成　绩	
评卷人	

研究生	李雪
学　号	2015363039

武漢紡織大學

研 究 生 课 程 论 文

论文题目	启发式搜索技术
完成时间	2020 年 10 月 24 日
课程名称	人工智能
专　　业	电子信息
年　　级	2020 级

武汉纺织大学研究生处制

一、 引言

人工智能诞生于20世纪中期，曾经历两起两落的重要历程。人工智能技术作为21世纪最前沿技术之一，其重大突破必将影响新一轮产业革命，目前它已在医学、教育、研究等领域得到了深远应用。

目前，人工智能面临的问题越来越复杂，其中以非结构化问题居多，以往盲目搜索需要搜索所有节点消耗了大量时间，这种弊病将会严重限制搜索能力，究其原因在于顾及了所有可能性，即一个一个盲目搜索。针对该问题，人们需要运用有知识的生成器避免走一些明显不可能搜索到正确答案的路径，即启发式搜索。启发式搜索已成为实际中求解智能规划问题的重要工具，特别是不确定性规划问题。近年来，放松规划在图规划问题中的探究、基于单值变量的启发式研究等均推动着对启发式搜索的探索。用启发式搜索思维构建问题解成为一种常用的思考方式，比如最大权独立集问题、普遍共享骑行问题等。启发式搜索结合模糊逻辑、频谱频率分配等领域技术更是加快了众多领域搜索发展的进程。归根究底，人们还是希望搜索路径沿着自己认为有希望的方向前进，这样就可以大大减少搜索时间。

A*算法的优点在绝大多数的情况下,在性能方面都远远优与DFS和BFS。算法的主要运行性能,取决于估价函数f的选取。然而由于算法本身的特点，根据估价函数找到的解路径不一定是最优解路径，此外A*算法在路径规划过程中所遍历的点过多，在其搜索过程中存在计算量庞大、内存占用严重等缺点。本文首先追溯研究起点，再从源头到8数码问题对启发式搜索及其启发能力等进行探讨。

二、 相关工作

2.1 启发式搜索算法

启发式搜索，即一种运用启发先验知识或者信息引导搜索方向的方法。为了评价启发式搜索的质量和效率，本文中给出如下定义：

搜索代价函数:F（p）=G（p）+H（p）

指算法在一次启发式搜索过程中，从搜索起点S（Start-point）到达目标点D（Destination）所耗费的代价（如距离等）。其中，G（p）表示从S出发到达某一中间节点p（point）的代价；H（p）表示某一中间节点p到达

D 的代价。

其中，当搜索已在节点 p 时，需要决定接下来的搜索路径，然而当前节点不知道接下来路径的真实值 H（p），于是需要评估得到 H*（p）。如果 H*（p）的值与真实值存在较大偏差，可能引导节点向相对错误的方向搜索。当 H*（p）=H（p）=0 时，即 F（p）=G（p），那么该搜索策略执行一致代价搜索，这种方法每次都会优先选择当前距离最短的路径前进，以最少距离为目标进行节点扩展。这种方法也会抛弃一些不可能得到最优解的节点，以此达到缩短搜索路径距离的目标。

2.2 分支定界法改进方法

2.2.1 使用低估值的分支定界法

易知上述简单纯粹的一致代价搜索对 H（p）处理上存在缺陷，故采用对 H（p）低估值的方法改进分支定界法，即 H*（p）=underestimate（H（p））。显然，这种启发也是可以被接受的，同时比没有启发搜索能力的分支定界更强。

