计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2019,28(5):95−101 [doi: 10.15888/j.cnki.csa.006923]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于 A*的双向预处理改进搜索算法①

秦　锋[1], 吴　健[1], 张学锋[1], 赵晶丽[2]

[1](安徽工业大学 计算机科学与技术学院, 马鞍山 243032)
[2](滁州职业技术学院 信息工程系, 滁州 239000)
通讯作者: 吴　健, E-mail: 402160329@qq.com

摘　要: 本文针对传统 A*算法存在冗余路径点较多与单向搜索耗时较长的缺点, 提出了一种改进 A*算法. 该算法采用双向预处理结构减少冗余节点数, 并通过归一化处理和增加节点标记信息进一步优化估价函数提高遍历速度. 利用仿真软件对改进 A*算法进行实验, 并与其它经典路径规划算法进行比较. 仿真结果表明, 改进后的 A*算法较于传统 A*算法能以较低的搜索节点数和搜索时长较好的完成全局路径规划.

关键词: A*改进算法; 路径规划; 预处理; 估价函数

引用格式: 秦锋,吴健,张学锋,赵晶丽.基于 A*的双向预处理改进搜索算法.计算机系统应用,2019,28(5):95–101. http://www.c-s-a.org.cn/1003-3254/6923.html

Improved Search Algorithm Based on A* for Bidirectional Preprocessing

QIN Feng[1], WU Jian[1], ZHANG Xue-Feng[1], ZHAO Jing-Li[2]

[1](School of Computer Science and Technology, Anhui University of Technology, Ma'anshan 243032, China)
[2](Department of Information Engineering, Chuzhou Vocational and Technical College, Chuzhou 239000, China)

Abstract: In this study, an improved A* algorithm is proposed for the traditional A-star algorithm, which has many redundant path points and long-term one-way search. The proposed algorithm uses a bidirectional preprocessing structure to reduce the number of redundant nodes, and further optimizes the evaluation function to improve the traversal speed by normalizing the processing and adding node marker information. Simulation software is used to simulate the improved A* algorithm and compare with other classical path planning algorithms. The simulation results show that the improved A* algorithm can complete the global path planning with lower search node number and search duration than the traditional A* algorithm.

Key words: A* improved algorithm; route plan; pretreatment; valuation function

引言

导航系统中的一项重要技术路径规划算法现已成为被广泛运用的核心, 属于智能出行研究的热点问题. 大批学者围绕 Dijkstra 算法提出了很多改进措施, 例如最常见的 A*算法. 除了经典算法之外, 还有遗传算法、蚁群算法、粒子群优化算法等等. 文献[1]对 Dijkstra 经典算法做了研究工作, 算法采用传统遍历模式搜索全局地图网格上的节点, 当网格中存在大量节点时, 由于算法的盲目搜索会造成大量冗余节点被遍历, 此时的算法效率非常低下[2]. 以传统 Dijkstra 算法作基础, A*算法于 1980 年被 Nilsson 首先提出[3], 该算法是一种利用启发式信息函数作支撑的全局路径规划

① 基金项目: 安徽省教育厅课题 (KJ2017ZD05); 安徽省自然科学基金青年项目 (1808085QF210)
Foundation item: Research Fund of Education Bureau, Anhui Province (KJ2017ZD05); Young Scientists Fund of Natural Science Fund of Anhui Province (1808085QF210)
收稿时间: 2018-11-27; 修改时间: 2018-12-18, 2019-01-03; 采用时间: 2019-01-07; csa 在线出版时间: 2019-05-01

