第四章　搜 索 技 术-2

——超越经典的搜索

主讲人　　赵曼

中国地质大学　计算机学院人工智能系

目 录

中国地质大学 China University of Geosciences

目　录

Zhaoman, AI,
Dept。of AI, CS, CUG

1.1 超越经典搜索

- 上一章，我们讨论了一个单一类别的问题，其解决方案是具有如下特点的一系列动作：
  - 可观测(observable)
  - 确定性(deterministic)
  - 已知环境(known environment）
- 本章我们将讨论：
  - 局部搜索、优化算法、CSP问题和联机搜索。

1.2 经典搜索

- 具有如下特点：
  - 搜索算法被设成系统地探索问题的空间。
  - 该系统性是由以下方法得到的：

    在内存中保持一条或多条路径，并且在沿着该路径的每个点上记录哪些已被探索过。
  - 目标被找到时，该路径也就构成问题的一个解。
- 然而，
  - 在许多问题中，到达目标的路径是无关紧要的 。

1.3 局部搜索

局部搜索算法和最优化问题

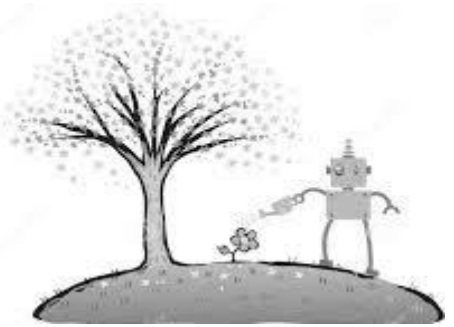
例子: *n*-皇后问题

- 在$n \times n$棋盘上摆放n个皇后，使得任意两个皇后互相都攻击不到。

- 我们关心最后的棋局，而不是皇后加入的先后次序。

1.3 局部搜索

- 局部搜索是一种不同类型的算法，具有如下特点：
  - 它不介意什么路径；
  - 局部搜索算法使用一个当前节点（而不是多条路径），并且通常仅移动到该节点相邻的节点；
  - 通常，搜索后不保留该路径 。
- 优点：
  - 使用很少的内存；
  - 在大的或无限（连续）状态空间中，能发现合理的解 。



局部搜索的优化问题

- 在很多的最优化问题中，不需要知道到达目标的路径，目标状态本身才是问题的解。
  - 除了寻找目标之外，局部搜索算法对解决纯优化问题也很有效。
  - 优化的目的是根据一个目标函数找到其最好的状态。
  - 但是许多优化问题并不适合采用前面介绍的搜索算法。

  例如，达尔文进化论可以被看作是**试图优化**，但对于这一问题，即没有“目标测试”、也没有“路径代价”。

  （将“**适者生存**”作为自然界的优化函数。当然这个问题不好做“目标测试”和计算“路径耗散”）

- 对于这类问题，可以采用**局部搜索**算法，通过不断改进当前状态，直到它满足约束为止。

局部搜索的应用

- 许多应用问题是与路径无关的，目标状态本身就是解。例如：
  - integrated-circuit design 集成电路设计
  - factory-floor layout 工厂车间布局
  - job-shop scheduling 车间作业调度
  - automatic programming 自动程序设计
  - Telecommunications network optimization 通讯网络优化
  - vehicle routing 车辆路由
  - portfolio management 投资组合管理

目 录

2.1 爬山法搜索

- 爬山搜索是一种属于局部搜索家族的数学优化方法。
- 爬山法基本思想：
  - 是一个沿着值增加的方向持续移动的简单循环过程;
  - 当到达一个山峰时就停止。
  - 爬山法不根据当前状态考虑计划未来后继节点。就像雾中登山一样。
  - 当有多个继节点时，爬山法选取最优后继节点集中的一个。
- 特点：大多数基本的局部搜索算法都不保持一棵搜索树。

爬山搜索算法

“就像失忆的人在浓雾中攀登珠峰”

```
function HILL-CLIMBING(problem) returns a state that is a local maximum
    persistent: current, a node
                neighbor, a node
    current← MAKE-NODE(problem.INITIAL-STATE)
    loop do
      neighbor←a successor of current
      if neighbor.VALUE≤ current.VALUE then return current.STATE
       current←neighbor
```

