Computer Science and Application 计算机科学与应用, 2022, 12(6), 1580-1586
Published Online June 2022 in Hans. http://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2022.126159

Hans汉斯

移动机器人全局路径规划仿真研究

徐　健，张骥祥，张　军，王曰浩

天津职业技术师范大学，天津

收稿日期：2022年5月16日；录用日期：2022年6月15日；发布日期：2022年6月22日

摘　要

本文将针对全局路径规划进行研究，选取两个最常用到的Dijkstra算法和A*算法进行讲解。首先进行情景假设，规定算法演示的规则，用数字大小表示距离的远近，引入字母A~H，逆时针方向代表目标点的八个方向。然后依据算法的实现原理，判断下一步中心点的选择，Dijkstra算法以到起始点的代价最小为准，而A*算法以起始点到终点的估算路程最短为判断准则。最后，遍历到终点，通过提取父节点信息，得到最短路径信息。仿真实验结果表明，能有效验证算法的实现过程，对比出两种算法的异同点。

关键词

全局路径规划，Dijkstra算法，A*算法

Simulation Study of Global Path Planning for Mobile Robots

Jian Xu, Jixiang Zhang, Jun Zhang, Yuehao Wang

Tianjin University of Technology and Education, Tianjin

Received: May 16th, 2022; accepted: Jun. 15th, 2022; published: Jun. 22nd, 2022

Abstract

This paper will study global path planning, and select the two most commonly used Dijkstra algorithm and A* algorithm to explain. Firstly, scenario hypothesis is carried out and rules of algorithm demonstration are stipulated. Numerical size is used to represent the distance, letters A~H are introduced, and counterclockwise direction represents the eight directions of the target point. Then, according to the realization principle of the algorithm, the choice of the next center point is judged. Dijkstra algorithm takes the minimum cost to the starting point as the criterion, while A* algorithm takes the shortest estimated distance from the starting point to the end point as the criterion. Finally, the shortest path information is obtained by extracting the parent node informa-

文章引用：徐健，张骥祥，张军，王曰浩．移动机器人全局路径规划仿真研究[J]．计算机科学与应用，2022，12(6): 1580-1586. DOI: 10.12677/csa.2022.126159

tion through traversing to the end point. The simulation results show that the algorithm can be validated effectively and the similarities and differences of the two algorithms are compared.

Keywords

Global Path Planning, Dijkstra Algorithm, A* Algorithm

1. 引言

随着科技的发展，移动机器人在农业机械，智能交通还有智慧城市等领域都有不凡的表现，而机器人的路径规划是学术界研究的一个热点及难点问题。路径规划简单来说，就是设定起点与终点两个点，通过智能算法在可行路径上规避障碍物，获得一个安全无碰撞，距离最短的路径。常用的评价指标有规划路径的长短信息，用时大小以及拐角频率等[1] [2] [3]。另外，还可以从面对的场景，环境信息的不同，分为全局路径规划和局部路径规划[4]。全局路径规划主要使用的是静态地图，在先验信息已知的情况下规划路径，常见的算法有传统算法包括 Dijkstra 算法[5]、A*算法[6] [7]等，智能算法包含粒子群优化算法，蚁群算法，遗传算法等。而局部路径规划面对对象通常为动态环境，在环境完全未知或者部分已知的前提下，基于多传感器识别局部环境，形成实时路径规划，常用的算法有动态窗口法(DWA)，人工势场法(APF)，模糊逻辑法(FL)等。

文献[8]使用 A*算法解决了 AGV 无冲突路径规划问题。文献[9]将 A*算法应用到了农业场景中，提高了采摘机器人的工作效率和控制精准度。张超等人[10]运用 Dijkstra 算法思想，在输电线路选线方面发挥了作用。以上文献均涵盖了全局路径规划思想，涉及不同的领域，因此，对两种经典全局路径规划算法展开研究显得尤为重要。

2. 全局路径规划

全局路径规划是导航功能中重要的一环，导航包中常常出现，导航包调用全局路径规划器，并且是以插件的形式，插件接收全局地图，以及人为设定的起点和目标点等信息，经过算法处理，输出处理好的全局路径信息。