算法, 算法明显优势在于扩展当前节点之前已经引入了全局路网信息作为启发信息, 用来权衡下一个最优路径节点的位置. 在扩展过程中需要不断的迭代计算所有候选节点距离目标节点的估价成本, 估价成本越接近实际耗费成本则该条路径越准确. 在文献[4]中引入用类椭圆型作为限制区域, 但计算复杂度大幅度提高. 文献[5]中又进一步修改限制区域类型为矩形, 降低了运行时间. 在遍历结构确定的情况下, 采用空间换时间的优化策略. 王士同等人[6]红色首先提出了启发式双向搜索路径规划算法, 并通过实验论证了该算法的可行性. 李清权等人[7]红色在传统算法上进行改进, 改进出一种立足于节点搜索的双向 A*启发式路径规划算法. 本质上 A*算法属于最佳优先算法范畴[8], 但该算法也存在明显的缺点, 当存在多个最小值时或者路网中道路疏密程度不均时 A*算法并不能保证搜索的路径为最优, 会出现文献[9]所论述的停滞不前的无限循环. 并且传统 A*算法在提高搜索精度的同时需要栅格地图被分割的更小, 为此带来的另一大缺点即搜索时间会成指数级别的增长, 因此本文在后续的改进措施中引入了二叉堆进行优化. 同样估价函数的合理性也成为 A*算法的一大缺点, 会直接影响搜索效率. 本文作者在研究过程中发现大部分的全局静态路网规划中障碍物节点信息总是保持不变, 因此在规划路径之前系统即可提前预处理地图信息将障碍物的邻节点信息存入内存, 以便在同样环境地图规划中直接通过键值对索引相应的邻节点信息, 避免了重复遍历地图的繁琐程序. 针对传统 A*算法存在的缺点, 本文对 A*算法提出了改进措施意图提高算法效率和精度. 引入预处理模式以及对遍历结构进行优化, 其次对启发函数式进行归一化操作消除不同量纲之间产生的影响. 同时在前文提及的双向搜索基础上引入堆结构作为效率较高的优先级序列进一步为路径规划提速. 实验结果表明了改进算法的可行性与合理性.

1 路径规划问题概述

1.1 路径规划的分类

路径规划按照环境情况可分为全局路径规划和局部路径规划. 全局路径规划代表地图环境信息全部已知的情况, 而局部路径规划则为环境部分未知或者完全未知需要自身感知力进行实时获取. 另外环境情况又可分为静态和动态, 至此路径规划最终细分为如下四类, 分别为全局静态路径规划、全局动态路径规划、局部动态路径规划、局部静态路径规划. 全局静态路径规划主要包含构型空间法、自由空间法、栅格法. 局部静态路径规划主要包含人工势场法、模糊逻辑算法. 本文选用的是基于栅格法的全局静态路径规划.

1.2 路径规划的步骤

路径规划的整体结构基于算法具体功能实现, 按照“整体感知-地图建模-最优路径规划-算法执行操作”的过程依次执行[10]. 首先对整体环境信息执行初始化操作, 依据节点和障碍物特性选择合适的地图. 其次对整张地图进行建模操作, 建模核心是需要与路径规划操作相适应, 地图必须能够高效的存储、提取、更新节点信息. 本文将物体的移动轨迹细分化变为单个方向上的路径信息, 将其离散化后路径信息便被保存在了栅格地图中. 选用栅格地图的原因在于相对于别的地图建模其更具位置唯一性, 因此在每个栅格节点中忽略物体的在任意方向上的微小偏移, 只参考节点四周 4 邻域或者 8 邻域的方向信息便于操作[11–13], 图 1 所示. 地图利用矩阵模式存储节点, 查找任意节点的行列数即可得知该节点具体信息. 建立栅格模型后, 需要对地图模型整体进行完整编码标记信息, 常见的用 0 或 1 代表当前节点有无障碍, 编码标记完成之后还需要对地图环境做几个既定的准则, 如限定障碍物的位置和大小不变, 栅格必须保证统一尺寸大小等. 建模完成后配合改进过后的算法对整张地图进行路线规划, 最后回溯拼接之前遍历过的节点即得到一条规划完成的路径.

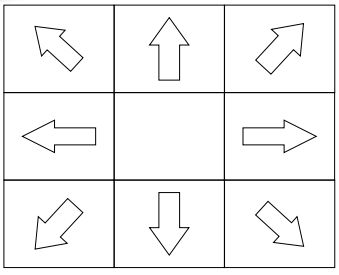


图 1 节点运动方向

1.3 Dijkstra 算法

Dijkstra 算法目的为找出从图中的某个顶点出发到达另一个顶点所经过边的权重之和最小的那一条路径. 算法采用贪心策略模式使用广度优先搜索解决赋权有向图或者无向图的单源最短路径. 主要思想如下, 首先引入一个向量 M, 它的的每个分量 $M[i]$ 代表从起

