中山大学计算机学院

人工智能

本科生实验报告

（2024 学年春季学期）

课程名称：Artificial Intelligence

教学班级	人工智能(22 信计+系统结构)	专业（方向）	信息与计算科学
学号	22336044	姓名	陈圳煌

一、 实验题目

使用 A*和 IDA*算法求 15-Puzzle 问题的最优解

二、 实验内容

1. 算法原理

A*：A*算法是一致代价搜索算法的升级版本，引入了一个启发式函数 h(n)，表示从当前状态到目标状态的距离或者难易程度，评价函数 f(n)=g(n)+h(n)，其中 g(n)是初始状态到当前状态的步数。综合 h 和 g 能够更好地选择下一步需要扩展的点。

IDA*：IDA*是迭代加深算法的升级版本，不再是迭代加深那样盲目的搜索，而是和 A *算法一样引入了启发式函数，在进行 DFS 扩展点时更具有目的性。

2. 思路

A*：从初始状态开始，每次选取当前状态节点(Open 列表中)的相邻状态节点中估价函数值最小的(h(x)最小)进行扩展，并且对已进行扩展的状态进行剪枝，直到找到最终节点，或者 Open 列表中已经没有待扩展的节点后停止算法。

IDA*:使用递归的方法(在实验分析中会介绍为何不使用栈的方法),从起始状态开始，选择相邻状态中的一个状态进行递归，若当前状态的估价函数大于限制的深度阈值，则进行剪枝，直到找到最终状态或者无解。

3. 伪代码

```
function A_star(matrix,count):
    Open 列表 Closed 列表
    moves 为每次移动的四个方向的步骤
    while Open 列表中有未进行扩展的状态:
        从 Open 列表中取出 f（x）最小的状态
        count=count+1#扩展次数
        #结束条件 1
        if cur_state.matrix 等于 correct_matrix:
```

```
            找到节点
        #结束条件2:
        if count>20000000:
            退出
        idx,idy=get_index(0,cur_state.matrix)
        for moves 中的每个移动步骤:
            idx_=idx+i
            idy_=idy+j
            if 0<=idx_<4 and 0<=idy_<4:
                tmp_matrix[idx][idy],tmp_matrix[idx_][idy_]=tmp_matrix[i
dx_][idy_],tmp_matrix[idx][idy]交换空格和相邻的数字的位置
                判断是否扩展过：是：跳过 否：加入Closed列表

function IDA_star(state,count):
    depth=当前状态的f(x)值
    while 1:
        count+1
        result=IDA_search(state,深度,depth)
        if result等于0：找到结果
        if result等于-1：结束
        更新depth

function IDA_search(state,深度,depth):
    f=g+h(state)
    if f>depth: 更新深度
    if state等于最终状态：结束
    最小代价=曼哈顿函数的极限值
    for 每个子状态:
        递归cost=IDA_search(下一个状态，深度+1，depth)
        if cost<mincost: 更新mincost6
        if cost等于0：找到结果
    返回最小代价
```

