计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2018,27(7):255−259 [doi: 10.15888/j.cnki.csa.006421]

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

A*算法在 Shortest-Path 方面的优化研究①

梁昭阳, 蓝茂俊, 陈正铭

(韶关学院 信息科学与工程学院, 韶关 512005)
通讯作者: 梁昭阳, E-mail: 3294261606@qq.com

摘　要: 在游戏和地理信息系统开发等领域中, 专门针对最短路径搜索方面的优化研究较多, 尤其是最短路径中启发式搜索算法中的 A*算法的效率优化研究. 本文将针对在人工智能或算法研究中的使用的地图大多数是基于任意图而不是网格图的状况, 通过任意图与网格图及方向的相结合, 提出了三种优化 A*算法的启发式函数搜索策略, 较好地减小了算法搜索的范围和规模, 有效地提高了 A*算法的运行效率. 最后的实验结果显示, 与传统的 A*算法相比较, 优化启发搜索策略后的 A*算法寻径更快速, 更准确, 计算效率更高.

关键词: 启发式搜索策略; A*算法; 方向; 最短路径搜索

引用格式: 梁昭阳,蓝茂俊,陈正铭.A*算法在 Shortest-Path 方面的优化研究.计算机系统应用,2018,27(7):255–259. http://www.c-s-a.org.cn/1003-3254/6421.html

Optimization of A* Algorithm in Finding Shortest-Path

LIANG Zhao-Yang, LAN Mao-Jun, CHEN Zheng-Ming

(College of Information Science and Engineering, Shaoguan University, Shaoguan 512005, China)

Abstract: In the field of NPC game or GIS system development, there are more studies on finding the efficient method of the shortest path search problem, especially the research of A* algorithm efficiency optimization in the search of finding shortest path algorithm. Most of the maps used in artificial intelligence or algorithm research are based on arbitrary graphs rather than grid-based graphs. Based on above mentioned scenario, by combining arbitrary graphs with grid graphs and directions, an optimization of A* algorithm is proposed in this study. The heuristic search strategy is improved, which can reduce the scale and range of algorithm search and improve the efficiency of A* algorithm. Finally, the experimental results show that compared with the traditional A* algorithm, after the optimization of A* algorithm, the heuristic search strategy is more accurate and the computational efficiency became more efficient and more quickly.

Key words: heuristic search strategy; A* algorithm; direction; fnding shortest-path

A*算法是一种优秀的启发式搜索算法, 它整合了 Greedy Best-First-Search 算法、Dijkstra 算法和一些距离计算公式的算法思想, 并利用了估值函数或评估函数作为启发方法, 从而可以高效动态地求得一条可能最优的最短路径, 因而 A*算法一直被游戏开发行业和 GIS 系统开发等领域所广泛采用[1]. 但目前的 A*算法研究大多数都是基于任意图而不是基于网格图的游戏设计的, 本文将从多角度对 A*算法在智能寻路中的搜索效率进行改进.

1　传统 A*算法的简介与分析

A*算法是一种寻找最短路径的有效方法, 它常用

① 基金项目: 韶关学院大学生创新创业训练计划立项项目 (国家级)(201710576001); 韶关学院第十六批教育教学改革研究一般项目 (SYJY20151623)
Foundation item: Innovation and Entrepreneurship Training Program for Undergradute of Shaoguan University (National Level)(201710576001); The 16th Batch of Research Projects on Education and Teaching Reform of Shaoguan University (SYJY20151623)
收稿时间: 2017-10-20; 修改时间: 2017-11-21; 采用时间: 2017-12-07; csa 在线出版时间: 2018-06-27

于现代游戏的非玩家游戏 (NPC) 设计以及地理信息系统 (GIS) 开发等中. A*算法像很多求解最短路径的算法一样, 都可以找到地图上两点间的最短路径. 在简单图上, 它跟 Greedy Best-First-Search (贪心最佳路径搜索算法) 的效率基本上是相同的, 同样可以利用启发式进行最短路径的搜索. 启发式 (Heuristic Function) 搜索是指在整个将要搜索的地图中, 先对每一个将要访问的节点进行一个代价评估, 然后选择最好的节点, 再不断迭代更新搜索, 直至寻找到目标节点.