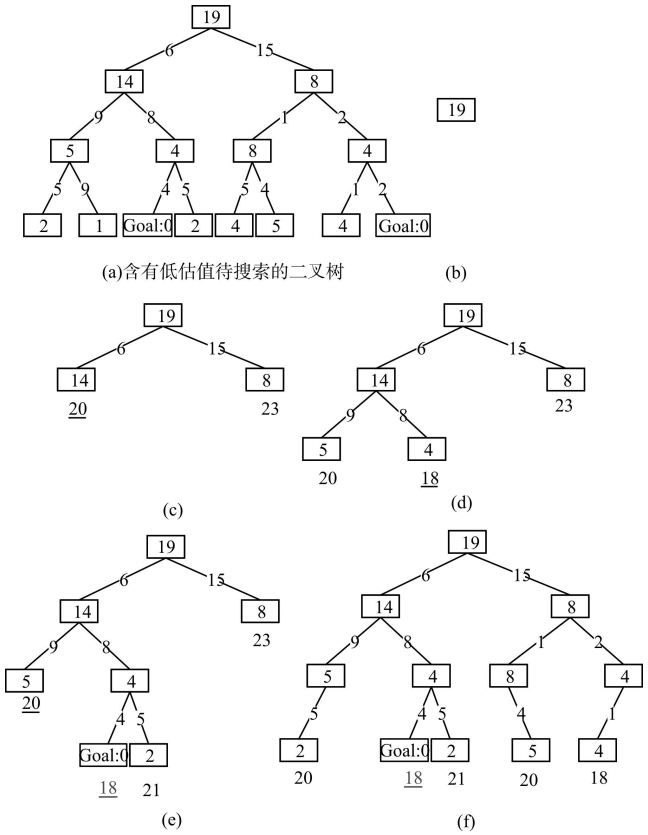

图 2.1 低估值的分支定界法

如图 2.1 所示，设矩形内的值表示该节点到目的地 Goal 的低估计值。从

根节点开始，发现不是目标节点，则扩展该根节点，如图 2.1（c）所示，得到两个评估函数的值 20（6+14）、23（15+8），选择较小者继续扩展，直到找到了一条到达目标节点的路径，如图 2.1（e）所示，之后继续搜索其它距离可能更短的路径，如图 2.1（f）所示，搜索完所有可能达到最短路径的节点。

采用低估值的方法有效提高了搜索质量，更加符合实际情况，但是就其搜索速度而论并没有明显改观。

2.2.2 基于最短路径的分支定界法

由现实生活经验可知，如果两条或多条路径到达同一节点，只需要存储距离消费最小的那条路径的距离即可。通过一个抽象处理后的实例对原理加以说明。若要求从 S 点城市前往 D 点城市，以下说明存储最短路径的分支定界法，如图 2.2（a）-图 2.2（f）所示。

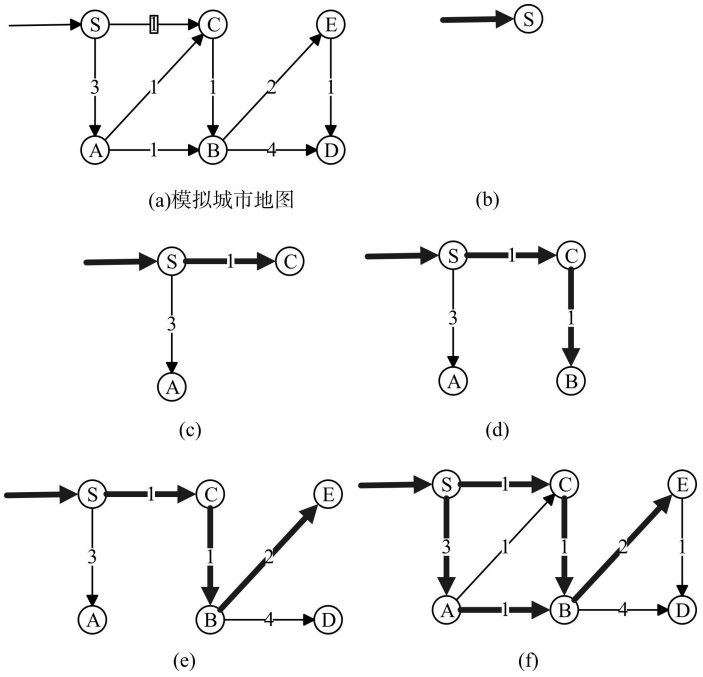

图 2.2 基于最优路径的分支定界法

图 2.2 基于最优路径的分支定界法从 S 点出发，面临 A、C 两点选择，由于 C 点距离更短则选择 C 点，如图 2.2（c）所示。到达 C 点后只能前往 B 点，此时距离 S 点距离为 2，如图 2.2（d）。同理继续前往 E 点，如图 2.2（e），此时距离 S 点距离为 4。接下来扩展距离比 4 更小的路径，即 S→A→B（其实还有另一种走法 S→A→C，基于后经过优先级更高的原则选择 B 点），

