基于A*算法的游戏地图最短路径搜索

崔振兴,顾治华

(武汉理工大学 计算机科学与技术学院,湖北 武汉 430063)

摘 要: 介绍了常用的搜索算法思想, 重点剖析了采用启发式A*算法实现大地图与复杂地形的最短路径搜索,在对估价函数特性进行分析的基础上,讨论了它的几个一般构造原则,并简要介绍一些常用的启发函数。

关键词:最短路径;Dijkstra算法;Best-First-Search;A*算法;启发函数

中图分类号:TP312 文献标识码:A 文章编号:1672-7800(2007)09-0145-03

0 前言

搜索分为无提示信息搜索(也称盲目搜索)和有提示信息(启发式)搜索[1]。最短路径搜索,就是根据游戏地图中的地形和障碍,寻找一条从起点到终点的最近、最直接的路径的算法。许多著名的游戏均采用了该技术,如帝国时代和圣剑英雄传等。在大多数计算机教材中,路径搜索算法大多建立在数学中图的基础上,即图是由边(edges)连接的一系列点(vertices)的集合。而在瓦片(tiled)游戏地图中,我们可以把地图中的每个瓦片(tile)看作点,并且邻接瓦片间由边(edge)连接。本文中,我们假设使用的都是这种二维栅格(grid)的title游戏地图。

1968年,人们提出了A*算法,该算法结合了像BFS(Best-First-Seareh)的启发式方法和Dijkstra算法的形式化方法。像BFS算法这样的启发式算法通常给你一种估计的方式来解决问题而不能保证你得到最优解。A*算法却能够保证找到最短路径。

1 A*算法

1.1 思想

A*算法把OPEN表中的节点按估价函数f(n)=g(n)+h(n)的值从小到大进行排序,对所有的搜索节点n,使h(n)<=h*(n),h*(n)为节点n到终点的实际距离,从而使得找到的路径为最短路径或最优路径。在保证h(n)<=h*(n)的前提下,h(n)的值越大,则启发信息也越大,可以减少搜索过程中扩展的节点数,加快搜索速度。

1.2 算法

(1)把起始节点S放到OPEN有序表中。

(2)如果OPEN表为空,则失败退出,无解。

(3)从OPEN表中选择一个f值最小的节点n。

(4)把n从OPEN表中移出,放入CLOSED表中。

(5)如果n是目标节点,则成功退出,求得一个解;否则跳到(6)步。

(6)扩展节点n,生成其全部后继节点。对于n的每个后继节点m,计算f(m)。①如果m不在OPEN表和CLOSED表中,把它加入OPEN表中,给m加一指向其父节点n的指针,以便找到目标节点时记住解答路径;②如果m已在OPEN表中,则比较刚计算的f(m)新值和该节点m在表中的f(m)旧值,如果f(m)新值较小,表示找到了一条更好的路径,则以此新f(m)值取代表中该节点的f(m)旧值,将节点m的父指针指向当前的n节点,而不是指向它的历史父节点;③如果m在CLOSED表中,则跳过该节点,返回(6)继续扩展其它节点。

(3)跳到步骤(2),继续循环,直到找到解或无解退出。

1.3 数据结构

(1)搜索树结构

根节点为开始节点,父节点为空,在到达目标节点后,从目标节点回溯到根节点,所遍历的节点就是最短路径。

```
typedef struct tree_node *TREE;
struct tree_node {
TREE parent ;// 该节点的父指针
int height ; // 该节点在搜索树中的高度
int tile ;// n(x,y)的位置标号(y*地图高度+x)
}
```

(2)OPEN表,CLOSED表

OPEN表和CLOSED表分别是用于保存未处理和已处理的节点的链表

```
typedef struct node_link *LINK;
struct node_link {
TREE node ;// 节点内容
int f ;// 估价函数
LINK next ;// 下一个节点
}
LINK open,close;
```

(3)估价函数f(x,y)=g(x,y)+h(x,y)

其中f是当前节点(x,y)的估价函数。

作者简介:崔振兴(1983–),男,山东临沂人,硕士,研究方向为算法设计与分析、计算机网络技术;顾治华(1963–),男,湖北武汉人,硕士,武汉理工大学计算机科学与技术学院副教授,研究方向为软件工程、数据库应用技术。

g(x,y)表示从起始节点(start_x,start_y)到当前节点(x,y)的步数,即该节点在搜索树中的高度。h(x,y)是从节点(x,y)到目标节点(end_x,end_y)最短路径的估计代价。

(4)路径的搜索过程可以描述如下:

```
A_starSearch()
{
OPEN=[起始节点];CLOSED=[];
While(OPEN表非空)
{
从OPEN表中取得一个节点S,并将其从OPEN表中删除
if(S是目标节点)
{
求得路径path,并返回路径path;
}
else
{
for(S的每一个子节点Y)
{
根据估价函数计算节点Y的估价值;
if(Y不在OPEN表和CLOSED表中)
{
if(Y的估价值小于OPEN表中的估价值)
更新OPEN表中的估价值及Y的父节点;
}
else
{
if(Y的估价值小于CLOSED表中的估价值)
{
更新CLOSED表中的估价值;
从CLOSED表中移出节点,并放入OPEN表中;
}
}
将节点S插入CLOSED表中;
按估价值将OPEN表中的节点排序;
}//end for
}//end else
}
end while
}end function
```