4. 关键代码展示（带注释）

优化前：

```
#A*算法
def A_star(matrix,count):
    Open=[]#待扩展列表
    Closed=[]#已扩展列表
    moves=[(0,1),(0,-1),(1,0),(-1,0)]
    root=node(0,h(matrix,correct_matrix),matrix)
    Open.append(root)
    #把扩展列表转化成堆
```

```
    heapq.heapify(Open)
    Closed.append(root.matrix)
    while len(Open)!=0:
        cur_state=heapq.heappop(Open)
        count=count+1
        if cur_state.matrix==correct_matrix:
            find_way(cur_state)
            return count

        idx,idy=get_index(0,cur_state.matrix)
        for i,j in moves:
            idx_=idx+i
            idy_=idy+j
            if 0<=idx_<4 and 0<=idy_<4:
                tmp_matrix=copy.deepcopy(cur_state.matrix)
                tmp_matrix[idx][idy],tmp_matrix[idx_][idy_]=tmp_matrix[i
dx_][idy_],tmp_matrix[idx][idy]
                if find_end(tmp_matrix,Closed):
                    continue
                Closed.append(tmp_matrix)
                next_state=node(cur_state.g+1,h(tmp_matrix,correct_matri
x),tmp_matrix)
                next_state.prev=cur_state
                next_state.changenum=tmp_matrix[idx][idy]
                heapq.heappush(Open,next_state)

    return count
#IDA*算法
def IDA_star(matrix,count):
    moves=[(1,0),(-1,0),(0,1),(0,-1)]
    root=node(0,h(matrix,correct_matrix),matrix)
    Open.append(root)
    depth=root.f
    #把扩展列表转化成堆
    Closed.append(root.matrix)
    while(len(Open)!=0):
        cur_state=Open.pop()
        depth=depth+1
        count=count+1
        if cur_state.matrix==correct_matrix:
            find_way(cur_state)
            return count

        idx,idy=get_index(0,cur_state.matrix)
```

```
        for i,j in moves:
            idx_=idx+i
            idy_=idy+j
            if 0<=idx_<4 and 0<=idy_<4:
                tmp_matrix=copy.deepcopy(cur_state.matrix)
                tmp_matrix[idx][idy],tmp_matrix[idx_][idy_]=tmp_matrix[i
dx_][idy_],tmp_matrix[idx][idy]
                if h(tmp_matrix,correct_matrix)+cur_state.g+1>depth:
                    continue
                else:
                    if find_end(tmp_matrix,Closed):
                        continue
                    Closed.append(tmp_matrix)
                    next_state=node(cur_state.g+1,h(tmp_matrix,correct_m
atrix),tmp_matrix)
                    next_state.prev=cur_state
                    next_state.changenum=tmp_matrix[idx][idy]
                    Open.append(next_state)

    return count
```

优化后：

```
#A*算法
def A_star(matrix,count):
    Open=[]#待扩展列表
    Closed=set()#已扩展列表
    moves=[(0,1),(0,-1),(1,0),(-1,0)]#移动的方向
    root=node(0,h(matrix,correct_matrix),matrix)
    Open.append(root)
    #把扩展列表转化成堆
    heapq.heapify(Open)
    #将当前节点的矩阵转化成字符串对应的哈希值加入已扩展列表
    Closed.add(hash(str(root.matrix)))
    while len(Open)!=0:
        cur_state=heapq.heappop(Open)#取出堆中最小元素进行扩展
        count=count+1#扩展次数
        #结束条件1
        if cur_state.matrix==correct_matrix:
            find_way(cur_state)
            return count
        #结束条件2:
        if count>20000000:
            print("Too many expansion nodes(more than 2000w)!")
            return count
```

```
        #获取0的位置，即空格位
        idx,idy=get_index(0,cur_state.matrix)
        for i,j in moves:
            idx_=idx+i
            idy_=idy+j
            if 0<=idx_<4 and 0<=idy_<4:
                tmp_matrix=copy.deepcopy(cur_state.matrix)
                tmp_matrix[idx][idy],tmp_matrix[idx_][idy_]=tmp_matrix[i
dx_][idy_],tmp_matrix[idx][idy]
                hash_val=hash(str(tmp_matrix))
                #判断当前节点扩展出的节点是否已扩展
                if hash_val in Closed:
                    continue
                Closed.add(hash_val)
                next_state=node(cur_state.g+1,h(tmp_matrix,correct_matri
x),tmp_matrix)
                next_state.prev=cur_state
                next_state.changenum=tmp_matrix[idx][idy]
                #将下一节点入堆
                heapq.heappush(Open,next_state)

    return count

#IDA*算法的递归步骤
def IDA_search(g,depth,count):
    cur_state=Open[-1]#此处的state是矩阵
    count+=1
    #当前节点的f(x)值
    f=g+h(cur_state,correct_matrix)
    #判断是否进行扩展
    if f>depth:
        return f,count
    if cur_state==correct_matrix:
        return 0,count

    mincost=90 #最远的曼哈顿距离不会大于每个都在对角线相反位置，即15*6=90
    moves=[(1,0),(-1,0),(0,1),(0,-1)]
    idx,idy=get_index(0,cur_state)
    new_list=[]
    for i,j in moves:
        idx_=idx+i
        idy_=idy+j
        if 0<=idx_<4 and 0<=idy_<4:
            tmp_matrix=copy.deepcopy(cur_state)
```

```
            tmp_matrix[idx][idy],tmp_matrix[idx_][idy_]=tmp_matrix[idx_]
[idy_],tmp_matrix[idx][idy]
            new_list.append(tmp_matrix)

    for next_state in new_list:
        if hash(str(next_state)) in Closed:
            continue
        Open.append(next_state)
        way.append(next_state[idx][idy])
        Closed.add(hash(str(next_state)))
        t,count=IDA_search(g+1,depth,count)

        if t==0:
            return 0,count
        if t<mincost:
            mincost=t
        #如果没有从当前节点的子节点中找到结果，就回退
        Open.pop()
        way.pop()
        Closed.remove(hash(str(next_state)))

    return mincost,count
#IDA*算法
def IDA_star(matrix,count):
    depth=h(matrix,correct_matrix)
    Open.append(matrix)
    Closed=set()
    Closed.add(hash(str(matrix)))

    while(1):
        result,count=IDA_search(0,depth,count)
        if count>20000000:
            print("Too many expansion nodes(more than 2000w)!")
            return count
        if result==0:
            return count
        if result==-1:
            print('Failed to find solution!')
            return count
        depth=result
```

