CS 188 春 人工智能介绍
季 2024

笔记4

作者（其他所有笔记）：Nikhil Sharma

作者（贝叶斯网络笔记）：Josh Hug和Jacky Liang，编辑：Regina Wang

作者（逻辑笔记）：Henry Zhu，编辑：Peyrin Kao

出处（机器学习和逻辑笔记）：一些章节改编自教材人工智能：现代方法.

最后更新：2023年8月26日

局部搜索

在之前的笔记中，我们想找到目标状态以及到达目标状态的最佳路径。但在某些问题中，我们只关注找到目标状态——重建路径可能是无关紧要的。例如，在数独游戏中，最佳配置就是目标。一旦你知道它，你就知道通过一个个填充来达到目标。

局部搜索算法允许我们在不考虑路径的情况下找到目标状态。在局部搜索问题中，状态空间是一组“完整的”解决方案。我们使用这些算法来寻找满足某些约束条件或优化某些目标函数的配置。

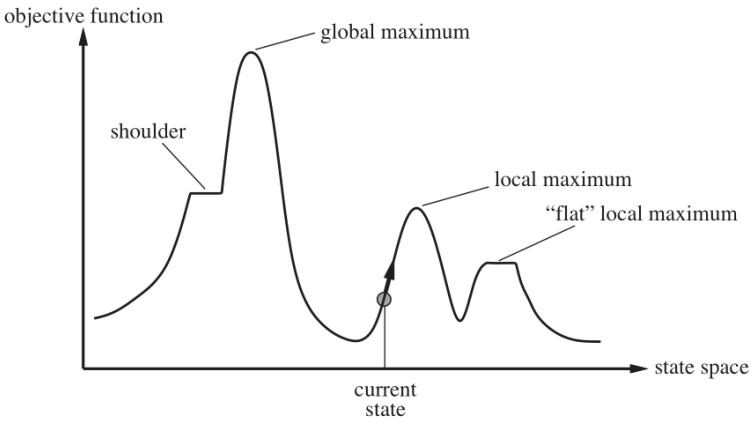


Figure 4.1 A one-dimensional state-space landscape in which elevation corresponds to the objective function. The aim is to find the global maximum. Hill-climbing search modifies the current state to try to improve it, as shown by the arrow. The various topographic features are defined in the text.

上图显示了状态空间上目标函数的一维图。对于那个

函数，我们希望找到对应于最高目标值的状态。局部搜索算法的基本思想是从每个状态局部移动到有更高目标值的状态，直到达到一个最大值（希望是全局最大值）。我们将介绍四种这样的算法，**爬山**、**模拟退火**、**局部束搜索**和**遗传算法**。所有这些算法还用于优化任务，以最大化或最小化目标函数。

爬山搜索

爬山搜索算法（或**陡升搜索**）从当前状态移动到增加目标值最多的邻近状态。该算法不保留搜索树，只保留状态及相应的目标值。爬山的“贪婪”使其容易陷入**局部最大值**（见图4.1），因为在局部这些点似乎是算法的全局最大值，以及**平台**（见图4.1）。平台可分为“平坦”区域，在这些区域没有方向可改善（“平坦局部最大值”），或从中进展缓慢的平坦区域（“肩膀”）。

爬山的变种，如**随机爬山**，随机选择可能的上坡移动中的一个动作，已被提议。随机爬山在实践中已显示出在更多迭代代价下收敛到更高的最大值。另一个变种，**随机侧向移动**，允许不严格增加目标值的移动，从而使算法能够逃离“肩膀”。

```
function HILL-CLIMBING(problem) returns a state
  current ← make-node(problem.initial-state)
  loop do
      neighbor ← a highest-valued successor of current
      if neighbor.value ≤ current.value then
          return current.state
      current ← neighbor
```

如上所示的是爬山算法的伪代码。顾名思义，该算法迭代地移动到具有更高目标值的状态，直到无法继续进展。爬山算法不完整。**随机重启爬山算法**另一方面，通过从随机选择的初始状态进行多次爬山搜索，它显然是完整的，因为在某一些情况下，随机选择的初始状态可以收敛到全局最大值。

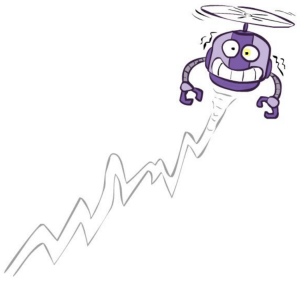
需要注意的是，在本课程后面你会遇到“梯度下降”这个术语。它的基本思想与爬山算法相同，区别在于我们将最大化一个目标函数替换为最小化一个代价函数。

