关于无人系统中最短路径算法的研究

刘帅 2020212267 2020219110 班

摘　要：最短路径规划问题是解决无人系统运行过程的关键问题，近百年来，出现诸多针对不同适用情景的路径规划算法。本文针对几个著名算法：广度优先搜索，Dijkstra 算法，Bellman-Ford 算法，SPFA 算法，A*算法，进行原理分析，并分析其对于无人系统的不同应用场景，并重点针对单源最短路径进行分析。

关键词：无人系统；最短路径；图论；单源最短路径；算法优化

引言：为了简化问题，很多情况下都要采用建立数学模型的方法，针对最短路径算法的应用问题，则需将实际问题转化为图论模型问题进行求解。针对有项加权图为点集边集二元图的特殊性质，则可进行加权等有关操作。

定义有向加权图 G=（V，E），定义加权函数 W：E→R 为从边到全职的映射，故路径 p=（v1,v2,…,vk）的带权路径之和为 w（p）=∑w（vi-1，vi）定义顶点 u 到顶点 v 间最短路径的权值为 g（u,v）={ min{w（p）：u→v}存在通路；　∞不存在通路}生活场景中的路径多可以抽象为有向有权图，但期中边权不仅仅是距离，还可指时间，代价，罚款，安全率等。最短路径规划问题的目的就是要找到 vi 到 vj 权重最小的边，即完成 g（u，v）最优化子问题的求解。

1 最短路径问题的几种分类

1.1　单源最短路径问题

该类问题的目的是求从某的固定点开始到所有顶点的最短路径问题。

1.2　单汇最短路径问题

“单汇”所指的是顶点确定情况的问题求解。对于求所有点到某个点的最短路径，反向思考则是求从该点出发到其他顶点的最短路径。

1.3 求任意顶点之间最短路径问题

求解该问题的过程实则是求单元最短路径的子问题，最后逐一计算。

2　广度优先算法

2.1　广度优先算法是逐层遍历的算法，多用于解决单源最短路问题。从原点开始，利用 vis 数组记录该结点是否被访问，建立 dist 数组储存节点到源点距离，每次访问结点即将节点入队，当其出队时入队与其相关联的点，即分层操作的实质，最终知道队列 q 空，bfs 过程结束。由于每个点仅遍历一次，故整体复杂度为 O(V+E)。

```
#define maxn 100
#include<vector>
vector<int>vec;
int vis[maxn];
int dist[maxn];
#include<queue>
using namespace std;
void BFS(){
    queue<int>q;
    q.push(1);
    int count = 0;
    while(!q.empty()){
        int top = q.front();
        q.pop();
        vis[top] = 1;
        dist[++count] = top;
        for (int i = 0; i < vec[top].size(); i++) {
            int v = vec[top][i];
            if (vis[v]) continue;
            q.push(v);
        }
    }
}
```

[1]从实际问题来讲，对于无人汽车而言，每个结点 V 代表收费站，E 代表该结点的收费金额，在收费金额相同的情况瞎可用 BFS 求出收费金额最小的路线。

3 Dijkstra 算法

3.1 简介

Dijkstra 针对有向图与无向图在权值为正条件下的最短路求解，主要思想为：从七点开始，寻找目前到结点距离最短且未被标记的节点，后用其到出发点的距离更新所有其他点到原点的距离（前提设所有无关点到起点距离为 INF），在稠密图（边数大于顶点数）的图中，时间复杂度由于每次更新起点附近节点距离即为 O(n^2)；对于稀疏图，借助优先队列可将复杂度降为 O(MlogN)。

```
30  void dijkstra(int s) {
31      for (int i = 1; i <= maxn; i++) dist[i] = INF;
32      dist[s] = 0;
33      for (int i = 0; i < maxn; i++) {
34          int u = -1;
35          int min = INF;
36          for (int j = 0; i < maxn; j++) {
37              if (vis[j]==0 && dist[j] < min) { u = j; min = dist[j]; }//更新节点
38          }
39          if (u == -1)return;
40          vis[u] = 1;
41          for (int j = 0; j < vec[u].size(); j++) {
42              int v = vec[u][j].v;
43              if (vis[v] == 0 && dist[u] + vec[u][j].dis < dist[v]) {
44                  dist[v] = dist[u] + vec[u][j].dis;
45              }
46          }
47      }
48  }
```