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）

测试样例:

1.txt
```
1 15 7 10
9 14 4 11
8 5 0 6
13 3 2 12
```

2.txt
```
1 7 8 10
6 9 15 14
13 3 0 4
11 5 12 2
```

3.txt
```
5 6 4 12
11 14 9 1
0 3 8 15
10 7 2 13
```

4.txt
```
14 2 8 1
7 10 4 0
6 15 11 5
9 3 13 12
```

5.txt
```
14 10 6 0
4 9 1 8
2 3 5 11
12 13 7 15
```

6.txt
```
6 10 3 15
14 8 7 11
5 1 0 2
13 12 9 4
```

7.txt
```
11 3 1 7
4 6 8 2
15 9 10 13
14 12 5 0
```

8.txt
```
0 5 15 14
7 9 6 13
1 2 12 10
8 11 4 3
```

优化前：

A*:

```
PS C:\Users\陈圳煌\Desktop\计算机学院\计算机学院（大二下）\人工智能\实验报告\E3\code> & C:/Users/陈圳煌/AppData/Local/M
/Users/陈圳煌/Desktop/计算机学院/计算机学院（大二下）/人工智能/实验报告/E3/code/test(IDA_star).py"
Please enter one filename:1.txt
The num of node expanded are: 38153
A optional solution is: 50
The way of task are:
4 7 15 14 9 8 5 9 14 15 7 4 6 11 4 14 8 5 9 3 2 6 14 8 15 7 10 4 8 15 3 2 6 14 15 10 7 3 2 6 14 15 10 7 3 2 6 10 11 12
Used Time 53.117644 sec
PS C:\Users\陈圳煌\Desktop\计算机学院\计算机学院（大二下）\人工智能\实验报告\E3\code> & C:/Users/陈圳煌/AppData/Local/M
/Users/陈圳煌/Desktop/计算机学院/计算机学院（大二下）/人工智能/实验报告/E3/code/test(IDA_star).py"
Please enter one filename:2.txt
The num of node expanded are: 96340
A optional solution is: 50
The way of task are:
15 8 7 9 8 7 9 8 7 14 4 15 14 9 8 7 3 5 11 13 5 14 12 2 15 12 2 11 14 2 9 8 10 4 8 10 7 3 2 9 10 7 3 2 6 5 9 10 11 15
Used Time 646.844767 sec
```

```
PS C:\Users\陈圳煌\Desktop\计算机学院\计算机学院（大二下）\人工智能\实验报告\E3\code> & C:/Users/陈
/Users/陈圳煌/Desktop/计算机学院/计算机学院（大二下）/人工智能/实验报告/E3/code/test(A_star).py"
Please enter one filename:3.txt
Choose the function to search:
1.曼哈顿
2.未复原方块数
1
The num of node expanded are: 536
A optional solution is: 40
The way of task are:
11 14 9 1 12 4 1 8 2 13 15 12 8 2 3 11 14 9 6 1 2 3 11 7 13 11 7 14 10 13 14 10 9 5 1 2 3 7 11 15
Used Time 0.052075 sec
PS C:\Users\陈圳煌\Desktop\计算机学院\计算机学院（大二下）\人工智能\实验报告\E3\code> & C:/Users/陈
/Users/陈圳煌/Desktop/计算机学院/计算机学院（大二下）/人工智能/实验报告/E3/code/test(A_star).py"
Please enter one filename:4.txt
Choose the function to search:
1.曼哈顿
2.未复原方块数
1
The num of node expanded are: 10424
A optional solution is: 40
The way of task are:
1 8 4 1 5 11 15 3 13 15 3 10 1 5 8 4 2 1 7 14 1 7 5 3 10 6 14 5 7 2 3 7 6 14 9 13 14 10 11 12
Used Time 10.277781 sec
/Users/陈圳煌/Desktop/计算机学院/计算机学院（大二下）/人工智能/实验报告/E3/code/test(IDA_star).py"
```

