中山大学计算机学院

人工智能

本科生实验报告

课程名称：Artificial Intelligence

学号	23336039	姓名	陈家诚

一、 实验题目

搜索算法

二、 实验内容

1. 算法原理

A. 15-puzzle:

a) 使用了 A*算法，启发函数是每个滑块到标准位置的曼哈顿距离总和。

b) 基本上是在广度优先遍历的基础上加上了一个启发式的权重，然后在结合优先队列（保存状态和解法路径）即可完成本次实验。

c) 注意要检查逆序对数量判断是否有解。

B. TSP：

a) 使用了遗传算法，即适应度计算、选择、杂交和变异四个步骤进行轮换，通过不断地迭代即可得出结果

b) 初始化算法：通过读文件即可实现，记录各城市的 x、y 坐标，然后根据种群容量利用 random 库初始化随机种群 population，种群的每个个体即为一个数组，直接存储路径，即城市的访问顺序

c) 适应度计算：计算按照个体记录的顺序访问得到的总距离，存入 fits 数组

d) 选择：先以适应度的总和减去个体适应度得到权重，根据权重随机抽取个体进入新 population 数组，然后再根据最优保留率保留最优个体。

e) 杂交：由于 population 的个体是没有重复的数组，所以采用对某几个城市的访问顺序进行交换的方法进行。

f) 变异：直接倒置 population 个体的某段的访问顺序，可以调节变异率

2. 关键代码展示

A 星算法：

滑块移动的实现

```
def mymove(puzzle):
    global Q
    global Qs
    global Ql
    t_path=Ql.get()[1]
    if is_finished(puzzle):
        print(t_path)
```

```
        return True
    i=0
    j=0
    k=0
    while i<len(puzzle):
        j=0
        while j<len(puzzle[0]):
            if puzzle[i][j]==0:
                k=1
                break
            j+=1
        if k:
            break
        i+=1
    if(i!=len(puzzle)-1):
        t_path1=copy.deepcopy(t_path)
        puzzle1=copy.deepcopy(puzzle)
        puzzle1[i][j],puzzle1[i+1][j]=puzzle1[i+1][j],puzzle1[i][j]
        if puzzle1 not in Qs:
            f_value=F(puzzle1,len(t_path)+1)
            Q.put((f_value,puzzle1))
            Qs.append(puzzle1)
            t_path1.append(puzzle1[i][j])
            Ql.put((f_value,t_path1))

    if(i!=0):
        t_path2=copy.deepcopy(t_path)
        puzzle2=copy.deepcopy(puzzle)
        puzzle2[i][j],puzzle2[i-1][j]=puzzle2[i-1][j],puzzle2[i][j]
        if puzzle2 not in Qs:
            f_value=F(puzzle2,len(t_path)+1)
            Q.put((f_value,puzzle2))
            Qs.append(puzzle2)
            t_path2.append(puzzle2[i][j])
            Ql.put((f_value,t_path2))

    if(j!=len(puzzle[0])-1):
        t_path3=copy.deepcopy(t_path)
        puzzle3=copy.deepcopy(puzzle)
        puzzle3[i][j],puzzle3[i][j+1]=puzzle[i][j+1],puzzle[i][j]
        if puzzle3 not in Qs:
            f_value=F(puzzle3,len(t_path)+1)
            Q.put((f_value,puzzle3))
            Qs.append(puzzle3)
```

```
            t_path3.append(puzzle3[i][j])
            Ql.put((f_value,t_path3))

    if(j!=0):
        t_path4=copy.deepcopy(t_path)
        puzzle4=copy.deepcopy(puzzle)
        puzzle4[i][j],puzzle4[i][j-1]=puzzle4[i][j-1],puzzle4[i][j]
        if puzzle4 not in Qs:
            f_value=F(puzzle4,len(t_path)+1)
            Q.put((f_value,puzzle4))
            Qs.append(puzzle4)
            t_path4.append(puzzle4[i][j])
            Ql.put((f_value,t_path4))
    return False
```

启发函数的实现

```
def H(puzzle):
    i=0
    c=0
    while i<len(puzzle):
        j=0
        while j<len(puzzle[0]):
            p_value=puzzle[i][j]
            if p_value==0:
                p_value=16
            p_value-=1
            ri=int(p_value/len(puzzle))
            rj=int(p_value%len(puzzle))
            c+=abs(i-ri)+abs(j-rj)
            j+=1
        i+=1
    return c
```