2.1. Dijkstra 算法

本文将通过复现 Dijkstra 算法的实现过程，详细地一步一步说明，来体会理解算法的核心思想，即 Dijkstra 算法将所有点的最短代价按照由近及远的顺序一一计算。在说明算法之前，需要进行情景假设，首先确定起始点，行走过程中，一次只能前进一格，不能够越过格子，所以可以得到周围八个相邻的格子。如果走上下左右四个方向的话，计它路程为 2。如果走左上、左下、右上、右下，那么就计它路程为 3。这里定义两个数组，一个是用来存放待确定路径的点，取名为 openlist = [ ]，另一个是用来存放已确定路径的点，取名为 closedlist = [ ]。说明一下，在 Python 里面是列表，如果在其他语言也可以设置为数组。根据命名规则，每个点通过下标信息都能记录其父节点，在提取路径时一目了然。不走已确定路径的点，即已放进 closedlist 的点。为了方便讲解，约定起点为 S 点，相邻的 8 个点以 A~H 来称呼表示

它的方向。第一步我们以 S 点为起点，把它放进 closedlist 里面，然后再计算 S 点的相邻 8 个点，那么由 S 点这个中心点出发，上下左右，这么直走，计它路程为 2，斜走为 3，然后结合刚刚的命名规则，左面这个点 2，它是在 S 点的 B 方向，所以是 $B_S = 2$，所以 B_S 这个点就代表了这个 2。然后下面也是同理，D_S 是 D 方向，斜走也是一样，A 方向是 3，斜走是 3，所以 $A_S = 3$，这么计算下来就把 8 个点都算出来，放进 openlist 里面。如图 1(a)所示。

第二步要进行中心点的选取，在 openlist 里面取一个最小值的点作为选取点，可以看到 openlist 里面一共有四个 2，这里先按上下左右的顺序去依次计算。首先取上面这个 2，它是 H_S 点，那么第二步计算就以 H_S 点为中心点来进行，在这个图上可以看到，在 H_S 点这个位置点了个小红点让它更醒目一些，然后在周围加一个小框框把它框柱，代表它周围 8 个相邻的点要进行计算，那么就把 H_S 点从 openlist 里面取出放进 closedlist 里面，然后再计算它相邻的 8 个点去累计路程。可以看到它周围的 8 个点有好几种情况，首先它下方这个点是 S 点，S 点作为起点已经被放进 closedlist 里面，那么这个点是不需要计算的。然后第二种情况是，它旁边的两个 3 和下方的两个 2 在上一步已经被计算过了，在这一步的时候就是存在这么一种情况，就是点已经在 openlist 里面，针对这种情况就要判断一下，选取路程较小的方案。举一个例子，我们先来看一下这个 3，这个 3 是怎么来的呢，是一开始从 S 点斜走一步取 3，在第二步从 H_S 点出发往左走一步也能到达这个点，但是这种方案的话就是 2 + 2 = 4，因为它走了两步，先从 S 点往上走一步，再往左走一步，那就是 2 + 2 = 4，4 > 3，那么就取第一种方案 3。那么右边这个 3 和下面的两个 2 都是同样的道理，它们原来的值更小那么就保留它们的值。那么第三种情况是 H_S 点的上方三个点本来是空白的，它没有在 openlist 里面，那么这里就要计算它的值，例如这个 4 就是 B_S 点往上走一步，就是 2 + 2 = 4，我们把这个 4 的点取名为 H_{HS}，这个命名规则是这样的，这个下标代表它的父节点 H_S，然后它本身的 H 就是代表它的方向，它在中心点的上方就是取 H，那么这个点就是 $H_{HS} = 4$。那么下面的 $A_{HS} = 5$ 也是一样的道理，在这个地方 H_S 这个中心点斜上走一步就是 5，这个 A_{HS}，然后再把这三个计算好的点放进 openlist 里面，那么到这里就可以认为，在 closedlist 里面的 H_S 这个点，它是最小路径，因为它每走一步都是正数，没有中转点使得 S 到 H_S 的路程缩短。这个结论在后面的每一步都是成立的，这样就能保证我们通过 Dijkstra 这种算法，算出来的最后的路径是一个最优解。如图 1(b)所示。