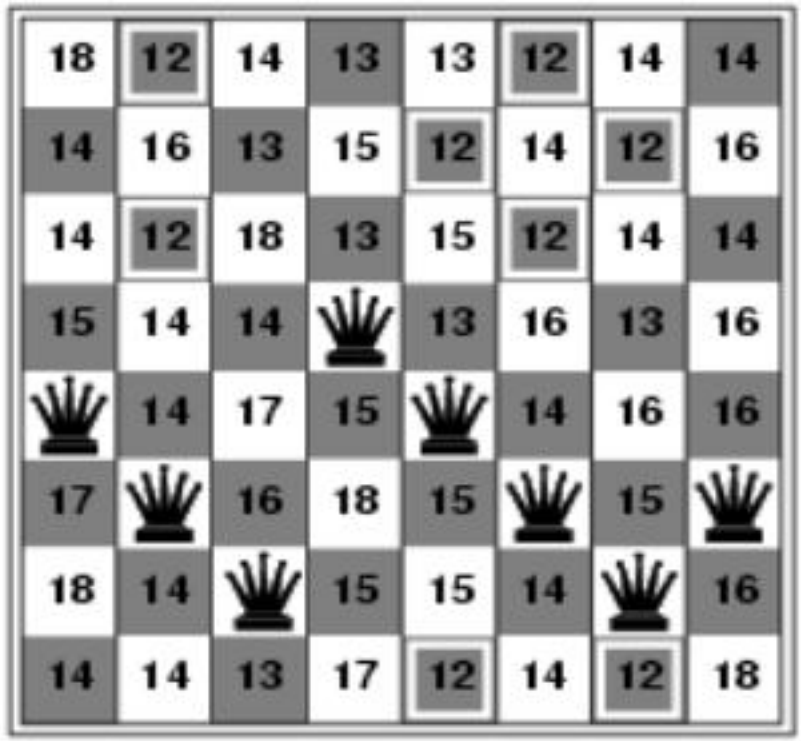
最陡爬坡版（*Steepest-ascent version*）：当前节点每一步都用最佳邻接点（具有最高值的邻接点）替换，否则到达一个“峰值”。

爬山法搜索算法：

- 爬山搜索算法是最基本的局部搜索方法。
- 它常常会朝着一个解快速地进展，因为通常很容易改善一个不良状态。
- 爬山法也往往被称为局部贪婪搜索，因为它只顾寻找一个好的邻接点的状态，而不提前思考下一步该去哪儿。
- 尽管贪婪被认为是“七宗罪”之一，但是贪婪算法往往表现的相当好。

问：局部贪婪搜索同前面讲到过贪婪算法的异同？

爬山搜索: 8-皇后问题



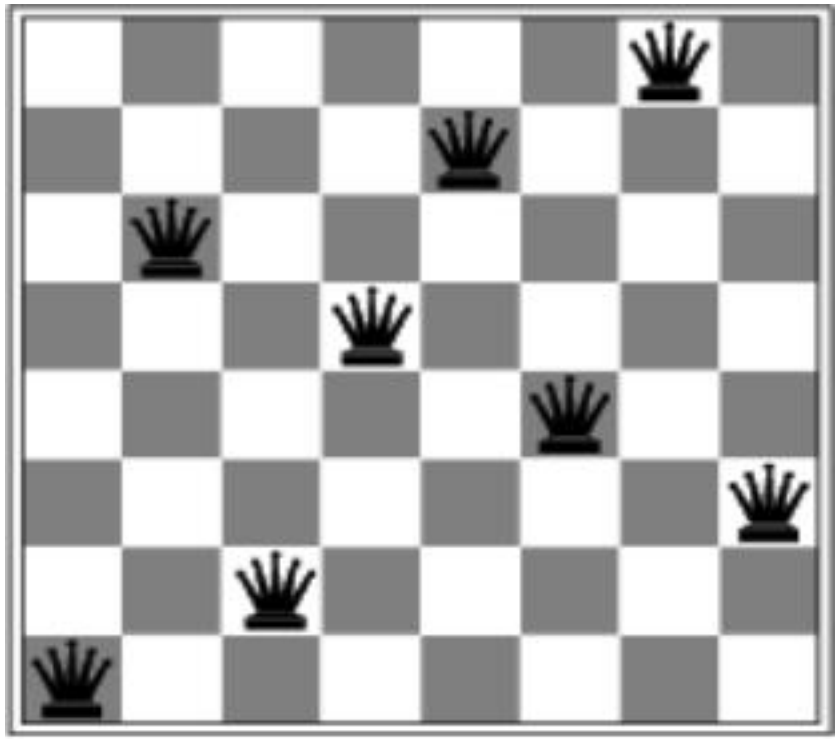
- **问题形式化：** 完全状态形式化。
- **后继函数：** 移动一个皇后到同列另一个方格中的所有可能状态。

