

第四章 搜索技术

——2. 超越经典搜索

主讲人 赵曼

中国地质大学 计算机学院计算机科学系

1

超越经典概述

2

局部搜索

3

优化算法

4

CSP问题

5

联机搜索

6

小结



1

概述

2

局部搜索

3

优化算法

4

CSP问题

5

联机搜索

6

小结



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，

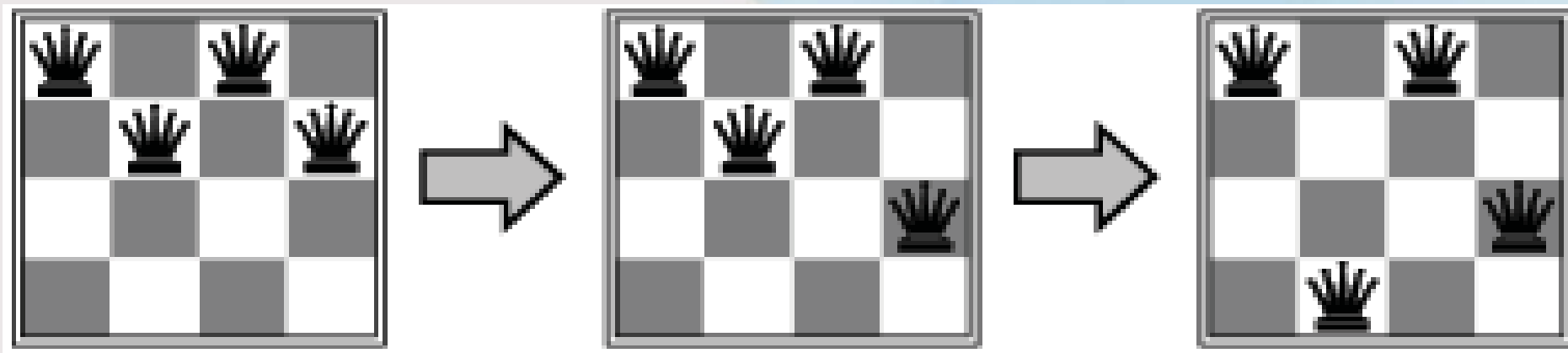
- 在许多问题中，到达目标的路径是无关紧要的。

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。

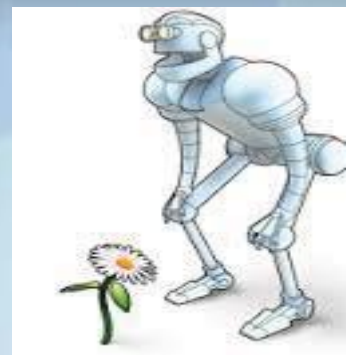
1.3 局部搜索

- 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



1

概述

2

局部搜索

2.1 爬山法 (Hill-Climbing Search)

2.2 局部束搜索 (Local Beam Search)

2.3 禁忌搜索 (Tabu Search)

3

优化算法

4

CSP问题

5

联机搜索

6

小结



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

18	12	14	13	13	12	14	14
14	16	13	15	12	14	12	16
14	12	18	13	15	12	14	14
15	14	14	♙	13	16	13	16
♙	14	17	15	♙	14	16	16
17	♙	16	18	15	♙	15	♙
18	14	♙	15	15	14	♙	16
14	14	13	17	12	14	12	18