再来看一下第三步，继续在 openlist 里面去取一个最小的点，以 D_S 作为中心点进行第三步，首先把它从 openlist 里面提取出来，放到 closedlist 里面，然后再重复第二个步骤，计算它相邻的 8 个点，然后进行累计路程，对于点已存在 openlist，取路程小的方案。$D_{DS} = 4$ 表示从 S 点出发，第一个 D 就代表 D 方向走一步，然后第二个 D 就是再往下走一步，那么就得到了 2 + 2 = 4。$C_{DS} = 5$ 是从 S 点出发往 D 方向走一步，再往 C 方向走一步，是 2 + 3 = 5，最后一个点 E_{DS} 也是类似。如图 1(c)所示。

第四步，继续在 openlist 里面取一个最小的点作为第四步的中心点，这里就取左边这个 2，把这个 B_S 放进 closedlist 里面，继续重复上面的步骤计算它相邻的 8 个点，累计路程，然后得到三个新增加的点，把它们也放到 openlist 里面。如图 1(d)所示。

第五步也是一样的做法，取右边这个 F_S=2 为中心点，把中心点 F_S 放到 closedlist 里面，经过计算也增加了三个新的点。那么到这一步，就已经把 2 的值算完了，下面开始算 3 的值，因为 3 是最小的。如图 1(e)所示。

第六步，取 $A_S = 3$ 这个点作为中心点，重复跟前面一样的步骤，可以看到它相邻的 8 个点有 3 个点，这三个点已经标了红点，它们已经被放进 closedlist 里面，那么这三个点就不用计算，还有两个 4 和两个 5 已经存在 openlist 里面了，我们就可以对比它的路程，取路程小的一个方案。看一下这个 5，这个 5 原本是怎么来的呢，是从 S 点出发往左走一步等于 2，然后再往斜上走一步是 5，2 + 3 = 5。如果以 A_S 为中心，往左走一步，3 + 2 也是等于 5，那么在这种相等的情况下我们就可以把原来的方案替换也可以不替

换，其实这等于是两种相同路程的路径，一种是先往左走一步，再往斜上走一步，另一种是先往斜上走一步，再往左走一步，它们的距离都是 5，那么这里就不进行替换，取原来的那种方式，最后就只新加了一个 6，斜走两步，3 + 3 = 6，那么下面的步骤也是一样的。如图 1(f)所示。