A*算法其实是在宽度优先搜索的基础上引入了一个估价函数,每次并不是把所有可扩展的结点展开,而是利用估价函数对所有未展开的结点进行估价,从而找出最应该被展开的结点,将其展开,直到找到目标结点为止。A*算法的实现难点在建立一个合适的估价函数,估价函数构造得越准确,则A*搜索的时间越短,然而建立估价函数还没有严格的方法可循,为此,下文针对这一问题进行了讨论。

2 Heuristics 启发式

2.1 启发函数

启发函数h(n)给A*一个从任意节点n到目标节点的最小花费时间的估计值,选择一个好的启发函数是非常重要的,因为启发式函数能够控制A*算法的行为。

(1)极端情况下,如果h(n)=0,则f(n)=g(n)+0=g(n),那么只有g(n)起作用,A*算法就变成了Dijkstra算法,这可以确保找到最短路径。

(2)如果h(n)总是小于或等于从n到目标的实际花费,那么A*算法就能确保找到最短路径。h(n)的值越小,A*算法要扩展的节点就越多,算法就越慢。

(3)如果h(n)的值恰好等于从节点n到目标节点的确切时间花费,那么A*算法就仅扩展最优路径上的节点而不会扩展其它任何节点,使A*算法十分迅速。虽然不能总是遇到这种情况,但我们可以在某些特殊的情况下使h(n)更加精确,要知道h(n)提供的信息越精确,A*算法就会越完美。

(4)有时如果h(n)比从节点n到目标节点的实际花费要大,那么A*算法不能保证找到的使最短路径,但是A*算法运行的比较快。

(5)另一种极端情况,如果h(n)相对于g(n)非常大,那么只有h(n)起作用,这时A*算法就近乎于BFS算法。

因此,我们选择A*算法将能以尽可能快的速度获得最短路径。如果h(n)过小,我们仍能得到最短路径,但速度会很慢;而如果h(n)过大,我们就得不到最短路径,但A*算法会运行得很快。在游戏中,A*算法的这种属性非常有用。例如在某些情况下,你会发现你想得到的是一条相对好的路径而不是一条最佳路径。因此,在g(n)和h(n)之间做出权衡,我们就可以改变A*算法的性能。

从技术上讲,如果h(n)>h*(n),这时该算法只能称为A算法,而不能称之为A*算法。然而,在本文中,作者仍称之为A*算法,因为它们的实现相同,而且游戏编程界并未严格区分它们。

2.2 构造启发函数时应考虑的因素

(1)速度与精确度的权衡

A*算法这种基于启发函数来改变其行为的能力在游戏中经常运用。速度和精度之间的运行速度折衷可以提高游戏有效性。在大多数游戏中,我们并没有必要找出两点之间的最佳路径,你所需要的可以是接近最佳路径的路径。你在速度和精度之间的权衡取决于你想在游戏中做什么,或者你计算机的速度。

假如游戏中有两种地形,平地和山脉,其路径的时间花费分别是1和4,那么,A*算法沿着平地搜索的距离将会是山脉的4倍。这是因为可能存在这样一条路径,沿着平地前进从而绕过山脉。你可以加快A*算法的搜索速度,一种方法是:使用2作为地图中两空间的启发距离,A*算法比较4和2,就不会像比较4和1那么糟糕了;另一种做法是:A*算法将山脉地形的运动时间花费设定为3来代替实际的4以减少算法在山脉周围搜索的次数。2种方法都放弃了最理想的路径而得到了更快的搜索速度。

速度和精确度之间的选择不一定是静态的。你可以根据CPU的速度,路径搜索的时间片数,地图上的单元数量,每个单元的重要程度,游戏的难度级别,或者其它的因素做出动态的选择。做出动态选择的一种方法是:创建一个启发函数,假设穿过一个栅格空间的最小时间花费是1,构建如下的花费函数:

$$g'(n)=1+\text{alpha}*(g(n)-1)$$

若alpha=0,则修正的花费函数g'(n)恒等于1。在这种情况下,地形的花费就完全被忽略了,地图上所有的栅格只有可通/不可通。若alpha=1,则g'(n)=g(n),g(n)为原始的花费函数,这时就会完全获得A*算法的优点。可以设置alpha为0到1间的任意值。

速度和精确度间的选择不一定必须是全局的,可以根据地图中某些区域的需

要动态的选择。比如，在当前位置附近选择一条更好的路径要比速度重要，这时我们就更看重精确度而不是速度。或许在地图中某些安全的区域找到一条最短的路径并不是十分重要，而相反，在不安全的地方，比如穿过敌人的阵地时，安全、快速才是最重要的。即要以最快的速度找到路径，而不是最短路径。