启发式搜索 A*算法公式表示为:

$$F(n)=G(n)+H(n) \tag{1}$$

$G(n)$ 表示从初始节点 S 到目标节点 T, 在其搜索路径的过程中, 到达期间某一节点 n 所花费的代价. $H(n)$ 表示从当前节点 n 到目标节点 T 的最优路径中将要花费的估计代价, 即最短路 (Shortest-Path). 简而言之, $G(n)$ 是实际代价, $H(n)$ 是估计代价, $F(n)$ 则表示节点 n 的估价函数. 在保证存在最短路径的条件下, A*算法的效率高低在于 $H(n)$ 的选取, 假设 $R(n)$ 表示 n 到目标节点的距离的实际数值, 如果 $H(n)>R(n)$, 则将要访问的节点数目越少, 搜索的范围越窄, 效率越高, 但不能保证得到最优路径. 如果 $H(n)\le R(n)$, 则将要访问的节点数目越多, 搜索的范围越广, 效率越低, 但肯定能得到最优路径[2].

在 A*算法的实现上, 使用 OPEN 列表和 CLOSED 列表来存放节点. 将起始点放入 OPEN 列表中, 对 OPEN 列表中的节点按照启发函数值进行升序排序, CLOSED 列表则用来存放计算后得出的 OPEN 列表的适合要求的节点. 取出 OPEN 列表中第一个节点, 如果是目标节点就直接输出最短路径. 否则就计算下一个节点的估价函数值, 将最优的下一个节点放进 OPEN 列表中, 同时将当前节点从 OPEN 列表移到 CLOSE 列表中, 直至找到最短路径[3].

A*算法效率高的主要原因是增加了估价函数, 通过估价函数 $F(n)$, 合适地选取下一个代价最小的节点, 以此类推, 直到找到最终节点, 相比传统的无启发函数的其他最短路径算法, 其在效率上得到了明显的提升. 但是 A*算法的启发式函数是一种由经验确定的函数, 估计函数往往决定了 A*的路径规划, 节点的选择也充满了不确定性, 很难保证每次选择的节点都是最优解的, 具有一定几率上在最优节点的选择出现错误, 从而导致节点的意外删除或者多余的节点选择, 导致最后计算的路径为非最优路径. 另外在计算过程中还可能出现“死循环”的现象, 形成“回环”等的无用搜索, 而无法获取到最优路径. 因此, 如何优化启发式函数成为 A*算法效率高低的主要因素.

A*算法在实际运用时还可能会出现一个问题, 当实际的地图并不是一整块的二维网格, 它是被很多障碍物给划分成众多个形状不规则的地图块甚至是地图点, 并且每一个块和点的密集程度都不可能一样. 直接运用 A*算法对整张地图寻找最短路径会导致搜索效率低下.

2 A*算法的优化方案

根据传统的 A*算法可能出现的一系列问题, 将从以下三个方面去考虑优化和改良新的 A*算法.

2.1 对可能出现“死循环”的优化

可以利用“分治”的思想去解决在现实地图中可能出现的“死循环”(Dead Loop) 问题. 对现实中的地图进行具体地划分成若干个块, 然后再在这些块中进行最短路径搜索, 用来解决“死循环”的问题. 例如在游戏常见的二维地图上, 地图被河流或者道路划分成若干个不等的小块, 每一个小块都可以作为一个独立的小地图, 小地图又由复杂的网络节点所构成, 如果形成的结构还比较稠密, 则再划分成一块块的独立的小地图, 这样就可以把这些独立的小地图看成是一块稀疏图[4]. 可以先在稀疏图中进行最短路径的搜索, 然后继续在上一层的稠密图中搜索最短路径. 这种搜索策略可以用来克服“死循环”的问题, 而且对地图进行“分治”搜索到的路径也是最优最短的.

2.2 加入方向因子的优化

启发式函数 (Heuristic Function) 直接或者间接地决定着传统的 A*算法的搜索效率, 因此对于启发函数的优化处理是主要研究方向之一. 在传统方法的基础中引入方向这个因素, 用来提供更多的启发信息[5]. 在寻路过程中把与目标方向相反的节点给剪枝掉, 减少多余的计算量. 传统的 A*算法的启发式函数 $H(n)$ 是以距离信息为计算机引导方向的. 常用的有曼哈顿距离 (Manhattan Distance), 欧几里德距 (Euclidean Distance), 对角线距离 (Diagonal Distance), 切比雪夫距离 (Chebyshev Distance) 等, 而传统的 A*算法以欧几里德距离作为启发函数[6]. 地点 A 到地点 B 的欧几里德

距离 (Euclidean Distance) 为:

$$Euclidean\_distance = \sqrt{(\mathrm{A}.x-\mathrm{B}.x)^2+(\mathrm{A}.y-\mathrm{B}.y)^2} \quad (2)$$

但是用欧几里德距离成为启发式函数有着很明显的漏洞, 因为实际代价 $G(n)$ 可能会与启发式函数 $H(n)$ 发生相斥, 而且其开平方根会使计算机的计算开销更大. 所以, 可以使用 Octile 距离来进行优化, Octile 距离的主要思想就是假设只能进行 45°转弯.