  7*8=56个后继状态
- **启发函数：** 可以互相攻击的皇后对数（直接或间接）。该函数的全局最小值是0.

  h = 互相攻击到的皇后对数

  如图这个状态：$h = 17$

爬山搜索: 8-皇后问题



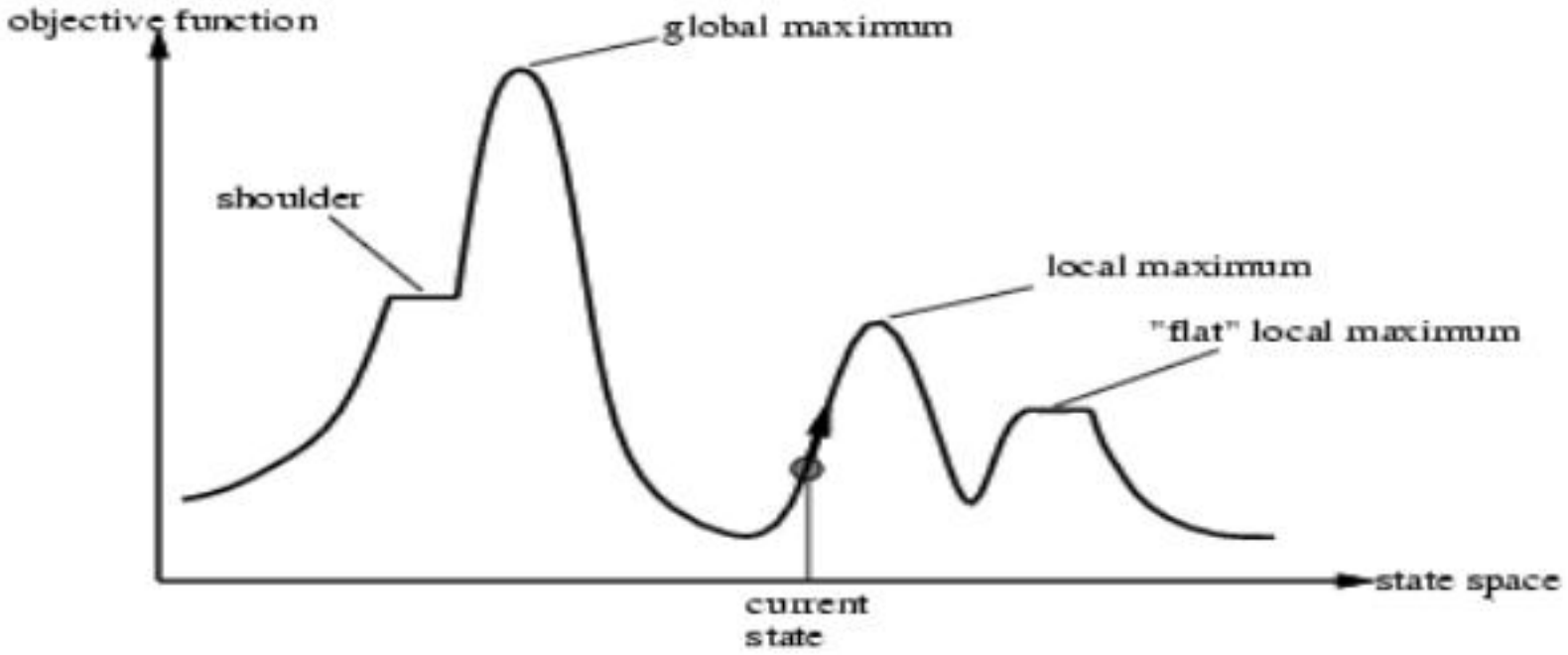
- 一个局部最小值点（ h = 1）
- 依照爬山法该问题求解以失败结束。

统计结果:

8皇后问题86%都会卡住，但算法速度很快成功的最优解平均步数是4。被卡住的平均步数是3。

爬山搜索

- 几个概念：目标函数、全局最优、局部最优、山肩、平原……



- 依赖于初始状态，容易陷于局部最优。

爬山法的弱点：

如下三种情况经常被困：

- 局部极大值（Local maxima）

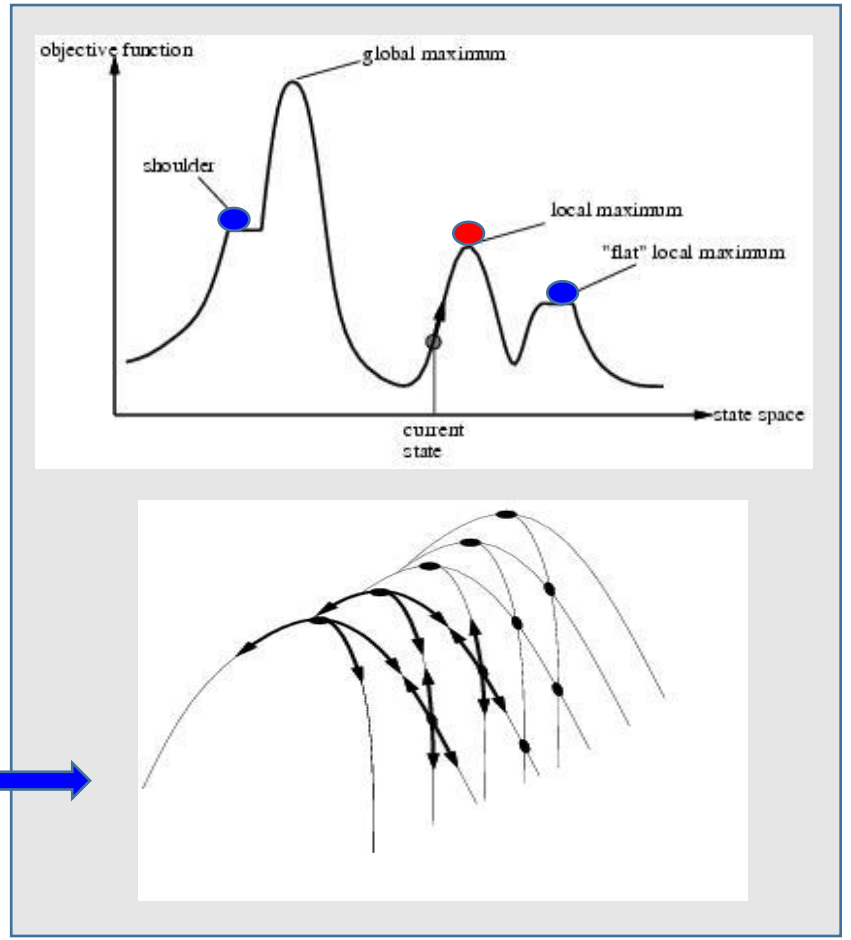
  高于相邻节点但低于全局最大值。

- 高原（Plateaux）：

  平坦的局部最大值，或山肩。即一块状态空间区域的评价函数值是相同的。

- 山脊 (Ridges):

  一系列连续的局部极大值，对于贪婪算法去引导的搜索是困难的。



爬山法变种（Variants of Hill Climbing）：

- **随机爬山法**
  - 随机沿上坡选取下一步。
  - 选取的可能性随山坡的陡峭程度变化移动。与最陡爬坡相比，收敛速度通常较慢。
- **首选爬山法**
  - 随机产生后继节点直到有优于当前节点的后继节点出现。

以上二种：试着避开局部最大。

- **随机重启开始爬山法**

随机生成的初始状态直到找到目标。它十分完备，重新开始

It adopts the well-known adage: "If at first you do not succeed, try, try again."。 **这个算法完备率接近1。**

- **侧向移动：**采用限制次数的方法限制侧移次数，避免死循环。

改进后的8皇后问题，侧移设置为100次成功率由14%上升到94%。