➤ **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。

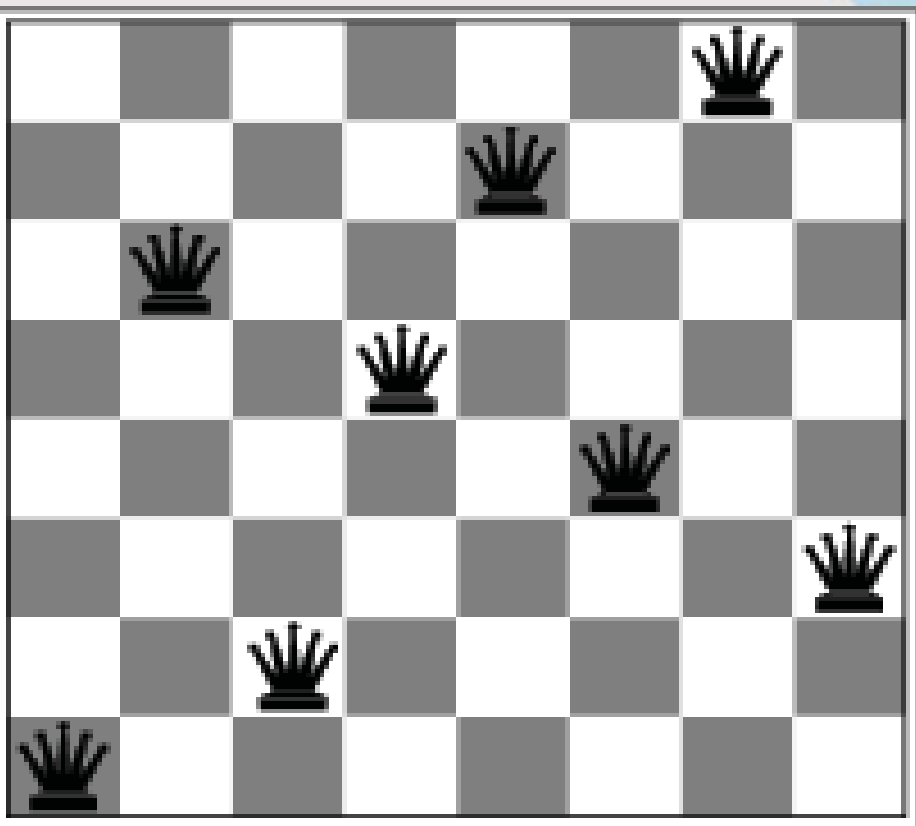
$7 \times 8 = 56$ 个后继状态

➤ **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



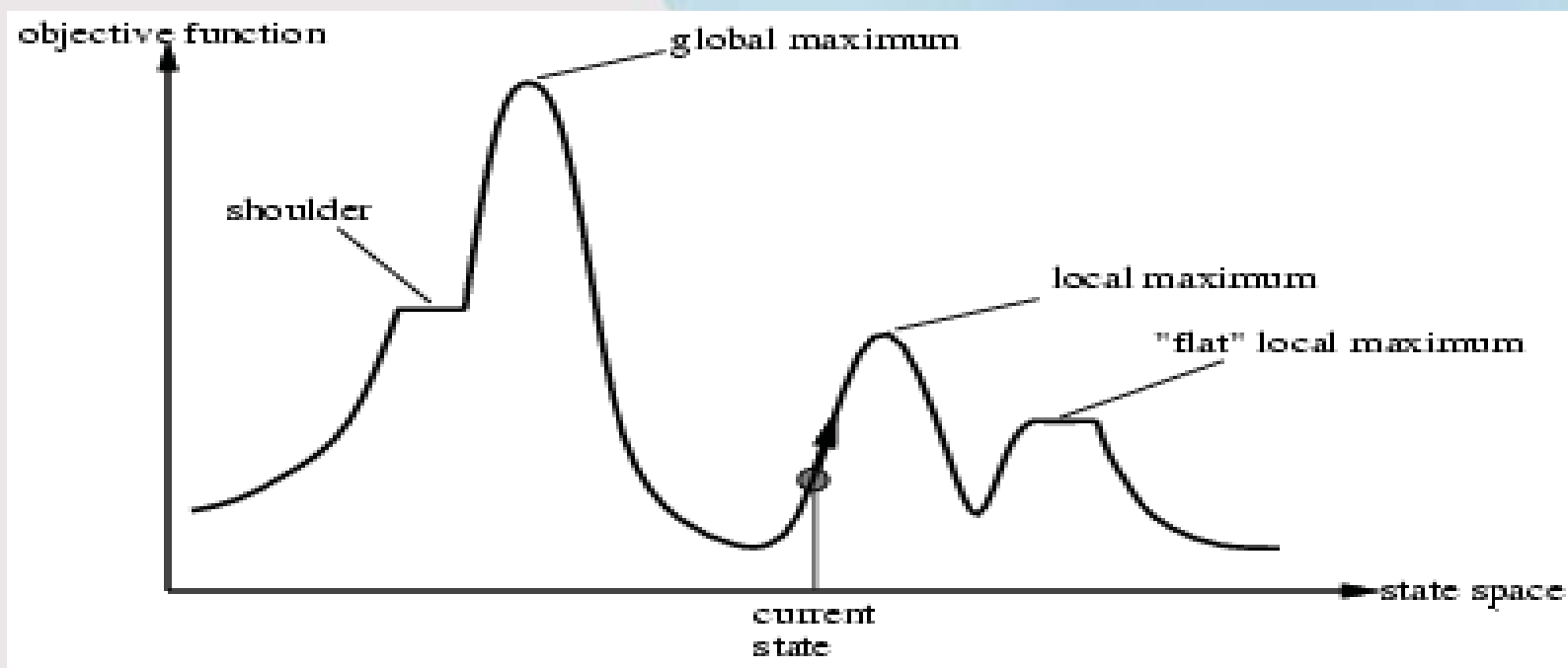