$$\begin{aligned} Octile\_distance = &\max(|\mathrm{A}.x-\mathrm{B}.x|,|\mathrm{A}.y-\mathrm{B}.y|) \\ &+(\sqrt{2}-1)\times\min(|\mathrm{A}.x-\mathrm{B}.x|,|\mathrm{A}.y-\mathrm{B}.y|) \end{aligned} \quad (3)$$

上式中 max 表示取最大值, min 表示取最小值. 另外, 在选择最优节点路径时还受到了方向 (Direction) 因素的影响, 如果只用距离来作为约束条件, 那么它就可能不是最优的约束条件. 考虑从一个位置移动到下一个位置的最小代价为 D, 并加入方向因子 (Direction Factor)θ 来提高算法的效率与准确性[7].

$$F(n) = G(n) + a\times\sin(\theta) + D\times Octile\_distance(next) \quad (4)$$

其中 a 作为权重比例系数, 它取决了方向在 A*算法中的关键性, 对于不同的情况, 这个权重比例系数要按照算法经验进行调整, 权重比例系数的大小和目标的重要性以及策略的选择有关. θ 则表示起始节点和目标节点构成的矢量, 当前节点与下一个最优节点形成的矢量, 两个矢量之间的夹角[8].

2.3 对节点排序的优化

在传统 A*算法中, 每次访问 OPEN 表必然需要找到 $F(n)$ 值最小的节点, 如果对 OPEN 表进行整体的搜索比较, 会导致耗费大量的时间. 可以采取对 OPEN 表的节点进行排序, 在查找 $F(n)$ 值最小的节点时就会节省大量时间. 对一些极限数据, 单一的排序效率并不高. 因此, 可以根据不同的数量级和不同的情况去采取不同的排序. 当数据量相对大时就采用快排 (QuickSort), 并分段递归排序. 当分段后的数据量小于某个阀值, 就采用插入排序 (InsertionSort). 如果递归过深, 就采用堆排序 (HeapSort). 采取这种优化过的排序方法可以更有效地对 OPEN 表进行升序排列. 最后, 每次对 OPEN 表中加入新的节点的时候我们采用二分插入的方法, 使得 OPEN 表始终保持有序状态. 这样比每次都盲目地搜索整个 OPEN 表的速度要快至少 50% 的时间.

3 优化的 A*算法的伪代码

伪代码如下:

```
//Pseudo code
while OPEN has Node do
  if temp_Node equal E then output the path with temp_Node
  else do
    Init Queue = {
      The neighbour of temp_Node;
    }
    for each node in Queue do
      if CLOSE not contains node then
        //判断节点的搜索方向
        if Queue.Count > 1 and node is not in direction from S to E
        then
          push node into Temp
        else do
        if node is in direction from temp_Node to E then
          //计算该节点的启发式函数
          calculate the value of heuristic function for node
          push node into OPEN
          set temp_Node as node's parent node
        else do
          push node into Temp
      //将临时节点存放在 CLOSE 列表中
      push temp_Node into CLOSE
      pop temp_Node from OPEN
      if OPEN.Count =0 and temp_Node is not E
      Pop up the first node in Temp and push into OPEN
  if temp_Node is not E
    Can not find a way from S to E //不存在最短路径
```

4 实验和对比分析

对于传统 A*算法中的启发函数的优化处理[9], 加入方向因子 θ, 剪枝, 划分小地图, 部分细节代码优化等优化方案后的 A*算法的效率对比.

实验一. 单独对节点排序的优化, 寻找最短路径时访问的节点数.

从表 1 的数据可以看出, 在障碍物稠密程度相同的情况下, 随着地图网格数的增大, 在运行时间和网格访问的数量上, 优化过的 A*算法要比传统的 A*算法都要少一部分, 在效率上, 经过排序优化的 A*算法更优.

表 1 批量数据比较结果

地图网格数	第 1 次运行结果		第 2 次运行结果		第 3 次运行结果		第 4 次运行结果	
	优化 A*	传统 A*	优化 A*	传统 A*	优化 A*	传统 A*	优化 A*	传统 A*
40×40	534	619	532	624	533	616	534	620
50×50	864	950	864	951	865	953	863	952
60×60	1252	1349	1255	1430	1254	1350	1252	1295
70×70	1714	1823	1720	1823	1718	1834	1715	1848
80×80	2245	2378	2244	2374	2245	2376	2247	2377
90×90	2882	2996	2880	3002	2890	2996	2886	2993
100×100	5195	5283	5197	5285	5194	5284	5195	5283

实验二. 选取有障碍的二维网格地图作比较.