此段距离到达 B 时距离 S 点为 3，此时是第二次访问 B 点，于是依据最短路径原则选择到达 B 点最短的距离 2，即保证存储了最短路径 S→A→B。这类似于动态规划，要记录下已经访问的节点，下次继续访问相同节点时，只需要在已经被访问的节点中查找到达某点的最小花费取出来使用即可。虽然这种方法仅仅对到达同一节点的情况进行了优化，但也已在许多相关路径问题解决中见到它的缩影，如快递物流优选路径问题、城市路径规划问题等。

三、 算法描述（伪代码）

3.1 队列实现 A*搜索算法策略

(1)建立一个队列，计算初始结点的估价函数，并将初始结点入队，设置队列头和尾指针；

(2)取出队列头(队列头指针所指)的结点，如果该结点是目标结点，则输出路径，程序结束。否则对结点进行扩展；

(3)检查扩展出的新结点是否与队列中的结点重复，若与不能再扩展的结点重复，则将它抛弃，若新结点与待扩展的结点重复，则比较两个结点的估价函数中 g 的大小，保留较小 g 值的结点。跳至 V；

(4)如果扩展出的新结点与队列中的结点不重复，则按照其估价函数厂的大小将它插入队列中的头结点之后待扩展结点的适当位置，使它们按从小到大的顺序排列，最后更新队列尾指针；

(5)如果队列头的结点还可以扩展，直接返回步骤(2)，否则将队列头指针指向下一结点，再返回步骤(2)。

队列实现 A*搜索算法伪代码如下：

```
Input:
    An 8 puzzle as a start state
    came_from is a DICT, initialized with {}
Output:
    return: GOAL or None
    came_from is changed
def Astar(puzzle8, came_from):
    frontier = PriorityQueue()
```

```
cost_so_far = {}
frontier.enqueue(puzzle8, 0)
cost_so_far[puzzle8] = puzzle8.cost
came_from[puzzle8] = None
while not frontier.is_empty():
    puzzle8 = frontier.dequeue()
    if puzzle8 is a goal:
        return puzzle8
    else:
        for newpuzzle8 in all neighbors of puzzle8:
            new_cost = newpuzzle8.cost
            if newpuzzle8 not in cost_so_far or
            new_cost < cost_so_far[newpuzzle8]:
                cost_so_far[newpuzzle8] = new_cost
                priority = new_cost + newpuzzle8.heuristics
                frontier.put(newpuzzle8, priority)
                came_from[newpuzzle8] = puzzle8
return None
```

3.2 八数码游戏的启发式搜索技术算法策略(OPEN-CLOSE 表实现）

(1)把起始节点 S 放到 OPEN 表中，计算 f(S)。

(2)如果 OPEN 表是个空表，则无解，失败退出；否则继续。

(3)从 OPEN 表中选择一个 f 值最小的节点作为代扩展节点 i。如果有几个节点同时最小，当其中有一个目标结点时，选择此目标结点，否则就选择其中任一个节点作为带扩展节点 i。

(4)把节点 i 从 OPEN 表中移出放入 CLOSE 表扩展节点表中。

(5)如果 n 是目标节点，问题得解，退出。否则继续。

(6)扩展节点 i，生成其全部后继节点。对于 i 的每一个后继节点 j。

1)计算 f(j)。

2)如果 j 既不在 OPEN 表中，又不在 CLOSED 表中，则用估价函数 f

把它加入 OPEN 表相应位置。从 i 加一个指向其父辈节点 i 的指针，以便于反向追踪解的路径。

3)如果 j 已在 OPEN 表(处理与父节点的关系)或 CLOSED 表(处理与父节点和子节点的关系)中，则比较新的 f(j)值和前面计算出来的 f(J)值。如果新的 f(J)值较小。则