- 一个局部最小值点 ($h = 1$)
- 依照爬山法该问题求解以失败结束。

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：

以下三种情况经常被困：

局部极大值 (Local maxima)

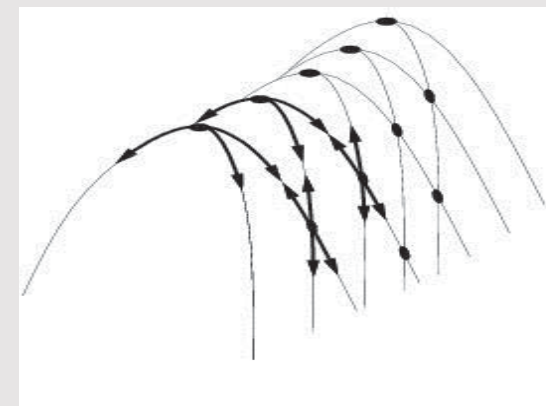
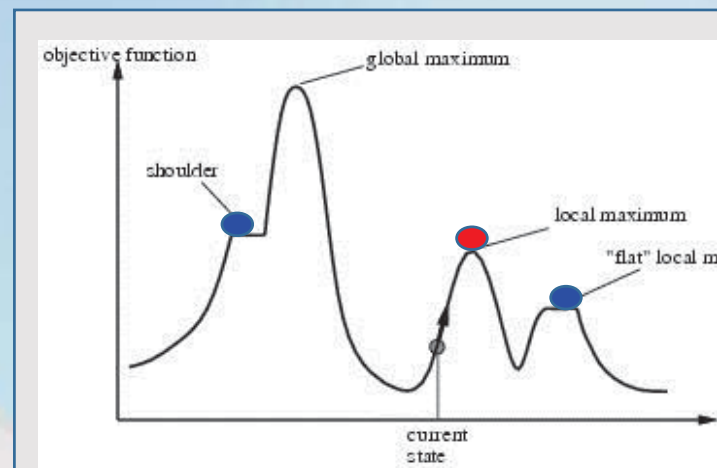
高于相邻节点但低于全局最大值。

