人工智能基础实验报告

搜索算法

罗晏宸

PB17000297

2020.5.17

数码问题

本次实验过程中，首先使用了基础的 A^* 搜索算法以及迭代 A^* 搜索算法对问题进行求解。随后通过启发式函数$h(x)$的优化，进一步得到效果较好的算法实现。

A^*

实验采用的算法伪代码与教材保持一致，其中为降低遍历优先队列的成本，不在插入时结点在队列中是否重复做判断，而是在出队时对于连续重复结点进行处理，如下

```
    frontier is the priority queue ordered by f = g + h
    explored is a set
    frontier.push(start)
    while frontier is not empty
        node <- frontier.pop()
        while node.State = frontier.top
            frontier.pop()
        if node pass the Goal-Test
            FINISH
        for each action of node
            child <- node + action
            // if child.State in frontier but with higher g
            //     replace the node with child
            // else
                frontier.push(child)
            if child.State not in explored
                insert node into explored
    return FAILURE
```

算法的时间和空间复杂度均为$O(b^d)$，其中b是结点的后继数，在此问题中b不超过 6，d是解所在深度，并与启发式函数有关。

IDA^*

实验采用的算法是用递归形式实现的迭代加深A^*算法，其伪代码如下

```
GraphSearch(start, goal)
    limit <- h(start)
    path.push(start)
    while limit != infinity
        t <- RecursiveSearch(path)
        if t = 0
            return path
```

```
        limit <- t
    return FAILURE

RecursiveSearch(path)
    path.pop(node)
    f <- g(node) + h(node)
    if (f > limit)
        return f
    if node pass the Goal-Test
        FINISH
    min = infinity
    for each action of node
        child <- node + action
        if child not in path
            path.push(child)
            if child.State not in explored
                insert node into explored
            t <- RecursiveSearch(path)
            if t = 0
                return 0
            if t < min
                min = t
            path.pop()
    return min
```

算法的时间复杂度是$O(b^d)$，空间复杂度仅为$O(m)$，其中b是结点的后继数，在此问题中b不超过6，d是解所在深度，并与启发式函数有关，m是可能的最大深度。

启发式函数

Manhattan 距离

实验首先采用简单的 Manhattan 距离，其可采纳性在教材中已有说明，其可以看作原问题的松弛问题需要移动的步数，对应的松弛问题为：假定不管想要移动的块的相邻位置是否为空，都进行移动，因此是可采纳的。

考虑线性冲突的 Manhattan 距离

尽管 Manhattan 距离是可采纳的，并在部分数据上也有不错的表现，但仍然有可改进的空间。线性冲突(Linear conflict)是指对于在一条直线（同一行或同一列）的两个数码，若其目标位置均在这条直线上，且其中一个在另一个归位的最短路径（这一路径一定是线性的）上成为阻碍，则称发生了一起“线性冲突”。考虑线性冲突的 Manhattan 距离是指在 Manhattan 距离上增加 2 倍（为解决一个由线性冲突引起的阻碍，往往需要其中一个数码移开该直线，随后再移回）的线性冲突数。这显然比简单的 Manhattan 距离包含了更多的信息，且并不会做出更高的估计，因为一起线性冲突的解决不可能产生数码的交叉，详细的证明参考

> 1 > 2

在报告中不再赘述。

实验结果

参考 `README` 中的内容，实验程序自带计时，对于三组输入数据，有如下的结果

	A^*	IDA^*
input1	24, 运行时间：0.108s	24, 运行时间：0.027s
input2	12, 运行时间：0.012s	12, 运行时间：0s
input3	\	57, 运行时间：47min

其中 A^* 在求解第 3 组数据上需要过多内存，在本地主机上难以完成完整搜索，为完成实验，使用了付费的服务器，考虑到各种客观原因，仅对IDA^*做了第三组数据的测试。**已求出的解都是最优解。**

具体行动序列的输出参见 `output` 文件夹下的对应文件，其中三组数据的结果均采用 IDA^* 算法所得结果，若使用A^*算法则前两组数据可能会有行动顺序上的不同，且第三组数据可能不能在合理的时间和内存中完成求解。

数独问题

本次实验中，首先实现了简单的前向检验回溯搜索算法，随后使用 MRV 和度启发式对其进行了优化，详细如下

- 前向检验：每次进行尝试赋值后进行的回溯，均重新计算变量可取值
- MRV：每次选择变量时选择可取值最小的变量
- 度启发式：每次选择具有最多约束的变量（约束位置空位最多的变量）

实验结果

由于实验代码采用条件编译，考虑到代码可读性，未优化的代码仍采取了前向检验，详细可见 `README` 中的内容，使用 MRV 和度启发式优化前后运行结果如下

运行时间	前向检验	前向检验 + MRV + 度启发式
input1	0.002s	0.007s
input2	0.06s	0.013s
input3	13.87s	0.548s

可以看到，采用前向检验+MRV+度启发式优化后，效果较好，并且随着数独空位增加，优化效果越明显。

具体数独的输出参见 `output` 文件夹下的对应文件，其中三组数据的结果均采用前向检验+MRV+度启发式优化后所得结果，若使用$未经优化的程序则数据可能略有不同，因为数独的解并不一定唯一。

思考题

可以通过爬山算法、模拟退火算法或是遗传算法等算法来解决。

- 爬山算法和模拟退火算法：对于每个不为解的状态，为重复出现的数字赋能，假设每个在冲突位置重复重复出现的数字都具有相同势能p，显然数独的解势能为 0，是能量最低的状态。可能出现的问题是，数独棋盘上的势能计算可能不能得到很好的倾斜程度（或导数）的结果。
- 遗传算法：考虑棋盘的线性展开（长为 81 的链）作为遗传基因，通过交叉与变易，对亲代、子代与变异基因进行选择，选择的依据是片段内重复碱基对（数字）最少者。可能出现的问题是，对于 9 个数字的值域，随机性的变异往往导致重复数字的产生，退化率过高