①用新的值取代旧的值。

②改变指针，从 j 指向 i，删除原来的父辈节点指针。

③如果节点 j 在 CLOSED 表中，则把它移回 OPEN 表(意味着：扔掉原来的子节点)。

(7)转向②。

八数码游戏的启发式搜索技术算法流程图如图 3.1 所示。

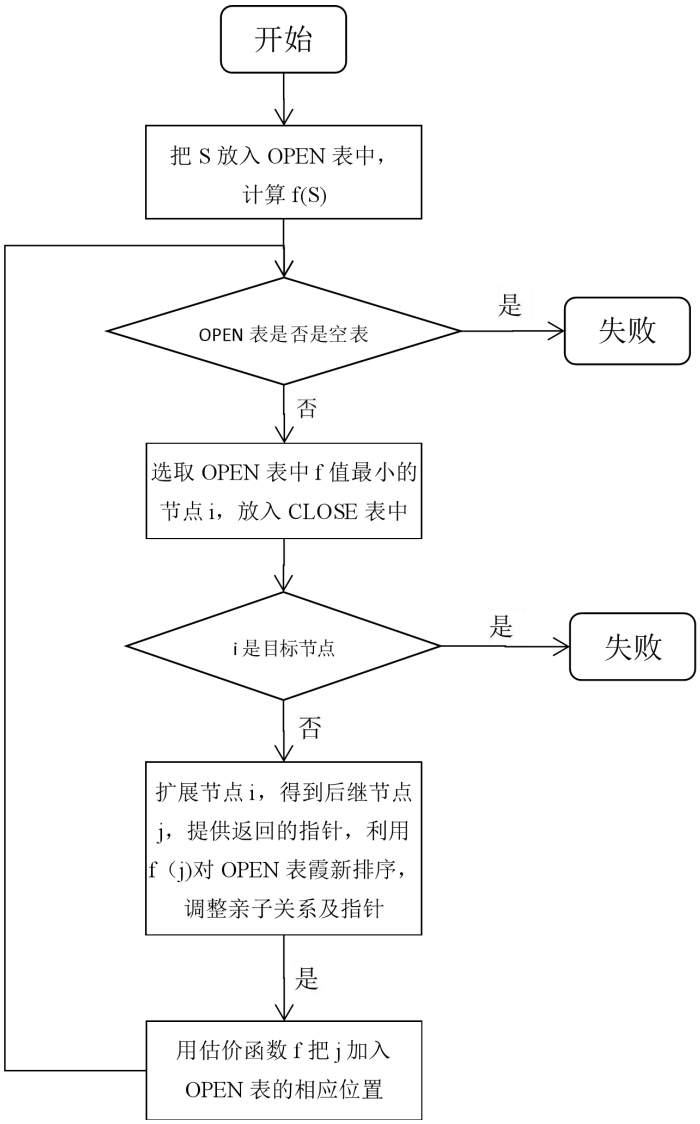

图 3.1 八数码游戏的启发式搜索技术

四、 算法验证（结果与分析）

4.1 八数码 A*搜索运行结果

八数码问题即在一个大小为 3×3 的九宫格上，放置 8 块编号为 1～8 的木块，九宫格中有一个空格，周围(上下左右)的木块可以和空格交换位置。对于问题，给定一个初始状态(如图 4.1 初始状态)，目标状态(如图 4.2 目标状态)是期望达到 1～8 顺序排列的序列，并且空格在左上角，问题的实质就是寻找一个合法的移动序列。