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

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%。

2局部束搜索

内存中仅保存一个节点似乎是对内存限制问题的极端反应。

局部束搜索：保存 k 状态，而不是一个状态

- 从 k 个随机产生的状态开始

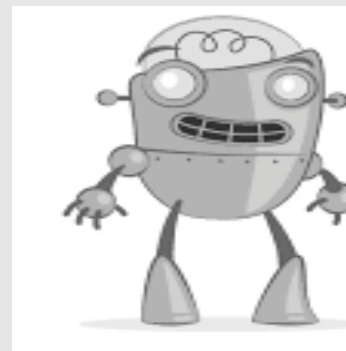
- 每次迭代,产生 k 个状态的所有后续

- 如果任何一个新产生的状态是目标状态，则停止。否则从所有后续中选择 k 个最好的后续，重复迭代。

//同随机重新开始的差别：在 k 条线程中共享信息。

缺点：局部束搜索会很快地集中在状态空间的某个小区域内，使得搜索代价比爬山还要昂贵。**缺乏多样性。**

改进的随机变种：它不是选择 k 个最佳后继节点，而是以选择后继节点的概率是其值增函数，来随机地选择 k 个后继节点。（随机束搜索有些类似于自然选择的过程）



2.3 禁忌搜索

- 禁忌 (prohibited or restricted by social custom) , 指的是不能触及的事物。
- 禁忌搜索是由弗雷德·格洛夫于1986年提出, 1989年加以形式化。
- 它是一种**元启发式(meta-heuristic)** 算法, 用于解决组合优化问题。
- 它使用一种局部搜索或邻域搜索过程, 从一个潜在的解 x 到改进的相邻解 x' 之间反复移动, 直到满足某些停止条件。
- 用于确定解的数据结构被称为**禁忌表** (tabu list) 。

禁忌搜索的三种策略

- Forbidding strategy 禁止策略
control what enters the tabulist. 控制何物进入该禁忌表。
- Freeing strategy 释放策略
control what exits the tabulist and when. 控制何物以及何时退出该禁忌表。
- Short-term strategy 短期策略
manage interplay between the forbidding strategy and freeing strategy
select trial solutions. 管理禁止策略和释放策略之间的相互作用来选择试验解。
- ✓ aspiration criterion 特赦准则

禁忌搜索算法中，迭代的某一步会出现候选集的某一个元素被禁止搜索，但是若禁该元素，则会使评价函数有所改善，因此我们需要设置一个特赦规则，当满足该条件时该元素从禁忌表中跳出。

Tabu Search Algorithm 禁忌搜索算法

function TABU-SEARCH(s') **return** a best candidate

$sBest \leftarrow s \leftarrow s'$

$tabuList \leftarrow$ null list

while(**not** STOPPING-CONDITION())

$candidateList \leftarrow$ null list

$bestCandidate \leftarrow$ null

for($sCandidate$ in $sNeighborhood$)

if((**not** $tabuList.CONTAINS(sCandidate)$) **and** ($FITNESS(sCandidate) > FITNESS(bestCandidate)$))

Then $bestCandidate \leftarrow sCandidate$

$s \leftarrow bestCandidate$

if($FITNESS(bestCandidate) > FITNESS(sBest)$) **then** $sBest \leftarrow bestCandidate$

$tabuList.PUSH(bestCandidate)$

if($tabuList.SIZE > maxTabuSize$) **then** $tabuList.REMOVE-FIRST()$

return $sBest$

简洁版

(1) 给定一个禁忌表 ($tabuList$) = null,
并选定一个初始解 s' 。

(2) 如果满足停止规则, 则停止计算, 输出结果;
在 s 的邻域 $sNeighborhood$ 中选出满足不受禁忌的候选解 $sCandidate$ 。在 $sCandidate$ 中选择一个评价价值最优的解 $bestCandidate$, $bestCandidate := sCandidate$;
更新历史记录 $tabuList$ 。
重复步骤 (2)

可用禁忌搜索 解决的问题

- Travelling Salesperson Problem
- Traveling Tournament Problem
- Job-shop Scheduling Problem
- Network Loading Problem
- The Graph Coloring Problem
- Hardware/Software Partitioning
- Minimum Spanning Tree Problem

