西安交通大學

论文

A-Star 算法解决重排九宫问题

作者：徐子越

班级：计算机 72 班

学号：2174410455

2019 年 12 月

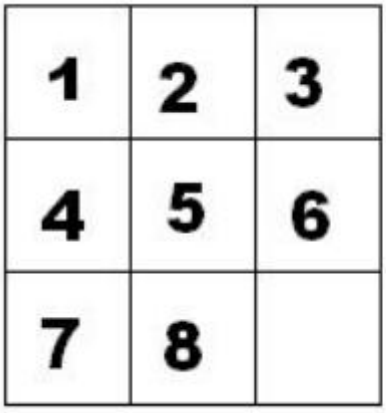
摘　要

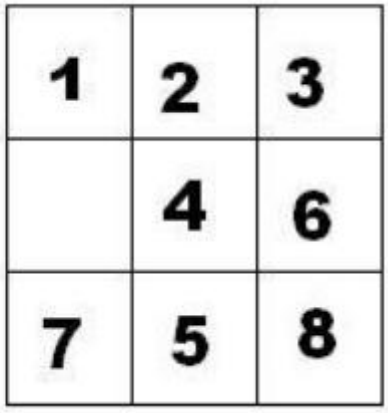
重排九宫问题是人工智能中的著名问题，可以使用多种搜索算法解决。本文通过使用 A-Star 算法对该问题加以分析解决，并给出具体的代码以及运行结果。

关 键 词：人工智能；A-star 算法；搜索算法；CLOSE 表；OPEN 表

1、问题描述

在 3x3 的方格棋盘上放置分别标有数字 1,2,3,4,5,6,7,8 的 8 张牌，初始状态为 S0，目标状态为 Sg，如下图所示。其中左图为初始状态，右图为目标状态。可以使用的算符有空格左移，空格上移，空格右移，空格下移，即它们只允许把位于空格左，上，右，下边的牌移入空格。要求寻找从初始状态到目标状态的路径。





2、算法分析

要用搜索算法解决重排九宫问题，先要确定搜索的结点。作者将每一个九宫图当做一个结点，可以采用一维数组表示。A-Star 算法采用搜索算法的基本框架，建立 OPEN 和 CLOSE 表。特殊的是算法将对 OPEN 表中的结点进行排序，

按照结点的估值函数 f(x)=g(x)+h(x)的值从小到大进行排序。其中 g(x)是对初始结点到当前结点的最小代价，h(x)是从当前结点到目标结点的代价的下界。在此问题中，作者设定 g(x)为从搜索排列树根节点到当前结点 x 的层数差。h(x)为从当前结点 x 到目标结点的不同位的计数和。以初始图为根结点，生成排列树。每个结点的子节点为当前结点对应的九宫图中空白格可以移动的结果。

A-Star 算法如下：

①把初始结点 S_0 放入 OPEN 表

②如果 OPEN 表为空，则搜索失败，退出

③把 OPEN 表中的第一个结点 a 从表中移出放入 CLOSE 表

④考察结点 a 是否为目标节点。如果是则得到了问题的解，退出

⑤若结点 a 不可拓展，则转第二步

⑥拓展结点 a，用估值函数 f(x)计算每个子结点的估计值；并为每个子结点配置指向父结点的指针，把这些子结点都送进 OPEN 表中，然后对 OPEN 表中的所有结点按照估值从小到大排序