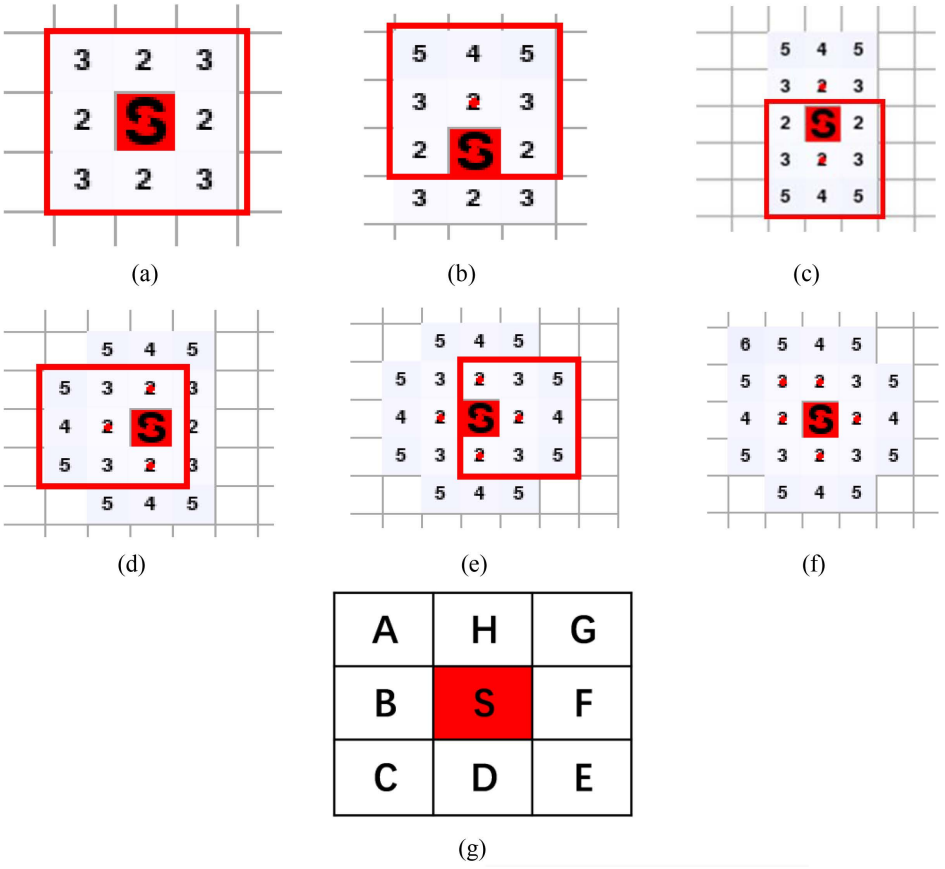


Figure 1. Dijkstra algorithm demonstration
图 1. Dijkstra 算法演示

2.2. A*算法

介绍 A*算法之前，先说明一下两种距离。第一个是欧几里得距离，也是经常用的一个概念，在二维的空间中，要算两个点之间的距离，可以直接用直线两两相连，计算公式如式(1)所示：

$$d(a,b)=\sqrt{(a_x-b_x)^2+(a_y-b_y)^2} \tag{1}$$

第二种距离是曼哈顿距离，这个名字的由来不是提出这个概念的人叫曼哈顿，而是他提出了一个在纽约曼哈顿的情景，建筑物就是障碍物，我们需要打车到目的地，不可能使用欧几里得距离，直接穿过障碍物，出租车只能沿着马路走，所以曼哈顿距离又称出租车距离，计算公式如式二所示：

$$d(a,b)=|a_x-b_x|+|a_y-b_y| \tag{2}$$

对比两种距离，欧几里得距离计算量将大大增加，但方向性很强，而曼哈顿距离计算量相对较小，但会存在很多距离等效的路线。

接下来，看一下 A*算法的关键，估算函数，它的公式如下所示：

$$F = G + H \tag{3}$$

式中，G 代表从起点到当前节点实际路程，H 代表从当前节点到终点最小估算路程，使用曼哈顿距离、或欧几里得距离，F 代表从起点到终点估算路程，下面将围绕这个核心公式，讲解 A*算法的实现过程。

情景假设与 Dijkstra 算法类似，不同的是算 H 估算距离时，忽略障碍物。首先第一步，以 S 点为中心点，根据估算函数来计算它周围的八个点，它们的父节点为 S，首先来看 G，G 的含义是起点到当前节点实际路程，周围的八个点，直走为 2，斜走为 3，跟 Dijkstra 算法相同，不同之处是还得加上一个估算路程。以 F_S 点为例，从起点左走为 2，即 G 值为 2，H 值采用曼哈顿距离，根据公式，通过数格子的方式计算估算距离，X 方向上，差三格，Y 方向上，差五格，一个格子计为 2，F 值即为 3*2 + 5*2，加上 G 的值 2，得到 F 值为 18。如图 2(a)所示。

G、H 和 F 值构成 openlist 和 closedlist 里的内容，其中最小的 F 值决定下一步的中心点选择，由此判断，E_S 为下一步的中心点。第二步，将 E_S 点从 openlist 里面取出，放进 closedlist，加个红点，表示已放入，加上红框，计算它周围的八个点。第一种情况为 S 点，已经在 closedlist 里面，不必计算。第二种情况，点在 openlist 里面，比较 G 值，取 G 值较小的结果。如图 2(b)所示。

第三步取 E_{ES} 点作为中心点，重复步骤，遇到障碍物或者已经存在 closedlist 的情况，直接略过，其余的点就按照 $F = G + H$ 来计算。如图 2(c)所示。