后四个例子时间过长。

IDA*:

设 depth 为定长 50 时:

```
PS C:\Users\陈圳煌\Desktop\计算机学院\计算机学院（大二下）\人工智能\实验报告\E3\code> & C:/Users/陈圳煌/AppData/Local/M
/Users/陈圳煌/Desktop/计算机学院/计算机学院（大二下）/人工智能/实验报告/E3/code/test(IDA_star).py"
Please enter one filename:1.txt
The num of node expanded are: 38153
A optional solution is: 50
The way of task are:
4 7 15 14 9 8 5 9 14 15 7 4 6 11 4 14 8 5 9 3 2 6 14 8 15 7 10 4 8 15 3 2 6 14 15 10 7 3 2 6 14 15 10 7 3 2 6 10 11 12
Used Time 53.117644 sec
PS C:\Users\陈圳煌\Desktop\计算机学院\计算机学院（大二下）\人工智能\实验报告\E3\code> & C:/Users/陈圳煌/AppData/Local/M
/Users/陈圳煌/Desktop/计算机学院/计算机学院（大二下）/人工智能/实验报告/E3/code/test(IDA_star).py"
Please enter one filename:2.txt
The num of node expanded are: 96340
A optional solution is: 50
The way of task are:
15 8 7 9 8 7 9 8 7 14 4 15 14 9 8 7 3 5 11 13 5 14 12 2 15 12 2 11 14 2 9 8 10 4 8 10 7 3 2 9 10 7 3 2 6 5 9 10 11 15
Used Time 646.844767 sec
```