模拟退火搜索

我们将介绍的第二种本地搜索算法是模拟退火。模拟退火旨在结合随机游走（随机移动到邻近状态）和爬山算法以获得一个完整且高效的搜索算法。在模拟退火中，我们允许移动到可能降低目标值的状态。

算法在每个时间步骤选择一个随机移动。如果移动导致目标值升高，则总是接受。如果移动导致目标值降低，则根据某种概率接受。这个概率由温度参数决定，起初温度较高（允许更多的“坏”移动），并按照某种“计划”逐渐降低。从理论上讲，如果温度降低得足够慢，模拟退火算法将以接近1的概率达到全局最大值。

```
function SIMULATED-ANNEALING(problem,schedule) returns a  state
current ← problem.initial-state
for t = 1 to ∞ do
    T ←schedule(t)
    if T = 0 then return current
    next ← a randomly selected successor of current
    ΔE ← next.value – current.value
    if ΔE > 0 then current ← next
              else current ← next only with probability e^(ΔE/T)
```



本地光束搜索

本地光束搜索是爬山搜索算法的另一种变体。两者之间的主要区别在于，本地光束搜索在每次迭代中跟踪k个状态（线程）。算法以随机初始化的k个状态开始，在每次迭代中像爬山算法一样采取k个新状态。这些不只是k个常规爬山算法的副本。关键是，算法从所有线程的所有后继状态的完整列表中选择k个最佳后继状态。如果任何一个线程找到了最优值，算法就停下来。

这些k个线程可以相互共享信息，使得“好”线程（目标值高的线程）也能吸引该区域的其他线程。

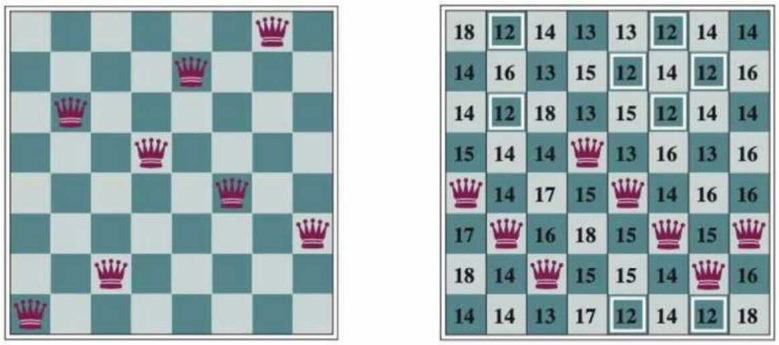
本地光束搜索也容易像爬山算法一样陷入“平坦”区域。**随机光束搜索**类似于随机爬山，可以缓解这个问题。

遗传算法

最后，我们介绍**遗传算法**，它是本地光束搜索的一种变体，也广泛应用于许多优化任务。顾名思义，遗传算法从进化中获取灵感。遗传算法以k个随机初始化状态（称为**种群**）开始。状态（称为**个体**）用有限字母表上的字符串表示。

让我们回到课堂上介绍的**8皇后问题**。回顾一下，8皇后问题是一个约束满足问题，我们希望在8x8的棋盘上放置8个皇后。满足约束的解决方案不会有任何**攻击的皇后对**，即不会有皇后在同一行、列或对角线。之前介绍的所有算法都可以应用于解决8皇后问题。

Figure 4.3



(a) The 8-queens problem: place 8 queens on a chess board so that no queen attacks another. (A queen attacks any piece in the same row, column, or diagonal.) This position is almost a solution, except for the two queens in the fourth and seventh columns that attack each other along the diagonal. (b) An 8-queens state with heuristic cost estimate $h = 17$. The board shows the value of h for each possible successor obtained by moving a queen within its column. There are 8 moves that are tied for best, with $h = 12$. The hill-climbing algorithm will pick one of these.

对于遗传算法，我们用从1到8的数字来表示每一个皇后在列中的位置（图4.6中的(a)列）。每个个体都通过一个评估函数（**适应度函数**）来进行评估，并根据该函数的值进行排名。对于8皇后问题，这个值是无冲突的皇后对的数量。