旅行推销员问题

旅行比赛问题

车间作业调度问题

网络加载问题

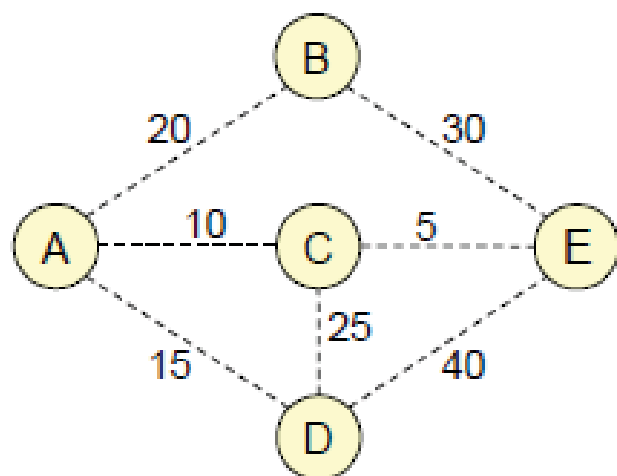
图着色问题

硬件/软件划分

最小生成树 问题

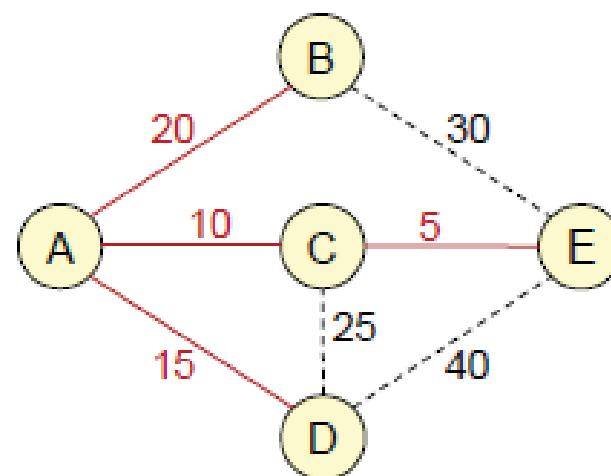
Minimum Spanning Tree Problem 最小生成树问题

- Objective 目标：用最小代价连接所有节点



Connects all nodes with minimum cost

用最小代价连接所有节点



An optimal solution without constraints

一个无约束的最优解

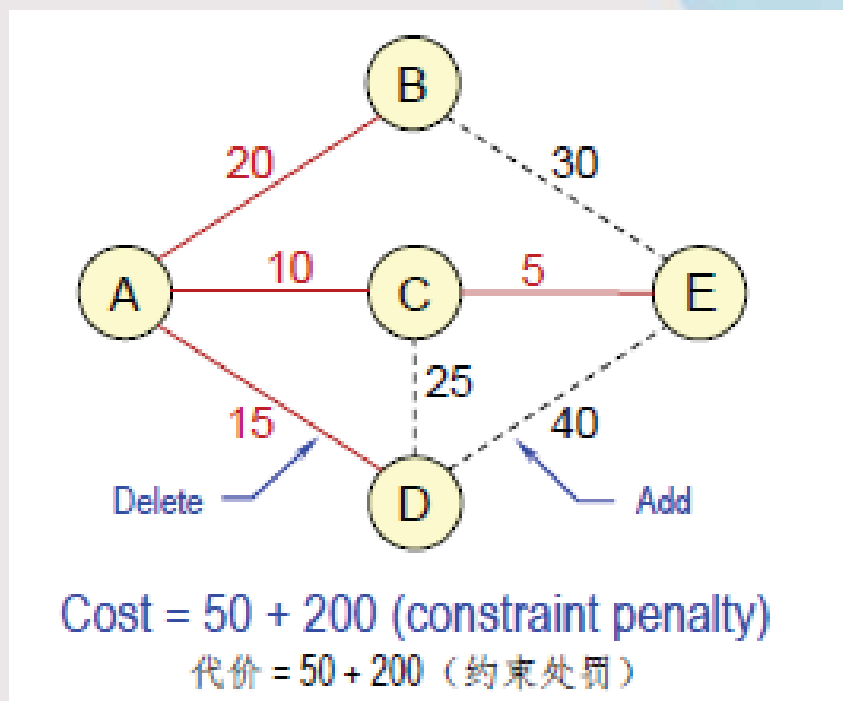
约束1：仅当包含连接DE时，才可以包含连接AD。（处罚：100）

约束2：至多可以包含三个连接（AD，CD和AB）中的一个。

（处罚：若选择了三个中的两个则处罚100，选择了全部三个则罚200）

Minimum Spanning Tree Problem 最小生成树问题

• Iteration 1 迭代1



Local optimum
局部最优

Add	Delete	Cost
BE	CE	$75 + 200 = 275$
BE	AC	$70 + 200 = 270$