始顶点 V 开始到其他各个顶点$V[i]$的距离. 若 V 到 $V[i]$有弧则将$M[i]$设为弧上的权值. 显然$M[j]=Min\{M|Vi\in V\}$即从起始顶点出发到顶点V_j距离最短的一条路径, 此时路径为(V,V_j). 接着定义一个存放从起始点 V 出发的最短路径的顶点集合 S. 则下一条距离最优的路径必然为$M[j]=Min\{M[i]|V_i\in V-S\}$, $M[i]$为弧(V,V_i)上的权值或者为$M[k](V_k\in S)$和弧(V_k,V_i)权值总和 (k 为 S 中的一个顶点). 直到遍历完整张地图网络之后, 算法根据局部最优准则选出一条最优路径, 但是缺点在于整个搜索过程属于盲目搜索只关注实际耗价值没有考虑到待选节点对目标节点所造成的影响, 因此规划过程将耗费大量的时间.

1.4 跳点算法

近年来研究人员推出了一种高效寻路算法 Jump Point Search, 也就是所谓的跳点算法. 其本质针对有序序列查找跳点与二分法类似, 主要思想就是大范围跳跃式搜索栅格网络中对称的路径, 只把其中的跳跃点当作待搜索的栅格节点. 其明显优势在于大幅度减少 openlist 中的待选节点得益于在固定间隔的跳跃搜索过程中已经剔除了网格地图中大量无用节点. 具体的实现大致涉及到两个方面, 第一步先遍历 openlist 表筛选出一个最优节点, 然后从几个既定好的方向展开搜索, 将每个方向获得的跳点加入到 openlist 表中. 第二步在之前得到的各个方向上的跳点中筛选出代价最小的跳点作为新的起始点, 从该点开始继续向既定方向展开搜索. 重复以上两步操作, 直到终点被存入到 openlist 表中, 寻路结束. 但该算法仍然存在一些不可避免的限制因素, 首先对地图要求条件较高必须是规则的栅格型地图, 不支持复杂地形, 网格节点不能带权重, 但相比传统 A*算法, 其算法效率已经提高了一倍左右.

1.5 传统 A*算法

A*算法引入启发函数$h(n)$对全局信息进行估价, 将此估价作为最佳路径可能性的量度. 算法中关键的是两个 list 列表, 用于存储已经遍历和未遍历的栅格节点, 同时配合启发函数对地图上任意栅格节点距目标节点的距离进行有效估价, 估价值越接近实际耗费, 在四邻域或八邻域的情况下则会更好的寻找搜索方向. 算法主要表达式: $f(n)=g(n)+h(n)$, 其中$f(n)$为初始状态经过状态 n 到达目标状态的代价估计, $g(n)$为初始状态到达状态 n 的移动耗费, $h(n)$为状态 n 到达目标状态的最低估计代价. A*算法步骤如下描述:

Step 1. 将起始点加入 openlist, 并初始化 openlist 和 closedlist.

Step 2. openlist 若为空, 结束算法. 若不为空, 计算最小的点, 将该节点添加到最优路径上.

Step 3. 对当前处理的节点, 以 4 方向或者 8 方向计算邻节点, 遍历 closedlist 和 openlist, 若均不包含邻节点, 则将其加入 openlist.

Step 4. 循环执行 Step 2 和 Step 3 到算法结束, 回溯遍历之前拓展选中的路径, 则路径规划完成.

其中一般选用曼哈顿距离、欧式距离或者切比雪夫距离 (当 $h(n)$=0 时, A*算法将变为 Dijkstra 算法),

$$Manhattan\ Distance=|X1-goalX|+|Y1-goalY| \quad (1)$$

$$Euclidean\ Metric\ Distance=\sqrt{(\mathrm{X1}-\mathrm{goalX})^2+(Y1-goalY)^2} \quad (2)$$

$$Chebyshev\ Distance=\max\left[|X1-goalX|,|Y1-goalY|\right] \quad (3)$$

2 改进双向预处理 A*算法概述

综合上述传统算法的优缺点, 本文提出改进策略提高算法效率. 以经典 A*算法为基础, 先执行双向搜索策略从起点终点同时开始搜索, 对算法中的角度和方向因子实现归一化处理以消除不同量纲对节点参数所带来的影响. 同时提出改进的算法结构, 将栅格地图中的障碍物信息预处理存入内存以减少资源消耗, 并且对遍历数组时的节点添加布尔类型的判定信息, 初始化地图后可直接依据此判定信息对节点是否在两个列表中进行判定而无须再做重复性的遍历操作, 进一步提高算法效率.