从图 1 和图 2 可以看出, 在障碍物稠密程度相同的情况下, 对于相同大小的地图, 在运行时间效率和节点访问数量上进行对比, 对未进行优化的传统 A*算法, 访问过的网格为 483 个. 但经过加入方向因子优化过的 A*算法, 范围过的网格仅为 191 个.

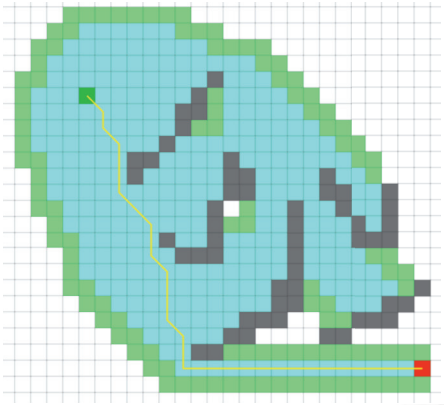

图 1 传统 A*算法最短寻径过程

实验三. 综合各种因素的 A*算法的优化, 显示寻找最短路径时访问的节点数.

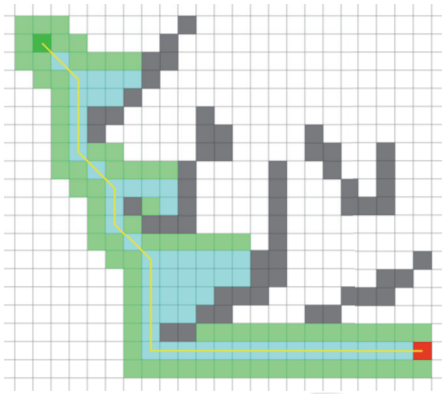

图 2 加入方向因子优化的 A*算法最短寻径过程

从表 2 的数据可以看出, 在障碍物稠密程度相同的情况下, 随着地图网格数的增大, 在运行时间和网格访问的数量上, 优化过的 A*算法要比传统的 A*算法都要少很多, 随着地图的增大, 二者在运行时间效率和网格访问数量上的差距越来越明显, 提升效率约为 50%.

表 2 批量数据比较结果

地图网格数	第 1 次运行结果		第 2 次运行结果		第 3 次运行结果		第 4 次运行结果	
	优化 A*	传统 A*	优化 A*	传统 A*	优化 A*	传统 A*	优化 A*	传统 A*
1600	417	619	422	624	418	616	416	620
2500	646	950	645	951	647	953	752	952
3600	931	1349	932	1430	929	1350	927	1295
4900	1258	1823	1253	1823	1232	1834	1249	1848
6400	1643	2378	1642	2374	1647	2376	1644	2377
8100	1845	2996	1846	3002	1847	2996	1844	2993
10 000	2492	5283	2490	5285	2489	5284	2495	5283
14 400	3289	7146	3287	7151	3289	7143	3291	7145

5 结论

在传统 A*算法搜索策略的基础上, 通过加入方向因子, 以及有效的剪枝和回溯, 部分细节代码优化等优化方案, 进一步地改进 A*算法的启发式搜索策略. 最

后的 A*算法的评估实验证明, 优化方案是有效的, 优化的 A*算法比传统的 A*算法, 较大地减小了算法搜索的规模, 减少了网格访问数量和运行时间.

参考文献

1 詹海波. 人工智能寻路算法在电子游戏中的研究和应用[硕士学位论文]. 武汉: 华中科技大学, 2007. 6–7.

2 黄海威, 邓开发. 改进的 A*算法在游戏地图寻路中的应用. 信息技术, 2015, 39(4): 188–191.

3 周小镜. 基于改进 A*算法的游戏地图寻径的研究[硕士学位论文]. 重庆: 西南大学, 2011. 24–30.

4 Moffat A. An empirical survey of shortest path algorithms. New Zealand Operational Research, 1983, 11(2): 153–163.

5 Yap PKY, Burch N, Holte RC, *et al*. Block A*: Database-driven search with applications in any-angle path-planning. Proceedings of the 25th AAAI Conference on Artificial Intelligence. San Francisco, CA, USA. 2013.

6 邱磊. 基于 A*算法的游戏地图寻路实现及性能比较. 陕西科技大学学报, 2011, 29(6): 89–93.

7 Yap PKY, Burch N, Holte RC, *et al*. Any-angle path planning for computer games. AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment. Stanford, CA, USA. 2011. 201–207.

8 Daniel K, Nash A, Koenig S, *et al*. Theta*: Any-angle path planning on grids. Journal of Artificial Intelligence Research, 2010, (39): 533–579.

9 Björnsson Y, Enzenberger M, Holte RC, *et al*. Fringe search: Beating A* at pathfinding on game maps. IEEE Symposium on Computational Intelligence and Games. Colchester, Britain. 2005. 125–132.