其余情况均无法得出结果

优化后：

A*：

```
Please enter one filename:1.txt
Choose the function to search:
1.曼哈顿
2.未复原方块数
1
Choose a function to run:
1.A*
2.IDA*
1
A*:
The num of node expanded are: 2779
A optional solution is: 40
The way of task are:
6 11 4 14 5 8 9 5 8 3 2 6 14 8 15 7 10 4 8 15 3 2 6 14 15 10 7 3 2 6 14 15 10 7 3 2 6 10 11 12
Used Time 2.093766 s
```

```
Please enter one filename:2.txt
Choose the function to search:
1.曼哈顿
2.未复原方块数
1
Choose a function to run:
1.A*
2.IDA*
1
A*:
The num of node expanded are: 1232
A optional solution is: 40
The way of task are:
15 14 4 2 12 15 14 8 10 4 8 9 3 5 11 13 5 14 2 12 15 11 14 2 9 10 7 3 2 9 10 7 3 2 6 5 9 10 11 15
Used Time 2.700032 s
```

```
Please enter one filename:3.txt
Choose the function to search:
1.曼哈顿
2.未复原方块数
1
Choose a function to run:
1.A*
2.IDA*
1
A*:
The num of node expanded are: 536
A optional solution is: 40
The way of task are:
11 14 9 1 12 4 1 8 2 13 15 12 8 2 3 11 14 9 6 1 2 3 11 7 13 11 7 14 10 13 14 10 9 5 1 2 3 7 11 15
Used Time 2.058577 s
```

```
Please enter one filename:4.txt
Choose the function to search:
1.曼哈顿
2.未复原方块数
1
Choose a function to run:
1.A*
2.IDA*
1
A*:
The num of node expanded are: 10424
A optional solution is: 40
The way of task are:
1 8 4 1 5 11 15 3 13 15 3 10 1 5 8 4 2 1 7 14 1 7 5 3 10 6 14 5 7 2 3 7 6 14 9 13 14 10 11 12
Used Time 1.765996 s
```

```
Please enter one filename:5.txt
Choose the function to search:
1.曼哈顿
2.未复原方块数
1
Choose a function to run:
1.A*
2.IDA*
1
A*:
The num of node expanded are: 4471407
A optional solution is: 53
The way of task are:
6 10 9 4 14 9 10 1 4 10 1 4 10 3 2 14 9 1 3 2 5 11 8 6 4 3 2 5 14 12 13 14 12 9 5 10 6 8 11 12 10 6 8 11 12 7 15 12 11 8 7 11 12
Used Time 301.598797 s
```

```
Please enter one filename:6.txt
Choose the function to search:
1.曼哈顿
2.未复原方块数
1
Choose a function to run:
1.A*
2.IDA*
1
A*:
The num of node expanded are: 1752215
A optional solution is: 48
The way of task are:
9 12 13 5 1 9 7 11 2 4 12 13 9 7 11 2 15 3 2 15 4 11 15 8 14 1 5 9 13 15 7 14 10 6 1 5 9 13 14 10 6 2 3 4 8 7 11 12
Used Time 139.925585 s
```

```
Please enter one filename:7.txt
Choose the function to search:
1.曼哈顿
2.未复原方块数
1
Choose a function to run:
1.A*
2.IDA*
1
A*:
The num of node expanded are: 18116128
A optional solution is: 56
The way of task are:
5 12 9 10 13 5 12 13 8 2 5 8 10 6 3 1 2 3 4 11 1 2 3 4 2 3 4 5 7 4 3 2 5 10 6 15 11 5 10 6 15 11 14 9 13 15 11 14 9 13 14 10 6 7 8 12
Used Time 3399.642099 s
```

```
Please enter one filename:8.txt
Choose the function to search:
1.曼哈顿
2.未复原方块数
1
Choose a function to run:
1.A*
2.IDA*
1
A*:
Traceback (most recent call last):
  File "c:\Users\陈圳煌\Desktop\计算机学院\
e 150, in <module>
    expand_num=A_star(matrix,expand_num)
               ^^^^^^^^^^^^^^^^^^^^^^^^^
  File "c:\Users\陈圳煌\Desktop\计算机学院\
e 101, in A_star
    Closed.add(hash_val)
MemoryError
```

第八个样例扩展节点过多，导致内存溢出。第五个样例未能得到最优解。

IDA*:

```
Please enter one filename:1.txt
Choose the function to search:
1.曼哈顿
2.未复原方块数
1
Choose a function to run:
1.A*
2.IDA*
2
IDA*:
The num of node expanded are: 39079
A optional solution is: 40
The way of task are:
6 11 4 14 5 8 9 5 8 3 2 6 14 8 15 7 10 4 8 15 3 2 6 14 15 10 7 3 2 6 14 15 10 7 3 2 6 10 11 12
Used Time 1.186014 s
```

```
Please enter one filename:2.txt
Choose the function to search:
1.曼哈顿
2.未复原方块数
1
Choose a function to run:
1.A*
2.IDA*
2
IDA*:
The num of node expanded are: 10207
A optional solution is: 40
The way of task are:
15 14 4 2 12 15 14 8 10 4 8 9 3 5 11 13 5 14 2 12 15 11 14 2 9 10 7 3 2 9 10 7 3 2 6 5 9 10 11 15
Used Time 0.864883 s
```

```
Please enter one filename:3.txt
Choose the function to search:
1.曼哈顿
2.未复原方块数
1
Choose a function to run:
1.A*
2.IDA*
2
IDA*:
The num of node expanded are: 5579
A optional solution is: 40
The way of task are:
11 14 9 1 12 4 1 8 2 13 15 12 8 2 3 11 14 9 6 1 2 3 11 7 13 11 7 14 10 13 14 10 9 5 1 2 3 7 11 15
Used Time 1.133243 s
```

```
Please enter one filename:4.txt
Choose the function to search:
1.曼哈顿
2.未复原方块数
1
Choose a function to run:
1.A*
2.IDA*
2
IDA*:
The num of node expanded are: 211321
A optional solution is: 40
The way of task are:
1 8 4 1 5 11 15 3 13 15 3 10 1 5 8 4 2 1 5 3 10 5 7 14 1 2 3 7 5 6 14 5 6 14 9 13 14 10 11 12
Used Time 6.048894 s
```

```
Please enter one filename:5.txt
Choose the function to search:
1.曼哈顿
2.未复原方块数
1
Choose a function to run:
1.A*
2.IDA*
2
IDA*:
The num of node expanded are: 14132568
A optional solution is: 49
The way of task are:
6 10 9 4 14 9 4 1 10 4 1 3 2 14 9 1 3 2 13 12 14 13 5 11 8 6 4 3 2 5 12 7 11 12 7 14 13 9 5 10 6 8 12 7 10 6 7 11 15
Used Time 338.849635 s
```

```
Please enter one filename:6.txt
Choose the function to search:
1.曼哈顿
2.未复原方块数
1
Choose a function to run:
1.A*
2.IDA*
2
IDA*:
The num of node expanded are: 19396384
A optional solution is: 48
The way of task are:
9 12 13 5 1 9 7 11 2 4 12 13 9 7 11 2 15 3 2 15 4 11 15 8 14 1 5 9 13 15 7 14 10 6 1 5 9 13 14 10 6 2 3 4 8 7 11 12
Used Time 463.477439 s
```