2.1 并行化处理

引入并行化处理模式即实现双向搜索算法, 该模式将搜索过程分解为前向后向两个独立的搜索进程, 起点终点分别作为对方向的目标节点. 双向搜索核心思路在于终止条件的判定, 在理想状态下两条搜索路径会在地图路径中心点相遇, 此时两个方向上的 openlist 中出现重复的节点即可判定规划完成, 这样做可大大减少单向遍历搜索所耗费的时间和空间资源.

2.2 矢量化和归一化的改进

A*算法在传统估价函数的影响下, 只会单一的进行往返搜索生成大量的搜索节点. 而在实际情况中当前节点距离目标节点的估价值一定不大于实际路径耗

费值, 因此需要加快当前节点向目标节点的收敛速度即增大估价函数 $h(n)$ 的权值. 考虑到大部分的权值设定都视具体的地图情况例如地图大小, 障碍物数量, 地形因素而定因此具有片面性. 并且当前节点距离目标节点的估价距离占实际距离的比重与当前节点在地图中的位置相关. 本文在前文基础上, 在当前节点远离目标点, 此时估价值低于实际值, 增大权重. 相对应的当前节点靠近目标点时, 当前估价值接近实际值, 相对应的降低权重. 因此本文引入以指数衰减方式的权重因子, 对传统 A*算法表达式做出如下修改. 公式为:

$$f^*(n)=(1-\lambda^*)\times g(n)+\lambda^*\times h(n) \tag{4}$$

权重因子$\lambda^*=\sqrt{e^{h(n)}}$. 这样修改的优势在于既可保证 $g(n)$ 与 $f(n)$ 的权重系数相互影响, 又可使 $h(n)$ 的权重系数的变化量与自身保持正相关, 便于调整当前节点向目标节点的收敛速度. 同样在估价函数 $h(n)$ 中也需要引入距离和角度两个矢量因子[14–18]. 意图减少搜索方向偏差过大的地图节点, 大幅度降低搜索节点数, 释放计算机资源, 公式为:

$$h(n)=w_1\times L+w_2\times\alpha \tag{5}$$

式中的w_1与w_2分别为距离和角度的权重系数, 其取值范围分别为 0.55–0.65 和 0.35–0.45[15]. L 表示当前节点距目标点的距离值, α表示起点至当前节点连线的线段与当前节点至目标节点连线线段的夹角值. 根据地图大小引入权重系数的优势在于, 同时权衡距离信息与方向信息, 加快搜索进程. 由于不同的评价标准具有不同的量纲, 为了消除指标之间量纲的影响, 需要进行数据归一化标准处理, 保留其本身所有特性, 但是减少参数大小产生的影响, 归一化大大加快了梯度下降求最优解的速度. 针对本文的思路, 对 $h(n)$ 估价函数的距离和角度因子进行归一化处理, 对遍历进行提速. 本文对文献[19]的估价函数提出改进, 采用 Z-score 标准化方法对距离和角度消除量纲影响因素, 标准化公式为:

$$X^*=(X-\mu)/\sigma \tag{6}$$

其中, μ 为所有样本数据的均值, σ为所有样本数据的标准差. 将距离和角度因子代入 (n 代表邻节点的数量) 公式中:

$$\alpha^*=\frac{\left(\alpha-\frac{1}{n}\sum_{i=1}^{n}\alpha_i\right)}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\alpha_i-\frac{1}{n}\sum_{i=1}^{n}\alpha_i\right)}} \tag{7}$$

$$L^*=\frac{\left(L-\frac{1}{n}\sum_{i=1}^{n}L_i\right)}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(L_i-\frac{1}{n}\sum_{i=1}^{n}L_i\right)}} \tag{8}$$