3.2 应用

借助公交网络，更有助于理解 Dijkstra 算法的更新过程：

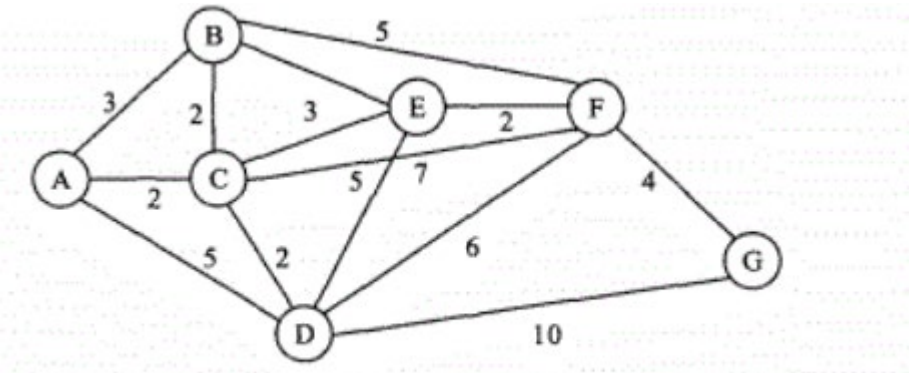


图 3　公交网络

Dijkstra 迭代步骤

路径集合	B	C	D	E	F	G	MIN	Sub V
A	3	2	5	∞	∞	∞	2	C
A,C	3	2	4	5	9	∞	3	B
A,C,B	3	2	4	5	8	∞	4	D
A,C,B,D	3	2	4	5	8	14	5	E
A,C,B,D,E	3	2	4	5	7	12	7	F
A,C,B,D,E,F	3	2	4	5	7	11	11	G
A,C,B,D,E,F,G	3	2	4	5	7	11		

Dijkstra 算法是目前较为常见的无人汽车路径规划方法。

4 Bellman-Ford 算法与 SPFA 算法

4.1 简介

Bellman-Ford 算法用来解决带有负权的最短路径问题，[2]首先引入零环、正环、负环的概念，如果该图系统中存在零环或正环，则不会影响最短路径的求解，若存在负环，则经过无数次负环后最短路径为-INF，产生矛盾，因此可得出结论：最短路径一定是无环结构。因而求解过程需遍历至多 n-1 个结点，时间复杂度为 O（N^2）。而何时图系统中能产生负值呢？以游戏坦克大战为例，在 n*n 的二维数组中，求坦克从 A 经历到 B 的最短用时，其中存在点 P 代表奖励机制，该模型即可化为带有负边权的最短路径问题，其中奖励点的边权为负，即走到该点所耗费的代价小于其减小的路径代价，从而达到全局最优。

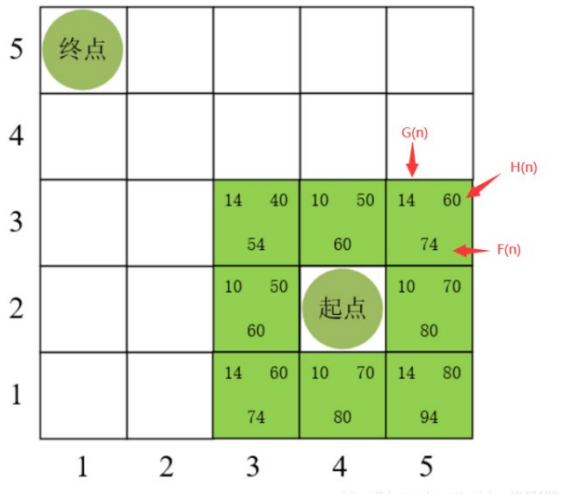
4.2 SPFA 的算法优化

由于 Bellman-Ford 算法的时间复杂度为 O（N*2），在遍历过程中经过了很多不必要的点。注意到当顶点 u 的距离 dist[u]改变时，其临界点的距离才会改变，因此自然想到 FIFO 的思想，即建立队列，对从其出发的所有边进行边权更新
（dist[u]+length[u,v]>d[v]?d[v]:dist[u]+length[u,v]）也是基于队列的优化，同样可采用优先队列进一步优化，原理类似于前文 Dijkstra 的优化过程。

5 A*算法

5.1 简介

A*算法不同于上述的几种算法，其主要针对二维地图求最短路径的算法，通常用于游戏上的人物自动导航（例如做任务游戏的自动寻路功能，实际上就是 A*算法的应用）对于目标点 B 与出发点 A，引入两个距离:D1 出发点周围点到出发点的距离； D2 周围点到终点的距离，这里采用[3]曼哈顿距离 H(n)=(abs(x1 - x2) + abs(y1 - y2))* 10(栅格宽度)。通过计算 D1+D2 的最小值更新路径，从而找到二维地图中的最小值，从人工智能的角度来看，距离算法的迭代实际上就是监督学习。



6 结语

论文主要分析了单源最短路径的有关算法和应用，对于 SPFA 算法，其正确性仍有待商榷，在我后续的学习中，可以尽自己所能证明其正确性；对于最短路径的其他类型问题，文章涉及较少，因此在后续的学习中仍可以自主进行梳理，对于 A*过程，则刻意的与上学期人工智能导论内容进行勾连，期待能在后续学习中巩固监督学习与无监督学习的思想，并将数据结构与算法的内容融入到人工智能专业的学习当中。

参考文献：

[1]张岩 关于最短路算法的一些研究[J] 西安文理学院学报：自然科学版，2014，17（4）：32-35
[2] 《算法竞赛入门经典（第 2 版）》
[3] 图解：A* 算法原理及编程思路讲解_入门初学者必看！(超详细-包会)(一) CSDN 博客
https://blog.csdn.net/weixin_46471339/article/details/110452835?ops_request_misc=%257B%2522request%255Fid%2522%253A%252216251526271678026192349 7%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=162515262716780261923497&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_positive~default-2-110452835.first_rank_v2_pc_rank_v29_2&utm_term=a*%E7%AE%97%E6%B3%95&spm=1018.2226.3001.4187