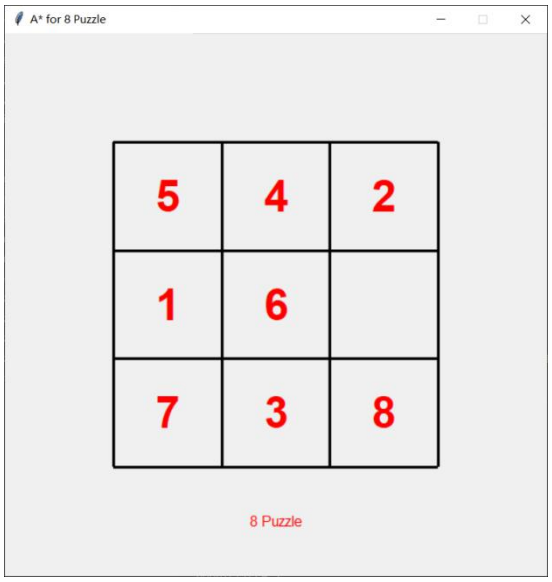

图 4.1 八数码初始状态图

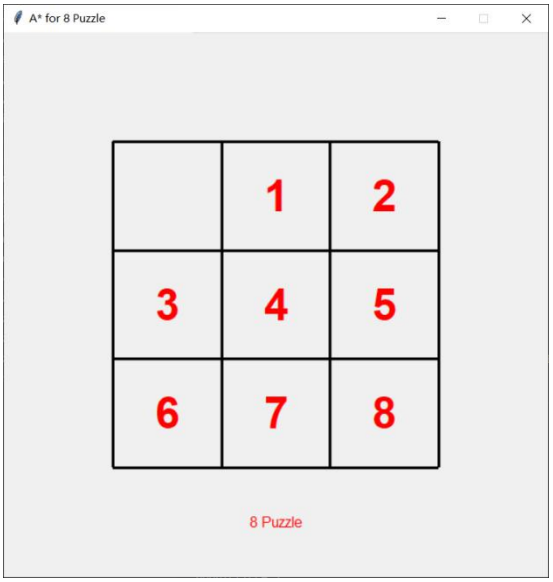

图 4.2 八数码目标状态图

4.2 八数码运行结果分析

下面用经典的三数码问题说明 A*算法，假设采用的低估值为曼哈顿距离，其中用到的算子（简单理解，算子就是每一步的操作，具体一点也可理解为每一步的步骤）是空格上下左右 4 种方向的移动，具体实例如图 4.3 所示。

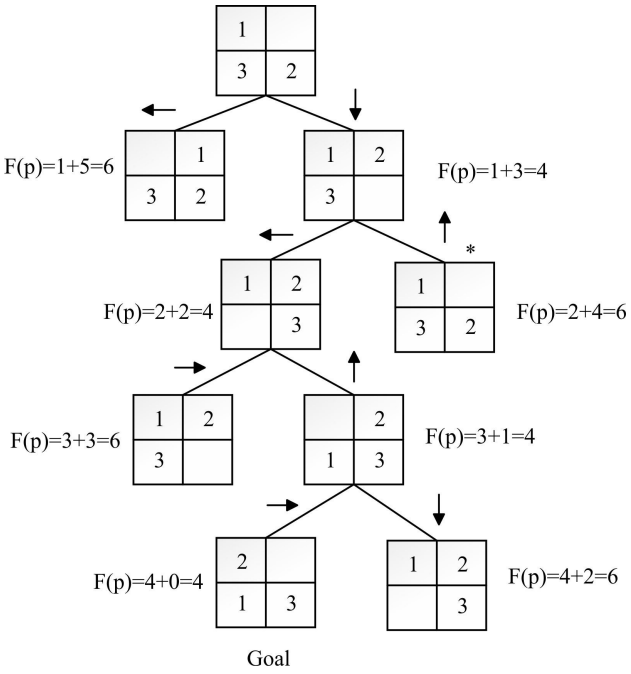

图 4.3 三数码问题

在图 4.3 中，标注有*号的三数码块 F（p）=2+4=6 的原因说明：*号三数码块距离起点完成了两个算子操作，由此 G（p）=2；与 Goal 相比，数字 1 至少向下移动 1 步可以到达 Goal，同理数字 2、3 分别是 2 步和 1 步，因此步数相加为 4 步，即 H*（p）=4。另外，*号数码块与起点数码块状态相同，因此通过比较存储最短距离对路径进行优化。由此还可以观察到，用曼哈顿距离作为搜索低估值时，有时收敛会更加快速。