遗传算法：

杂交的实现：

```
def cross(self):#杂交
        #这里采用交换某些地点访问顺序的方法杂交
        i=int(self.population_size/2)
        while i>0:
            indexs=random.sample(range(0,self.n),2)
            indexs.sort()
            k1=indexs[0]
            k2=indexs[1]
            changed_part1=self.population[2*i-1][k1:k2]
            changed_part2=[]
            j=0
```

```
            k=0
            while j<self.n:
                t_value=self.population[2*i-2][j]
                if t_value in changed_part1:
                    changed_part2.append(t_value)
                    self.population[2*i-2][j]=changed_part1[k]
                    k+=1
                j+=1
            self.population[2*i-1][k1:k2]=changed_part2
            i-=1
```

3. 创新点&优化

优化：

a) A 星算法使用了优先队列直接存储路径，避免了递归查找。

b) 遗传算法直接使用了路径作为个体，避免了编码的麻烦。

三、 实验结果及分析

1. 实验结果展示示例（可图可表可文字，尽量可视化）

Astar 算法结果：

```
PS C:\Users\Administrator\aiexp\exp3\puzzle15> python .\main.py
1 2 3 4 5 6 7 8 9 10 11 12 0 13 14 15
[13, 14, 15]
```

```
PS C:\Users\Administrator\aiexp\exp3\puzzle15> python .\main.py
1 2 3 4 5 0 7 8 9 10 11 12 6 13 14 15
[5, 9, 14, 13, 13, 14, 10, 5, 6, 10, 11, 12]
PS C:\Users\Administrator\aiexp\exp3\puzzle15> python .\main.py
14 2 3 4 5 6 7 8 9 10 11 12 0 13 1 15
无解
```

遗传算法结果：

变异率为 0.05

最优保留为 0.1

迭代 100 次

种群数量为 1000

使用 dj38.tsp（ch71009.tsp 运行时间很长）

以下注释右边第一个数为最优保留率，第二个为变异率

20685.82451960881

20574.827984863845

20064.182475007066

20312.81446294179

20544.808866416493

20816.821785551

20056.15364225746

19675.286101765992
20653.013819427924
20256.920038223267 #0.05 0.05

19708.975816311515
20350.452308614065
20371.0855115874
20657.82701054112
20412.795932487355
19586.723922039335
20564.25300798041
20687.27046149635
20943.768856677354
20380.93824087562 #0.1 0.05

20853.998047821686
21565.191746869146
20702.12193617157
20105.39569045267
20981.40379069248
20802.456520682077
20799.19870742243
20869.475224740094
20946.39489397767
20839.279705344652 #0.5 0.05

19760.321941406415
20808.275085927642
20559.3998793555
20674.287048446567
20186.991011607635
21086.48012661254
20806.22499066752
21391.47129385903
20791.901607719497
21087.525326301246 #0.05 0.1

20090.59255358511
21553.73449413136
20023.241454062165
20401.940358964737
21007.779499687837
21263.380247194404

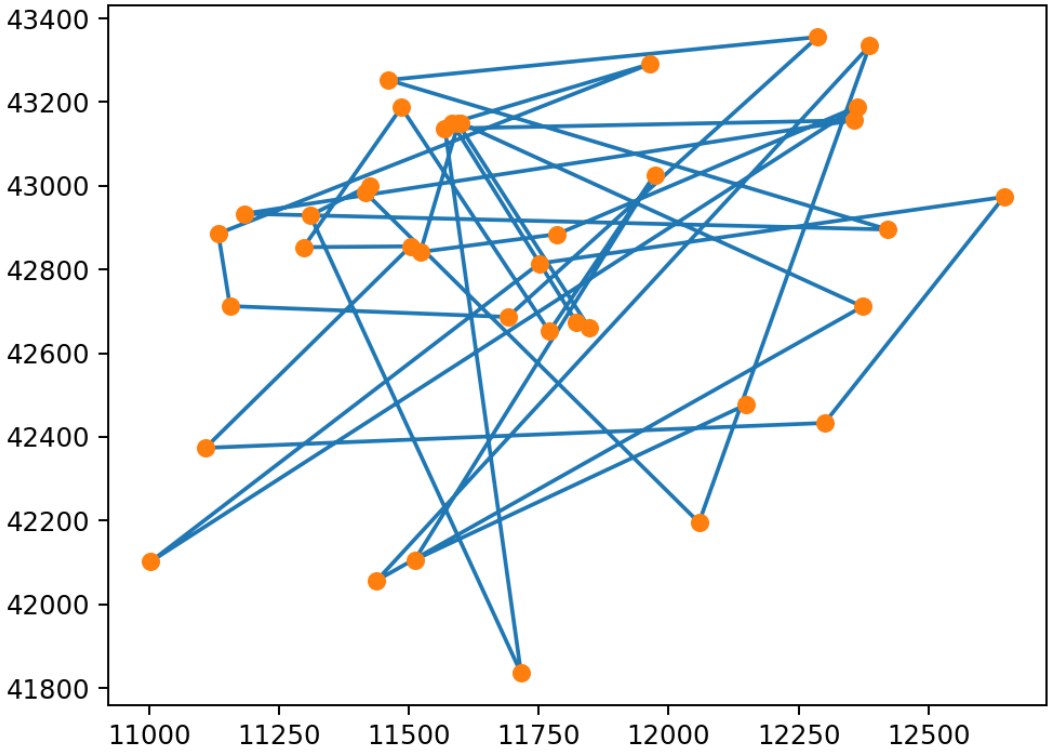
21171.890475632383
20487.851447812227
20222.981455307494
19984.883768158372 #0.1 0.1