重复以上步骤，得到 F 值最小的点即 D_{EFEES}，通过父节点反推，从而获取路径信息。

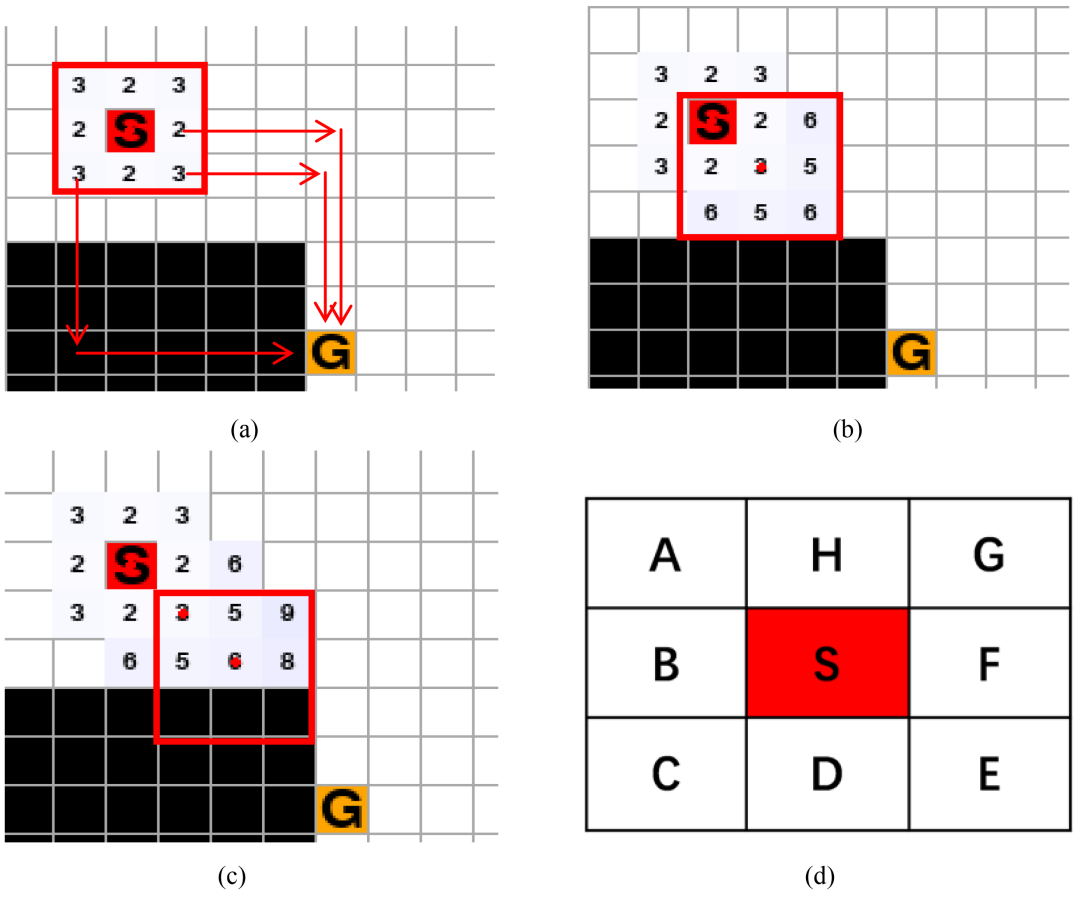
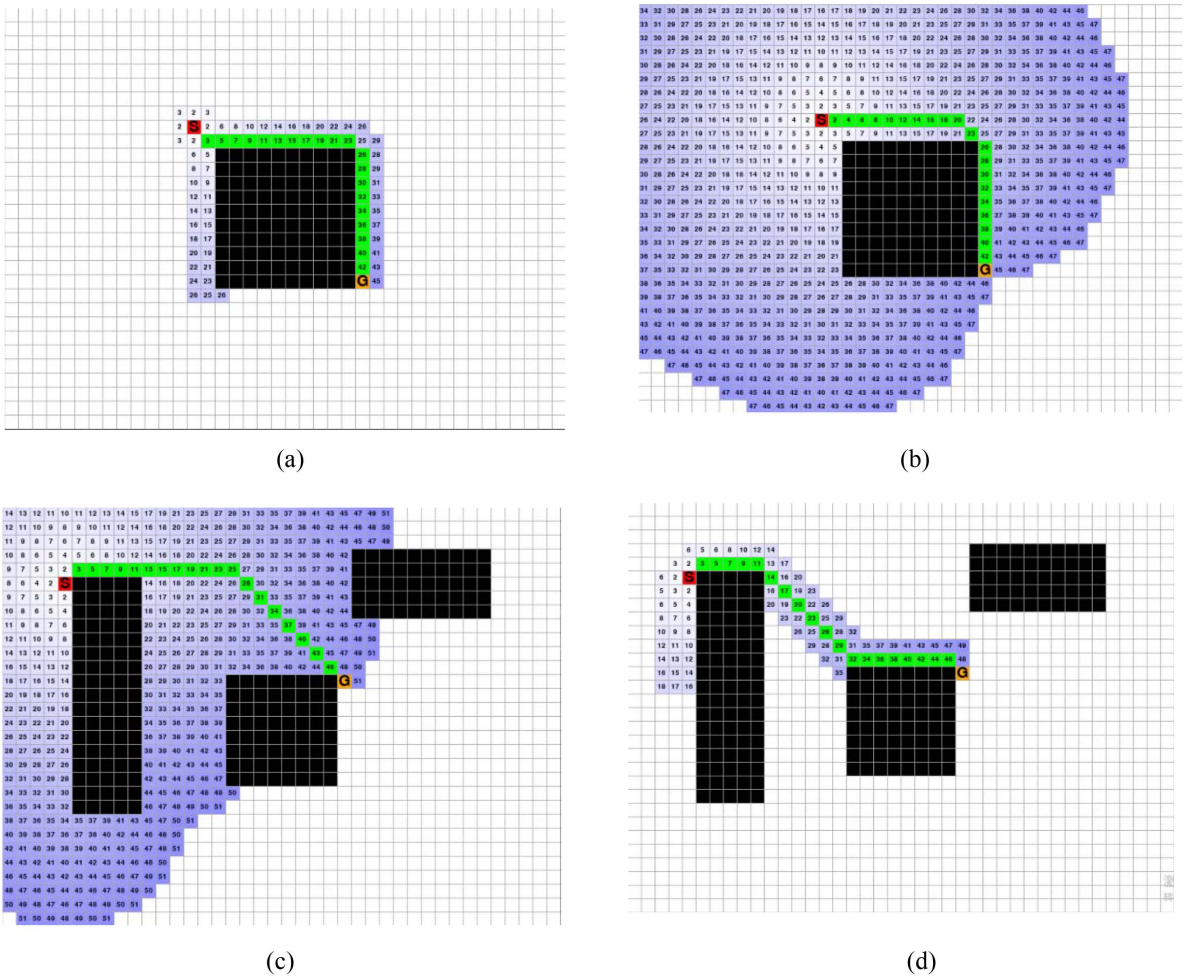