此时$h^*(n)$修改为

$$h^*(n)=(1-\omega)\times L^*+\omega\times\alpha^* \tag{9}$$

原算法 $f(n)$ 改进为

$$f^*(n)=(1-\lambda^*)\times g(n)+\lambda^*\times h^*(n) \tag{10}$$

式 (10) 意义在于当估价函数$h^*(n)$数值较大时, 权重因子增大同时$g(n)$占比重减小, 当前节点加速向目标节点收敛. 当$h^*(n)$较小时, 权重因子也随之减小反之$g(n)$占比重增大. 当权重因子逼近 1 时表明当前节点已经靠近目标点. 仿真实验中表 2 的数据结果表明相比于文献[19]的 A*算法和跳点算法, 搜索节点数和搜索时长有明显降低, 实时性有显著提高. 同时参照几张运行效果截图, 表明在距离和角度这两个权重系数的影响下, 搜索深度明显减少, 搜索方向被显著的约束在更趋向于目标点的方向上证明了该公式的可行性.

2.3 基于堆结构的完全排序

传统 A*算法中, 计算机性能耗费关乎地图大小和反复遍历 openlist 中 F 值最小点这两个因素, 所以路径规划耗费的时长和所达到的精度取决于算法中存储列表方式的设计. 本文在传统 A*算法中引入二叉堆. 二叉堆实则是以堆结构存在的一种特殊的树, 可以保证数据结构的有序性以提高搜索效率, 是一种优先级序列. 它的优点在于在堆结构中子节点与父节点的键值总是以一种特定的有序关系相联系, 由此带来的好处是在计算过程中并不需要直接得到完整的全局有序信息, 而只需要全局节点的极值以避免造成不必要的资源浪费. 本文中堆结构被用于优化从开启列表移除节点数据, 向关闭列表添加节点数据, 可以保证两个列表始终保持内部数据有序. 本文引入的二叉堆结构主要涉及向 open 列表添加新元素和在切换至 close 列表后从堆顶删除 F 值最小的元素. 添加元素操作主要步骤为, 首先将起始节点添加至 open 列表, 同时计算新元素的 G、H、F 值并将新元素添加至开启列表的底端, 然后依次比较该元素和该元素的父节点直到该元素到达表中正确的位置. 删除操作主要步骤为在删除某个元素之后将当前的末元素移动至堆顶端, 依次比较该元素和它两个子节点元素的 F 值, 如果该元素的 F 耗价值高于其两个子节点则交换它与其中较低 F 值的节

点, 重复该过程直到该元素找到在堆中的相应位置.

实现二叉堆主要伪代码如下:

算法 1. 向堆中添加新元素

```
获取表最后一位下标;
last←open. length-1;
while (last > 1)
half←last;
与父节点比较 F 值大小;
If(open[last]. F>=open[half]. F) THEN break;
在表中与父节点调换位置;
temporary←open[last];
open[last]←open[half];
open[half]←temporary;
```

算法 2. 从堆中删除元素

```
将堆底端元素移至堆顶;
open[1]←open[last];
重新获得堆末位元素下标
last←open. length-1;
head←1;
先比较堆顶节点两个子节点 F 值大小, 找出较小 F 值的子节点;
childMin←open[child1]. F<open[child2].F?child1:child2;
比较父节点与较小子节点 F 值大小;
IF(open[head]. F<=open[childMin]. F THEN break;
若父节点 F 值大于子节点 F 值, 则交换位置;
temporary←open[head];
open[head]←open[childMin];
open[childMin]←temporary;
交换父节点与较小子节点的位置;
head =childMin;
```