(2) 度量

A*算法要计算 f(n)=g(n)+h(n)。为了使两个值相加，它们必须具有相同的度量。若 g(n)以小时来度量，而 h(n)的单位是米，A*算法就认为 g 或 h 要么太大要么太小，那么算法就不能得到最佳路径或者算法运行较慢。

(3) 精确启发

如果启发函数的值正好等于最优路径上的实际距离，那么 A*算法就会扩展非常少的节点。这是最理想的情况。这里称为精确启发。如何构建理想的启发函数呢？①预先计算的精确启发。一种构建精确启发的方法就是预先计算每对节点间的最短长度。虽然这对大多数游戏地图都不适合，但仍有办法可以接近精确启发。②线性精确启发。在一些特殊的情况下，我们无需任何预先计算就可以做出精确的启发，如果我们有一张没有任何障碍和特殊地形的地图，那么从起始点到目标点的最短路径就是这两点之间的直线距离。

如果使用的是非常简单的启发函数，比如该启发函数根本不知道地图中的障碍，那么它就应该达到精确启发的效果。如果不能，那么可能就是度量或者选择的启发类型有问题。

5.3 栅格地图的启发函数

在栅格地图上，有一些常用的启发函数。

(1) 曼哈顿距离 (Manhattan distance)

曼哈顿距离是准启发函数。找出花费函数和从一个栅格移动到临近栅格的最小花费 D。因此，游戏中的启发函数应该是曼哈顿距离的 D 倍。即 h(n) = D×(abs (n.x-goal.x)+abs(n.y-goal.y))。注意：应该使用和花费函数匹配的度量。

(2) 对角线距离 (Diagonal distance)

从 (1，1) 到 (5，5) 的曼哈顿距离是 8×D。而如果允许在地图上沿对角线运动，那么我们可以沿对角线移动，因此启发距离为 4×D。即我们可以用一个不同的启发函数：h (n)=D*max (abs (n.x-goal.y), abs (n.y-goal.y))。这里假设沿直线和对角线移动的花费都是 D。

若沿对角线移动的花费不是 D，而是像 D2 = sqrt(2)×D 之类的，那么上面的启发函数就应改为如下形式：

h_diagonal (n)=min (abs (n.x-goal.x), abs (n.y-goal.y))

h_straight (n)=(abs (n.x-goal.x)+abs (n.y-goal.y))

h (n)=D2×h_diagonal (n)+D×(h_straight (n)-2×h_diagonal(n)))

其中，h_diagonal(n)=你可以沿对角线移动的步数；h_straight(n)=曼哈顿距离。然后把二者结合起来，且所有沿对角线的花费是 D2，剩下的沿直线的花费为 D。

(3) 欧几里德距离 (Euclidean distance)

若地图上的单元能够以任意角度移动，而不是只能沿格网方向，那我们就应该用两点间的直线距离：h(n)=D×sqrt((n.x - goal.x)2+(n.y-goal.y)2)

然而，在这种情况下，如果直接使用 A*算法就会遇到问题，因为花费函数 g 和启发函数 h 可能不匹配。因为欧几里德距离比曼哈顿距离或对角线距离要小，使用 A*算法你能够得到最短路径，但是算法将会花更长的时间。

当然，除了上述常用的启发函数，针对不同的问题，存在其它的启发函数，这就要求在构造启发函数时，要考虑问题本身的特性，充分利用问题求解过程中的启发性信息，构造尽可能好的启发函数，加速问题的求解过程。

6 结束语

A*算法是人工智能的经典算法，不仅可用于游戏地图的最短路径搜索，也常用于棋牌类复杂状态空间的最优状态推演，在估价函数 f=g+h 中，需要对不同的游戏定义合适的启发函数。该文剖析了采用启发式 A*算法来实现大地图与复杂地形的最短路径搜索，在对启发函数特性进行分析的基础上，讨论了构造启发函数时考虑的因素及原则，并简要介绍一些常用的启发函数。

参考文献：

[1]蔡自兴.人工智能及其应用[M].北京:清华大学出版社,1996.

[2]严尉敏,吴伟民.数据结构(C 语言版) [M].北京:清华大学出版社,2002.

[3]邢传鼎,杨家明,任庆生. 人工智能原理与应用[M]. 上海:东华大学出版社,2005.

[4]钟敏.A 算法估价函数的特性分析[J].武汉工程职业技术学院学报,2006,(2).

[5]Jung, Soojung. A genetic algorithm for the vehicle routing problem with time-dependent travel times[D]. University of Maryland College Park. 2000.

(责任编辑：王　钊)

Shortest Paths Searching in Game Map Based on A* Algorithm

CUI Zhen-xing, GU Zhi-hua

(Wuhan University of Technology, Wuhan 430063, China)

Abstract:In this paper, the characteristics of heuristic function are analyzed, and several principles on its design are discussed. In addition, some heuristic functions are introduced.

Key words: shortest paths;Dijkstra;Best-First-Search,A* Star algorithm;heuristic function