,n)+1;
            Parent_m(m-1,n)=m;
            Parent_n(m-1,n)=n;
        end
    end
    if(n-1~=0 && map(m,n-1)==0)
```

```
        if(Distance(m,n-1)>Distance(m,n)+1)
            Distance(m,n-1)=Distance(m,n)+1;
            Parent_m(m,n-1)=m;
            Parent_n(m,n-1)=n;
        end
    end
    if(m+1~=size(map,1)+1 && map(m+1,n)==0)
        if(Distance(m+1,n)>Distance(m,n)+1)
            Distance(m+1,n)=Distance(m,n)+1;
            Parent_m(m+1,n)=m;
            Parent_n(m+1,n)=n;
        end
    end
    if(n+1~=size(map,2)+1 && map(m,n+1)==0)
        if(Distance(m,n+1)>Distance(m,n)+1)
            Distance(m,n+1)=Distance(m,n)+1;
            Parent_m(m,n+1)=m;
            Parent_n(m,n+1)=n;
        end
    end
    %斜向
    if(m-1~=0 && n-1~=0 && map(m-1,n-1)==0)
        if(Distance(m-1,n-1)>Distance(m,n)+1)
            Distance(m-1,n-1)=Distance(m,n)+sqrt(2);
            Parent_m(m-1,n-1)=m;
            Parent_n(m-1,n-1)=n;
        end
    end
    if(m+1~=size(map,1)+1 && n-1~=0 && map(m+1,n-1)==0)
        if(Distance(m+1,n-1)>Distance(m,n)+1)
            Distance(m+1,n-1)=Distance(m,n)+sqrt(2);
            Parent_m(m+1,n-1)=m;
            Parent_n(m+1,n-1)=n;
        end
    end
    if(m+1~=size(map,1)+1 && n+1~=size(map,2)+1 && map(m+1,n)==0)
        if(Distance(m+1,n+1)>Distance(m,n)+1)
            Distance(m+1,n+1)=Distance(m,n)+sqrt(2);
            Parent_m(m+1,n+1)=m;
            Parent_n(m+1,n+1)=n;
        end
    end
    if(m-1~=0 && n+1~=size(map,2)+1 && map(m,n+1)==0)
        if(Distance(m-1,n+1)>Distance(m,n)+1)
            Distance(m-1,n+1)=Distance(m,n)+sqrt(2);
```

```
            Parent_m(m-1,n+1)=m;
            Parent_n(m-1,n+1)=n;
        end
    end
    DisAStar=weight*DisPre+Distance;
end
t2=clock;
t=toc;
RunTime=etime(t2,t1);
RunTime=t;
m=NodeEnd(1);
n=NodeEnd(2);
i=1;
while(m~=NodeStart(1) || n~=NodeStart(2))
    route(1,i)=m;
    route(2,i)=n;
    i=i+1;
    map(m,n)=0.5;
    tempm=Parent_m(m,n);
    tempn=Parent_n(m,n);
    dis=dis+sqrt((tempm-m)^2+(tempn-n)^2);
    m=tempm;
    n=tempn;

end
map(NodeStart(1),NodeStart(2))=0.7;
map(NodeEnd(1),NodeEnd(2))=0.7;
RouteMap=map;
```

附录二：主程序

```
clc,clear
close all
%%
flag=1;
MapSize=24;
BlockNum=10;
weight=100;
if flag==1
    %自定义地图
    NodeStart=[1,1];
    NodeEnd=[23,23];
    map =[

```