2.4 实现预处理结构

前面已经提及采用并行二叉堆且加入矢量化因子已经提高了计算性能, 但优化的实质大多数是空间换时间, 因为在全局静态地图中障碍物信息保持不变并且邻域栅格的耗价值也是不变的, 但如果每次寻路时都去计算邻节点网格是否存在障碍物则会消耗大量时间. 因此把算法中部分过程的计算先执行预处理后存入内存. 本文首先对障碍物邻节点进行预计算, 对当前节点首先判断该节点周围是否存在障碍物并同时计算邻域栅格的耗价值. 具体步骤如下, 首先为所有的节点添加一个新的代表邻节点的数组 AdjNodeArray, 同时构造新的邻节点耗费值数组 AdjNodeCost, 这两个数组是为了初始化时存入邻域的节点信息. 保证 AdjNodeArray[i]与 AdjNodeCost[i]两个数组的映射关系以便于在查询某个邻域节点时可直接对应查找其耗价值. 其次在地图初始化的过程中, 先进行预处理操作遍历所有的地图网格, 找出地图中节点标记编号为 0(栅格地图中障碍物编号为 1, 非障碍物为 0) 的所有非障碍方格节点存入邻节点 AdjNodeArray, 同时计算邻节点耗费值映射存入 AdjNodeCost. 最后涉及到具体寻路过程需要判断当前节点附近邻节点信息时, 直接从邻节点数组 AdjNodeArray 中读取当前节点的属性值就能得知此节点周围邻节点的全部属性值而无需再重复计算. 例如在预处理地图之后寻路过程中需判定二维坐标号为 (6, 9) 的当前节点信息以及下一节点的选择情况, 只需要先在邻节点数组找到当前点判定是否为非障碍物节点, 并根据映射关系就可直接得到该节点的邻节点耗价值, 接着相应的通过估价函数选择下一步行进的路径节点, 重复以上操作直至找到终点. 预处理大大缩减每次遍历地图计算计算节点信息的时间. 另外算法结构中每次搜索数组采用的常规遍历操作 indexof 实际占用了大量的运算时间, 本文同样对数组遍历进行优化, 对每个节点赋予布尔类型属性 isopen 和 isclosed 作为标记信息, 每次对当前节点进行 push 操作时, 在 openlist 中设置当前节点的布尔属性 isopen 为 true, 当下一步操作将该节点移出开启列表后, 同时更改 isopen 值为 false, 这样处理的优势在于, 判断当前节点是否在 openlist 中, 只需要对应的获取它的布尔值即可. 同理可以判断节点是否存在于 closedlist 中.

3 实验结果对比分析

为了验证改进 A*算法的实际效果, 本文设置不同的对照组在栅格地图中进行大量的数据测试. 实验环境为 Intel Core i5, Win10, 8 GB 内存, 采用 Matlab 2012a 编译.

3.1 实验过程

本文建立一个 20×30 的栅格地图, 将起始点与终点设定在栅格地图的 (2, 2), (20, 30) 坐标处. 如图 2 所示.

应用本文改进 A*算法对图 2 所示地图模型进行计算, 选取权重系数w为 0.59 首先进行第一次预处理将地图信息存入内存, 预处理完成后进行第二次路径规划, 运行结果如表 1 所示.

其次保持地图环境不变, 选用其它传统算法同样对该地图进行仿真计算, 得出多组结果如表 2 所示. 其中, M 和 E 分别代表选用曼哈顿距离或欧式距离, Dijkstra 与跳点算法不区分邻域数及选用的距离函数. 部分程序运行截图如图 3 所示.