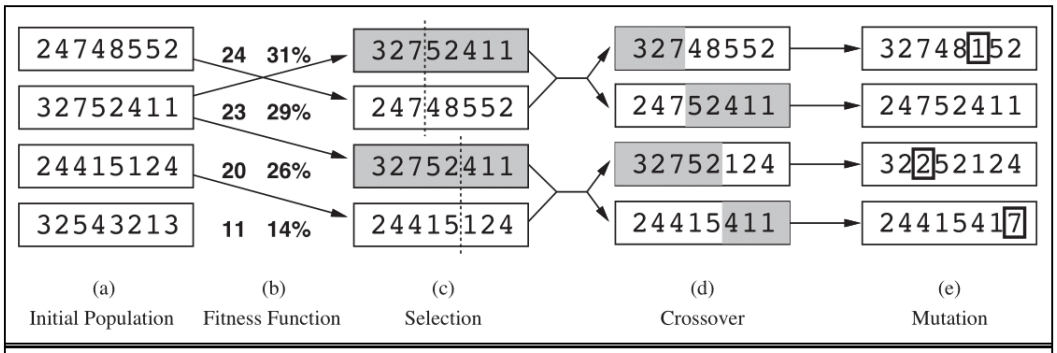


Figure 4.6 The genetic algorithm, illustrated for digit strings representing 8-queens states. The initial population in (a) is ranked by the fitness function in (b), resulting in pairs for mating in (c). They produce offspring in (d), which are subject to mutation in (e).

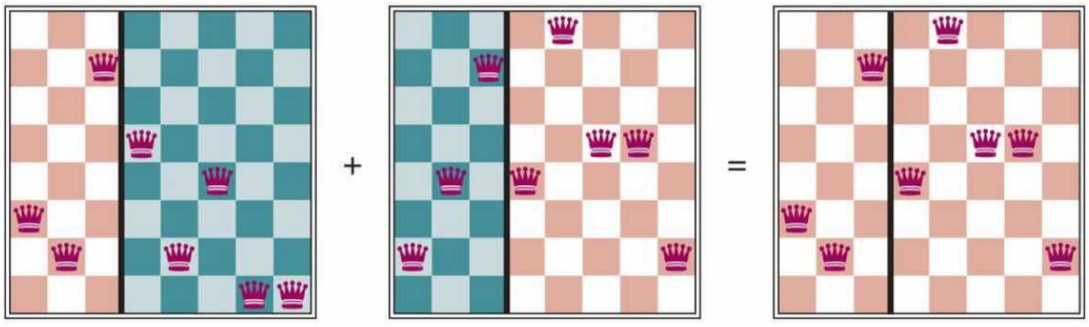
选择一个状态进行“繁殖”的概率与该状态的值成正比。我们通过从这些概率中抽样来选择要繁殖的状态对（图4.6中的(c)列）。通过在交叉点对父母字符串进行交叉来生成后代。交叉点是对每对随机选择的。最后，每个后代都有独立概率发生随机突变。遗传算法的伪代码如下所示。

```
function GENETIC-ALGORITHM(population, FITNESS-FN) returns an individual
  inputs: population, a set of individuals
          FITNESS-FN, a function that measures the fitness of an individual

  repeat
    new_population ← empty set
    for i = 1 to SIZE(population) do
      x ← RANDOM-SELECTION(population, FITNESS-FN)
      y ← RANDOM-SELECTION(population, FITNESS-FN)
      child ← REPRODUCE(x, y)
      if (small random probability) then child ← MUTATE(child)
      add child to new_population
    population ← new_population
  until some individual is fit enough, or enough time has elapsed
  return the best individual in population, according to FITNESS-FN

function REPRODUCE(x, y) returns an individual
  inputs: x, y, parent individuals

  n ← LENGTH(x); c ← random number from 1 to n
  return APPEND(SUBSTRING(x, 1, c), SUBSTRING(y, c + 1, n))
```

Figure 4.8 A genetic algorithm. The algorithm is the same as the one diagrammed in Figure 4.6, with one variation: in this more popular version, each mating of two parents produces only one offspring, not two.

与随机束搜索相似，遗传算法在探索状态空间的同时试图向上移动并在线程之间交换信息。它们的主要优势是交叉 — 进化并导致高评价的大块字母可以与其他这样的块组合并产生得分高的解决方案。

Figure 4.7



The 8-queens states corresponding to the first two parents in Figure 4.6(c) and the first offspring in Figure 4.6(d). The green columns are lost in the crossover step and the red columns are retained. (To interpret the numbers in Figure 4.6: row 1 is the bottom row, and 8 is the top row.)

总结

在这篇笔记中，我们讨论了局部搜索算法及其动机。当我们不在乎到达某个目标状态的路径，并且想要满足某些约束或优化某个目标时，可以使用这些方法。局部搜索方法允许我们在处理大的状态空间时节省空间并找到合适的解决方案！

我们介绍了一些基础的局部搜索方法，它们相互构建：

- 爬山法
- 模拟退火
- 局部束搜索
- 遗传算法

在本课程后面，尤其是在我们讨论神经网络时，优化函数的思想将会再次出现。