五、 算法应用

随着经济日益发展以及生产管理水平的进步，高度自动化、智能化的 AGV 应用愈加广泛，柔性制造系统、智慧仓储、自动化码头、智能停车场都是 AGV 常见的应用场景。对 AGV 调度而言，最重要的内容之一就是路径规划。AGV 的路径规划是指选取从任务起始点到目标点的一条路线，使一定目标（时间、距离、能耗等）达到最优化，并且避免与已知障碍物的碰撞。为 AGV 选择有效的路径，可以提高物流效率，降低运输成本。因此，对 AGV 路径规划算法进行研究具有重要意义。

在点点间运输问题的路径规划中，常见的算法有 Dijkstra 算法、Floyd 算法、人工势场法等。A*算法同样作为常见的点点间路径规划算法，与 Dijkstra 算法和 Floyd 算法相比具有更强的启发式信息，在最短路径的搜索

效率上存在一定优势。A*算法是一种基于全局的最短路径搜索算法，能够较好地避免人工势场法频繁出现的局部最优问题。A*算法已经广泛应用于多种场景，包括室内机器人的路径规划、无人船的路径规划和电子游戏中的无碰撞检测等。但 A*算法在实际应用中存在着遍历节点多、搜索过程中计算量庞大等问题，在大规模环境下不断调用 A*算法会占据大量内存资源。

六、 结论与展望

本章学习的 A*算法的基本思想与广度优先算法相同，但是在扩展出一个结点后，要计算它的估价函数，并根据估价函数对待扩展的结点排序，从而保证每次扩展的结点都是估价函数最小的结点。从代价函数的观点看，可以将之前学过的的各种搜索算法统一起来：不同的 f(n)决定了不同的搜索策略。如果 f(n)=g(n)=depth(n)，其中 h(n)≡0，那么该搜索策略执行深度优先搜索；如果 f(n)=g(n)，其中 h(n)≡0，那么该搜索策略执行一致代价搜索，即一致代价搜索也是最佳优先搜索和 A*搜索算法的特例；如果在一致代价搜索中，进一步让 f(n)= g(n)=depth(n)，则该策略执行宽度优先搜索，即宽度优先搜索也是一致代价搜索的特例。此外，若只使用启发函数 h(n)，即 f(n)= h(n)，g(n)≡0，那么该搜索策略将执行贪心最佳优先搜索。

通过基本搜索技术和启发式搜索技术的学习，得出以下结论：①DFS 的解路径长度趋于搜索深度界限，搜索效率受深度影响很大，并且搜索结点冗余多、速度慢；②BFS 找到的一定是最优解，但是在算法效率上，虽然比 DFS 好，却远远不如 A*算法，同时 BFS 在搜索深度较深时，产生的冗余结点较多；③A*算法在效率上相对最优，时间和空间上都比 DFS 和 BFS 更优，但缺点是，找到的解不一定是最优解。

如今启发式搜索已在众多领域得到了落实，比如路径规划、智能机器人等，但是在今后智能规划等任务中，仍然需要理论和实践的创新突破，例如更加精确的搜索代价函数、机器及时对当前现状进行动态更新的启发能力、尽量减少时间或空间消耗等。可以预见，有了更加高效且相对低成本的搜索策略，将对许多领域起到巨大推动作用。

七、 参考文献

[1] 李鹏,周海,闵慧.人工智能中启发式搜索研究综述[J].软件导刊,2020,19(06):35-38.

[2] 孔继利,张鹏坤,刘晓平.双向搜索机制的改进 A*算法研究[J/OL].计算机工程与应用:1-12[2020-10-21].

[3] 蔡自兴，徐光佑．人工智能及其应用[M]．北京：清华大学出版社，2011：33—94.

[4] 杜小勤。《人工智能》课程系列，Part I: Python 程序设计基础，2018/06/13。

[5] 杜小勤。《人工智能》课程系列,Part II: Python 算法基础,2018/07/31。

[6] 杜小勤。《人工智能》课程系列,Chapter 4:启发式搜索技术,2018/10/08。