Figure 2. A* algorithm demonstration
图 2. A*算法演示

3. 对比仿真实验

本文采用栅格地图进行仿真，左上角为零点。在同一张地图上，设置相同的起点与终点，来对比两种算法的异同之处。



(a) (b) (c) (d)

Figure 3. The simulation results
图 3. 仿真实验

图 3(a)~(b)两图为简单地图上分别运行 A*算法和 Dijkstra 算法，图 3(c)~(d)两图为较复杂地图上分别运行 Dijkstra 算法和 A*算法。从对比仿真的结果入手，可以发现两种算法的异同点。如图所示，红色 S 点表示起点，终点用黄色 G 点表示，基于前文的介绍，从起点到终点的路径长短信息可以用数字的大小来表示，且为了方便呈现出来，通过加深蓝色来代表距离的增长，即距离越长颜色越深。观察两图可以发现，简单的图中，两种算法虽然路径方案不同，但路程长短相同，用数字表示均为 44。图 3(a)从起始点向 E 方向前进一格，计为 3，接着朝 F 方向前进，依次增加 2，直到来到代价为 23 的点，再次向 E 方向前进一格，加 3 变为 26，最后向着 D 方向前进，一格计为 2，来到终点 44。图 3(b)是先朝着 F 方向前进，以 2 为步长，路程代价增加为 20，再向 E 方向前进 2 格，步长为 3，来到之前的代价为 26 处，之后的步骤与图 3(a)相同。接着，将障碍物增多，地图复杂化，选取起点和终点，总代价均为 49，对比两图的区别，在代价数字为 11 处出现不同的行进路线，图 3(c)向 F 方向拓展，而图 3(d)向 E 方向前进，两图