21121.220415506257
19984.003965025524
19621.840667313696
21791.689135193767
20058.58182472397
20783.958574532346
20406.826402513238
18452.715179301184
20691.998360083697
20710.619438295238 #0.5 0.1

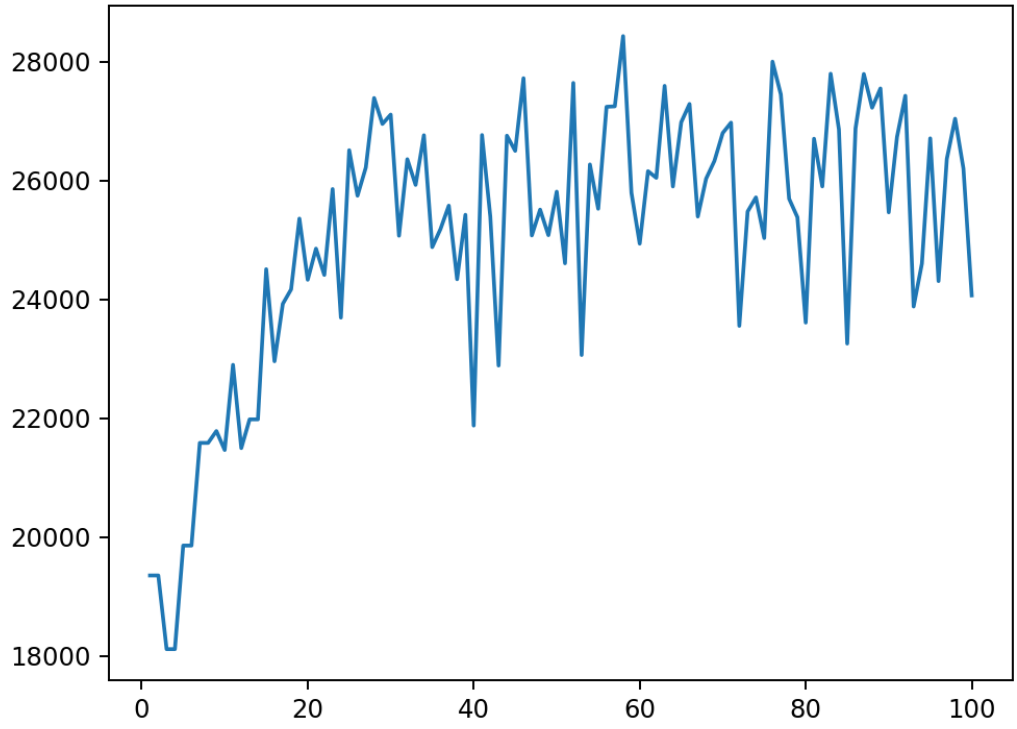
21224.192192903512
19339.835654417762
20216.561914448743
20408.14610068028
20250.790152186946
20712.060979369824
19607.53507808822
20912.072819677804
20772.216063008273
21134.782289051727 #0.05 0.5

18255.691806933675
21388.76702848224
20676.167719054705
20329.54338476788
20774.70579044407
21162.932891938617
21837.10883471937
21107.553111603105
20587.27420056601
21532.814525485897 #0.1 0.5

综上来看，当保留率为 0.5，变异率为 0.1 时效果比较好
但是最小值是在当保留率为 0.1，变异率为 0.5 时运行出来的
最后附上一个 10w 样本 100 次迭代的结果与路径图
适应度：16873.91703960047
路径图：



100 次迭代的最优适应度变化图：



2. 评测指标展示及分析

基本上以最终适应度为指标（然而最优解是 6000 左右）

主要问题一是算力不足，而是算法可能还存在问题，比如运行结果卡在某个数值形成了局部最优。

参考资料

基本思路主要来自网络和课件。代码由自己实现。

[1] 【算法】超详细的遗传算法(Genetic Algorithm)解析_遗传算法中的染色体和个体的概念-CSDN 博客

[2] 15-puzzle - OI Wiki

[3] A 星算法详解(个人认为最详细,最通俗易懂的一个版本)-CSDN 博客