```
14.    0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0
    0  0  0  0
15.    0  0  0  0  0  0  1  1  0  0  1  0  0  0  0  0  0  0  0  0  0
    0  0  0  0
16.    0  0  0  0  0  0  0  1  0  0  1  0  0  0  0  0  0  0  0  0  0
    0  0  0  0
17.    0  0  0  1  0  0  0  1  0  0  1  0  0  0  0  0  0  0  0  0  0
    0  0  0  0
18.    0  0  0  0  0  0  1  1  0  1  1  0  0  0  0  0  0  0  0  0  0
    0  0  0  0
19.    0  0  0  0  0  0  1  0  0  1  1  1  1  1  1  1  1  1  1  1  0
    1  1  1  0
20.    0  0  0  0  0  0  1  0  0  0  0  0  1  0  0  0  0  0  0  0  0
    0  0  0  0
21.    0  0  0  0  0  0  1  0  0  0  1  1  1  0  0  0  1  1  1  1
    1  0  0  0
22.    1  1  1  1  1  0  1  1  0  0  1  0  0  0  0  0  0  0  0  0  0
    0  0  0  0
23.    0  0  0  0  0  0  0  1  0  0  1  0  0  0  1  1  1  1  0  1
    1  1  1  1
24.    0  0  0  0  0  0  0  1  1  0  1  1  1  1  1  0  0  0  0  0
    0  0  0  0
25.    0  0  0  0  0  0  1  1  0  0  0  0  0  0  1  0  0  0  0  0
    0  0  0  0
26.    0  0  0  0  0  0  0  1  1  1  0  0  0  0  1  0  0  0  0  0
    0  0  0  0
27.    0  0  0  0  1  1  1  1  0  1  0  0  0  1  1  0  0  0  0  0
    0  0  0  0
28.    0  0  0  1  1  0  0  1  0  0  0  0  0  0  1  0  0  0  1  1
    1  1  1  1
29.    0  0  1  1  0  0  0  1  0  0  0  0  0  0  1  0  0  0  1  0
    0  0  0  0
30.    0  1  1  0  0  0  0  1  0  0  0  0  0  0  1  0  1  0  0  0
    0  0  0  0
31.    0  1  0  0  0  0  0  1  0  0  0  0  0  0  1  0  0  0  0  0
    0  0  0  0
32.    0  0  0  0  0  0  0  1  0  0  0  0  1  1  1  0  0  1  0  0
    0  0  0  0
33.    0  0  0  0  0  0  0  1  0  0  0  0  1  0  0  0  0  0  0  0
    0  0  0  0
34.    0  0  0  0  0  0  0  1  0  0  0  0  1  0  0  0  0  0  0  0
    0  0  0  0
35.    0  0  0  0  0  0  0  1  0  0  0  0  1  0  0  0  0  1  0  0
    0  0  0  0
```

```
    0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0
  0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0
  0   0   0   0
    ];
else
    %随机地图
    map=zeros(MapSize,MapSize);
    MapSize=MapSize-1;
    NodeStart=[1,1];
    NodeEnd=[MapSize,MapSize];
%     NodeStart=[ceil(rand*MapSize),ceil(rand*MapSize)];
%     NodeEnd=[ceil(rand*MapSize),ceil(rand*MapSize)];
    while(NodeStart(1)==NodeEnd(1) && NodeStart(2)==NodeEnd(2))
        NodeStart=[ceil(rand*MapSize),ceil(rand*MapSize)];
        NodeEnd=[ceil(rand*MapSize),ceil(rand*MapSize)];
    end
    Block=ceil(rand(2,BlockNum)*MapSize);
    for i=1:BlockNum
        map(Block(1,i),Block(2,i))=1;
    end
    map(NodeStart(1),NodeStart(2))=0;
    map(NodeEnd(1),NodeEnd(2))=0;
end
figure()
pcolor(map)
axis ij
%%
[route,dis,RouteMap,RunTime]=AStar(map,NodeStart,NodeEnd,weight);
figure()
pcolor(RouteMap)
axis ij
%% 权重对总路程和运行时间的影响
AllWeight=0.001:0.01:30;
RunTime=zeros(1,length(AllWeight));
Dis=RunTime;
for i=1:length(AllWeight)
    disp(num2str(AllWeight(i)))
    [route,Dis(i),RouteMap,RunTime(i)]=AStar(map,NodeStart,NodeEnd,AllWeight(i));
end
figure()
plot(AllWeight,RunTime,'linewidth',2)
xlabel('权重');
ylabel('时间(s)');
set(gca,'FontWeight','bold','FontSize',10)
```

```
figure()
plot(AllWeight,Dis,'linewidth',2)
xlabel('权重');
ylabel('最短路程');
set(gca,'FontWeight','bold','FontSize',10)
```