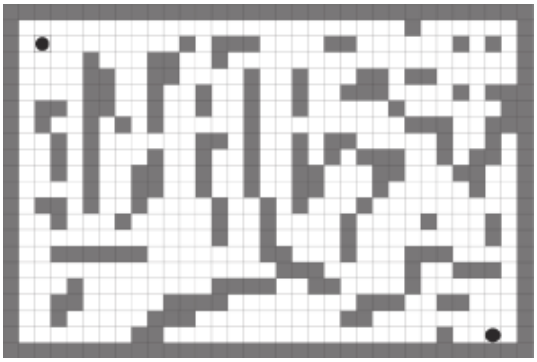


图 2 栅格地图

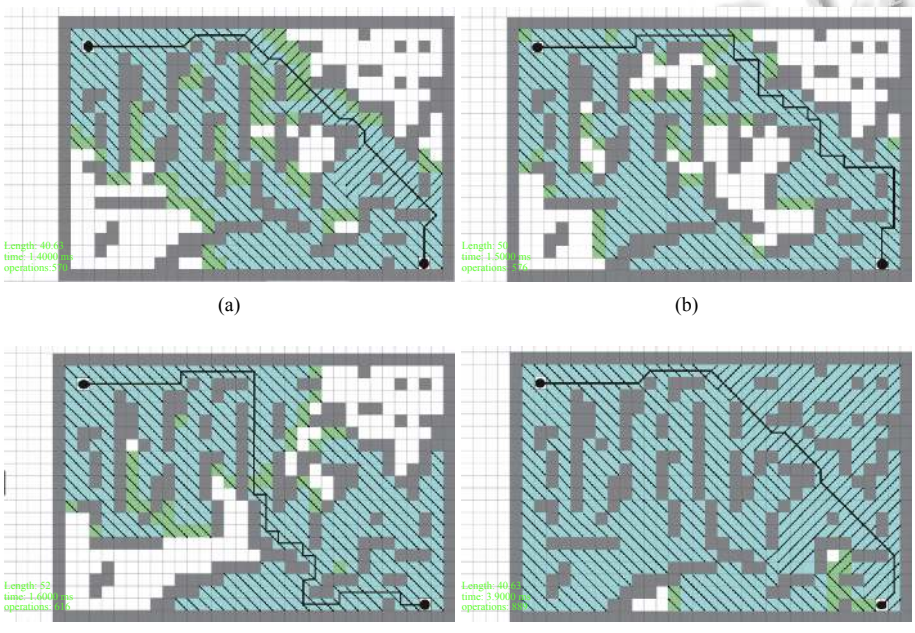
图 3 中图 (a)、(b)、(c)、(d) 依次是改进 A*算法分别计算 8 邻域欧式距离, 4 邻域曼哈顿距离, 4 邻域欧式距离运行效果截图, 图中黑色实心圆点为起点与终点, 粗体黑色实线为规划路线, 斜向黑色实线表示本次规划所遍历的节点方格. 以及 Dijkstra 算法运行效果截图.

表 1 预处理前后运行效率对比

邻域	指标	第一次预处理	预处理完成	提速百分比 (%)
8 邻域	Distance	42.28	42.28	27.8
	Time(ms)	1.8	1.3	
	节点数	454	454	
4 邻域	Distance	50	50	6.67
	Time(ms)	1.5	1.4	
	节点数	576	576	

表 2 不同算法运行效率对比

邻域	指标	Dijkstra 算法	文献[19]A*算法	Jump 算法	改进 A*算法
8 邻域 *M*	Distance	40.63	40.63	42.38	42.28
	Time(ms)	3.9	2.1	1.6	1.3
	搜索节点数	889	499	699	454
4 邻域 *M*	Distance	40.63	50	42.38	50
	Time(ms)	3.9	4.1	1.6	1.4
	搜索节点数	889	761	699	576
8 邻域 *E*	Distance	40.63	40.63	40.63	40.63
	Time(ms)	3.9	2	2.3	1.4
	搜索节点数	889	563	811	570
4 邻域 *E*	Distance	40.63	50	40.63	52
	Time(ms)	3.9	5.1	2.3	1.6
	搜索节点数	889	797	811	616



(a) (b)

(c) (d)

图 3 几种规划算法运行效果截图

3.2 计算效率对比

从表 2 可以看出, 在相同的地图环境下, 本文改进的 A*算法所仿真得到的路径相比于文献[19]的 A*算法和跳点算法, 虽然路径长度有一定程度的增加, 但因为其归一化的方向和角度因子, 以及对数组遍历的优化, 使得遍历节点数比例降低, 尤其在 4 邻域的条件下节点数下降趋势更为明显, 数据表明在 8 邻域标准下, 本文改进过后的算法相比于 Dijkstra, 文献[19]的 A*, 跳点算法节点数分别下降 48. 93%, 9. 01%, 35. 05%. 在 4 邻域标准下分别下降 35. 21%, 24. 31%, 17. 60%. 证明了改进 A*算法的矢量化与归一化处理能够有效的降低搜索节点数. 同时, 搜索时长较 A*算法与跳点算法都有不同程度的提升, 8 邻域条件下相比于文献[19]A*与跳点算法, 时长分别降低 35% 和 18. 75%, 4 邻域条件下, 时长分别降低 65% 和 39. 13%, 由此可以证明引入堆结构可大幅度降低遍历时长. 同样表 1 可以看出第一次预处理地图信息之后的第二次遍历时长在 8 邻域和 4 邻域的标准下分别下降 27. 78% 与 6. 67%, 表明对地图节点的预处理操作提前处理节点附近的障碍物信息并且进行内存预存储以及对数组添加布尔型标记信息的优化, 都能有效降低后续遍历时长.