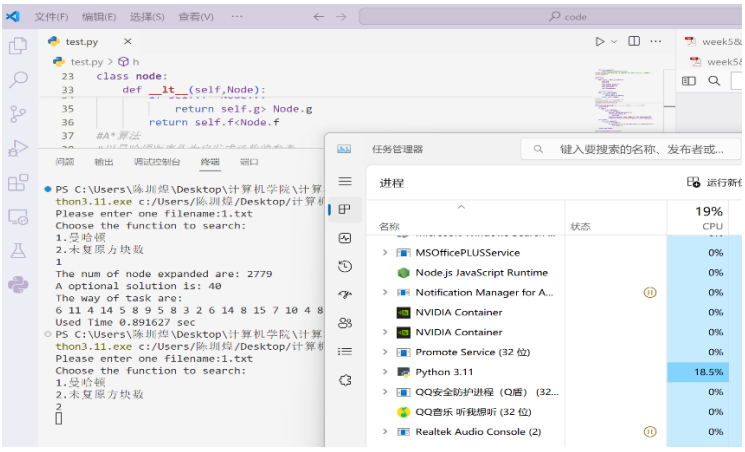
```
Choose the function to search:
1.曼哈顿
2.未复原方块数
1
Choose a function to run:
1.A*
2.IDA*
2
IDA*:
Too many expansion nodes(more than 2000w)!
The num of node expanded are: 96447947
A optional solution is: 0
The way of task are:

Used Time 2328.530933 s
```

第 7.8 个样例扩展节点过多，无法得到最终结果。

2. 评测指标展示及分析（机器学习实验必须有此项，其它可分析运行时间等）

在这次实验中，首先先对比了启发式函数使用曼哈顿距离和未复原方块数时所使用的时间：



可以看出使用未复原方块数作为启发式函数时明显比曼哈顿慢(图中使用未复原方块数作为启发式函数不能得出结果)，因此选择曼哈顿距离作为启发式函数计算方法。

在 A*算法未进行优化前使用列表进行存储矩阵的数据结构，发现对于例 4 以及之后的例子，会因为扩展节点太多而严重影响速度，后续优化后使用哈希表，将矩阵转换成字符串并用哈希值存储在 set()类型中，在扩展规模较大时速度明显提高。

在 IDA*算法中，刚开始使用栈的方式进行 DFS 扩展，但是未找出不能扩展的原因，修改后使用递归算法，较少步骤的例子能够找出正确答案。

对比实验结果，可以得出在需要扩展较少状态的测试中，使用 IDA*算法可以更快找到结果，但是如果需要扩展的状态增多，则使用 A*的效率更高。

四、 参考资料

[1] 【人工智能】A*算法和 IDA*算法求解 15-puzzle 问题（大量优化，能优化的基本都优化了）_十五数码问题 a*-CSDN 博客

[2] Python heapq 自定义比较器 - 知乎 (zhihu.com)

[3] 【路径规划】全局路径规划算法——A*算法（含 python 实现 | c++实现）_a*算法路径规划-CSDN 博客

[4] Python 中栈的概念和使用_python 中栈的用法-CSDN 博客

[5] 人工智能（第三版）.pdf

[6] week5&6 Astar & IDAstar.pdf

[7] HeuSearch23.pdf