分别在 25 和 32 处进行新的转折。

分析一下两种算法出现两种路径方案的现象，可以得到两个算法实现的差异之处。Dijkstra 算法是遍历中心点周围所有的点，选取代价最小的，而 A*算法是引入估算函数，具有方向性，结果就导致了选取方案的差别。A*算法由于存在方向性函数的指引，计算量大大减小，而 Dijkstra 算法搜索区域较大，花费时间较长。简单对比一下两者：Dijkstra 是一种基于广度优先的算法，操作过程如下，从起点开始由近到远去遍历所有的点，把起点到每一个点的路径都算出来，然后由近及远都算一遍，直到遇到了目标点，那么就把起点到目标点这个路径给提取出来，这个就是广度优先，特点是把每个点都算了一遍。而 A*算法是一种基于深度优先思想的算法，并没有从起点开始把每个点都算一遍，而是目标点对路径规划有一个方向性的指引作用，它会优先朝着目标点方向去计算，计算量将远远小于前者。另外一个特点是 Dijkstra 算法能保证路径一定是最短最优的路径，而 A*算法找的路径不一定是最优解。

4. 结语

通过对比 Dijkstra 算法与 A*算法的实现过程，加深了对全局路径规划的理解。本文用数字将路径代价量化，清晰地表示了距离信息，字母表示中心点周围的方向，按照算法原理选取中心点位置，直至遍历到目标点，读取父节点信息，得到路径方案。仿真结果表明，算法实现过程有效，能够展现两种算法的异同点。

参考文献

[1] Skou-Nielsen, N., Villa-Henriksen, A., Green, O. and Edwards, G.T.C. (2017) Creating a Statistically Representative set of Danish Agricultural Field Shapes to Robustly Test Route Planning Algorithms. *Advances in Animal Biosciences*, **8**, 615-619. https://doi.org/10.1017/s2040470017000188

[2] Freitas, H., Faial, B.S., Cardoso e Silva, A.V. and Ueyamal, J. (2020) Use of UAVs for an Efficient Capsule Distribution and Smart Path Planning for Biological Pest Control. *Computers and Electronics in Agriculture*, **173**, Article No. 105387. https://doi.org/10.1016/j.compag.2020.105387

[3] 徐博, 陈立平, 谭彧, 等. 多架次作业植保无人机最小能耗航迹规划算法研究[J]. 农业机械学报, 2015, 46(11): 36-42.

[4] 张漫, 季宇寒, 李世超, 等. 农业机械导航技术研究进展[J]. 农业机械学报, 2020, 51(4): 1-18.

[5] 陈智康, 刘佳, 王丹丹, 等. 改进 Dijkstra 机器人路径规划算法研究[J]. 天津职业技术师范大学学报, 2020, 30(3): 30-35.

[6] 迟旭, 李花, 费继友. 基于改进 A*算法与动态窗口法融合的机器人随机避障方法研究[J]. 仪器仪表学报, 2021(3): 132-140.

[7] 陈昱衡, 吴鸿云, 郭旭, 等. 基于采矿车动力学分析的改进 A*算法全局路径规划[J]. 矿业研究与开发, 2021, 41(2): 170-177.

[8] 郭超, 陈香玲, 郭鹏, 王强, 汪世杰. 基于时空 A*算法的多 AGV 无冲突路径规划[J]. 计算机系统应用, 2022, 31(4): 360-368. https://doi.org/10.15888/j.cnki.csa.008454

[9] 代玉梅, 张瑞玲, 马黎. 改进 A*算法的采摘机器人路径规划与跟踪控制[J]. 中国农机化学报, 2022, 43(3): 138-145. https://doi.org/10.13733/j.jcam.issn.2095-5553.2022.03.019

[10] 张超, 李欣, 赵祥伟. 基于改进 Dijkstra 算法的输电线路路径规划[J]. 电力勘测设计, 2022(2): 1-5. https://doi.org/10.13500/j.dlkcsj.issn1671-9913.2022.02.001