4 结束语

如今以启发式搜索为核心的 A*算法在导航寻路规划领域得到了广泛的应用. 结合经典 A*算法与本文的双向预处理改进 A*算法, 本文算法在规划时长方面更具优势, 得益于改进的预处理算法结构以及权重系数对归一化处理后的方向距离因子的收敛加速特性, 运行实验结果表明本文改进算法规划路径的可行性.

参考文献

1 秦昆, 关泽群, 李德仁, 等. 基于栅格数据的最佳路径分析方法研究. 国土资源遥感, 2002, 14(2): 38–41. [doi: 10.3969/j.issn.1001-070X.2002.02.009]

2 Cormen TT, Leiserson CE, Rivest RL. Introduction to Algorithms. Cambridge: MIT Press, 2001.

3 Mohajer B, Kiani K, Samiei E, *et al*. A new online random particles optimization algorithm for mobile robot path planning in dynamic environments. Mathematical Problems in Engineering, 2013, 2013: 491346.

4 陆锋, 卢冬梅, 崔伟宏. 交通网络限制搜索区域时间最短路径算法. 中国图象图形学报, 1999, 4(10): 849–853.

5 Fredman ML, Tarjan RE. Fibonacci heaps and their uses in improved network optimization algorithms. Proceedings of the 25th Annual Symposium on Foundations of Computer Science. Singer Island, FL, USA. 1984.

6 王士同. 双向启发式图搜索算法 BFFRA. 电子学报, 1990, 18(6): 34–39. [doi: 10.3321/j.issn:0372-2112.1990.06.006]

7 郑年波, 李清权, 徐敬海, 等. 基于转向限制和延误的双向启发式最短路径算法. 武汉大学学报·信息科学版, 2006, 31(3): 256–259.

8 熊壬浩, 刘羽. A*算法的改进及并行化. 计算机应用, 2015, 35(7): 1843–1848.

9 熊伟, 张仁平, 刘奇韬, 等. A*算法及其在地理信息系统中的应用. 计算机系统应用, 2007, 16(4): 14–17. [doi: 10.3969/j.issn.1003-3254.2007.04.004]

10 王殿君. 基于改进 A*算法的室内移动机器人路径规划. 清华大学学报 (自然科学版), 2012, 52(8): 1085–1089.

11 张宏烈. 移动机器人全局路径规划的研究[硕士学位论文]. 哈尔滨:哈尔滨工程大学, 2002.

12 魏宁, 刘一松. 基于栅格模型的移动机器人全局路径规划研究. 微计算机信息, 2008, 24(11): 229–231. [doi: 10.3969/j.issn.1008-0570.2008.11.094]

13 陈华华, 杜歆, 顾伟康. 基于遗传算法的静态环境全局路径规划. 浙江大学学报 (理学版), 2005, 32(1): 49–53. [doi: 10.3321/j.issn:1008-9497.2005.01.013]

14 Chabini I, Lan S. Adaptations of the A* algorithm for the computation of fastest paths in deterministic discrete-time dynamic networks. IEEE Transactions on Intelligent Transportation Systems, 2002, 3(1): 60–74. [doi: 10.1109/6979.994796]

15 Whangbo TK. Efficient modified bidirectional A* algorithm for optimal route-finding. In: Okuno HG, Ali M, eds. New Trends in Applied Artificial Intelligence. Berlin Heidelberg: Springer, 2007: 344–353.

16 Elbeltagi E, Hegazy T, Hosny AH, *et al*. Schedule-dependent evolution of site layout planning. Construction Management and Economics, 2001, 19(7): 689–697. [doi: 10.1080/01446190110066713]

17 Hart PE, Nilsson NJ, Raphael B. Correction to "a formal basis for the heuristic determination of minimum cost paths". ACM SIGART Bulletin, 1972, (37): 28–29.

18 Guesgen HW, Mitra D. A multiple-platform decentralized route finding system. Proceedings of the 12th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems. Cairo, Egypt. 1999. 707–713.

19 史辉, 曹闻, 朱述龙, 等. A*算法的改进及其在路径规划中的应用. 测绘与空间地理信息, 2009, 32(6): 208–211. [doi: 10.3969/j.issn.1672-5867.2009.06.070]