⑦转第二步

3、C++代码实现

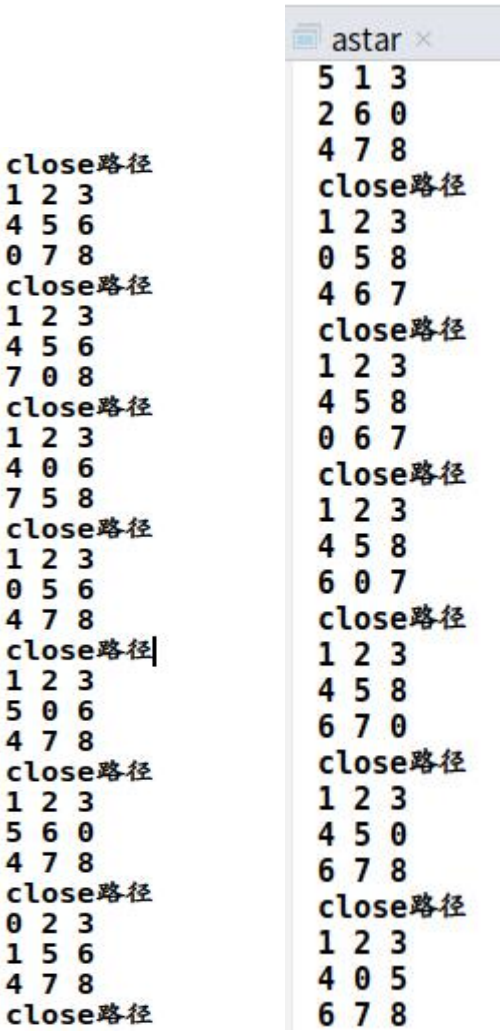
```
#include <iostream>
#include <algorithm>
#include <vector>
using namespace std;
class Node;
bool cmp(Node *a, Node *b);
vector<vector<int>>table =
{{1,3},{0,2,4},{1,5},{0,4,6},{1,3,5,7},{2,4,8},{3,7},{4,6,8},{
5,7}};                          //0 置换表
```

```
vector<Node>open;
vector<Node>close;
static vector<int> target = {1,2,3,4,0,5,6,7,8};

//目标九宫格

static vector<int> init= {1,2,3,4,5,6,0,7,8};

//起始九宫格

class Node{
public:
    vector<int> card;
    int floor;
    int dis;
    int h;
    Node(vector<int> temp, int f){
        this->card = temp;
        this->floor = f;
        int d = 0;
        for(int i = 0;i < 9; i++){
            if(this->card[i]!=target[i])
                d++;
        }
        this->dis = d;
        this->h = this->floor + this->dis;
    }
    static int findzero(Node
*temp){

                //获取0的位置

        for(int i = 0; i < 9; i++)
            if(temp->card[i]==0)
                return i;
            }
    //cmp降序排列
    static bool cmp(const Node& a, const Node& b){
        return a.h > b.h;
    }
    vector<Node>
astar(){

                                    //获得

        int p = findzero(this);
        cout<<"0位置是："<<p<<endl;
        vector<Node>poss;
```

```
        for(int i = 0; i < table[p].size(); i++){
            Node temp(this->card, this->floor+1);
            swap(temp.card[p],temp.card[table[p][i]]);
            temp = Node(temp.card, temp.floor);
            poss.push_back(temp);
        }
        sort(poss.begin(),poss.end(),cmp);
        return poss;
    }
};
int main(){

    cout<<"请输入初始九宫格卡片顺序:"<<endl;

    Node g0(init, 1);
    open.push_back(g0);

    cout<<"起始九宫图为："<<endl;

    for(int i = 0; i<9; i++){
        cout<<open[0].card[i]<<" ";
        if(i%3 == 2)
            cout<<endl;
    }
    cout<<"open"<<open[0].h<<endl;

    //open.back()储存 h 最小的 Node

    int t = 0;
    while(open.back().dis!=0){
        close.push_back(open.back());
        t++;

        cout<<"寻找次数"<<t<<endl;

        vector<Node> temp = open.back().astar();
        open.pop_back();
        open.insert(open.end(),temp.begin(),temp.end());
        sort(open.begin(),open.end(),Node::cmp);

        //检查剔除 open 与 close 的重复部分

        for(int i = 0; i<open.size();i++)
            for(int j = 0; j<close.size(); j++)
                if(open[i].card == close[j].card)
                    open.erase(open.begin()+i);
    }
    close.push_back(open.back());
    cout<<"sucess"<<endl;
```

```
    for(int j = 0; j<close.size(); j++){
        cout<<"close 路径"<<endl;
        for(int i = 0; i<9; i++){
            cout<<close[j].card[i]<<" ";
            if(i%3 == 2)
                cout<<endl;
        }
    }
    return 0;
}
```

4、程序运行结果



分析：从初始结点到当前结点共走了 123 次，如右图。

参考文献

鲍军鹏，张选平.人工智能导论[M].西安交通大学.机械工业出版社