BE	AB	$60 + 100 = 160$
CD	AD	$60 + 100 = 160$
CD	AC	$65 + 300 = 365$
DE	CE	$85 + 100 = 185$
DE	AC	$80 + 100 = 180$
DE	AD	$75 + 0 = 75$

New Cost = 75
新代价=75

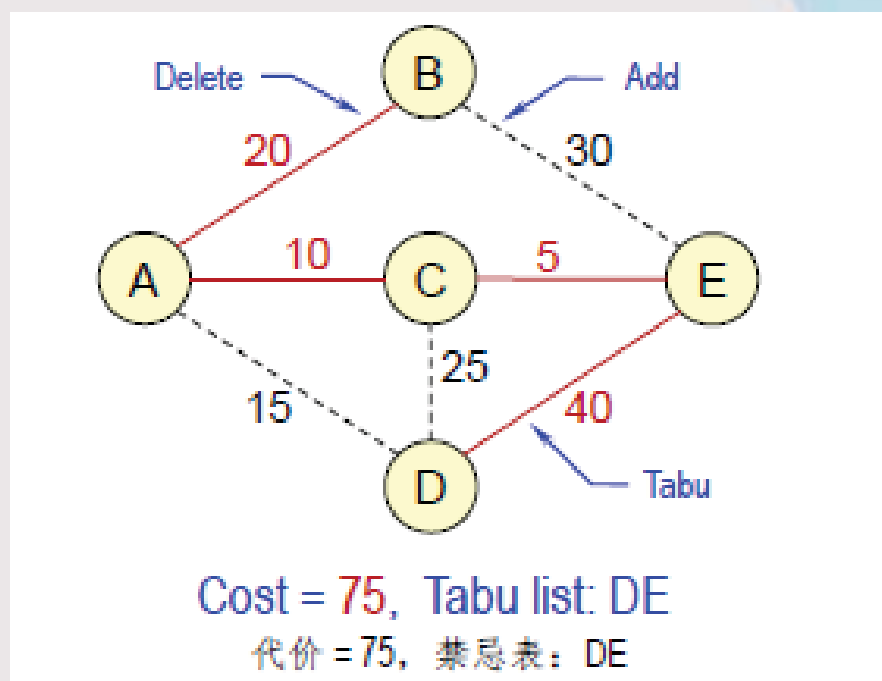
约束1: 仅当包含连接DE时, 才可以包含连接AD。(处罚: 100)

约束2: 至多可以包含三个连接(AD, CD和AB)中的一个。

(处罚: 若选择了三个中的两个则处罚100, 选择了全部三个则罚200)

Minimum Spanning Tree Problem 最小生成树问题

Iteration 2 迭代2



Escape local Optimum
溢出局部最优

Add	Delete	Cost
AD	DE*	Tabu Move
AD	CE	$85 + 100 = 185$
AD	AC	$80 + 100 = 180$
BE	CE	$100 + 0 = 100$
BE	AC	$95 + 0 = 95$
BE	AB	$85 + 0 = 85$
CD	DE*	Tabu Move
CD	CE	$95 + 100 = 190$

New Cost = 85
新代价=85

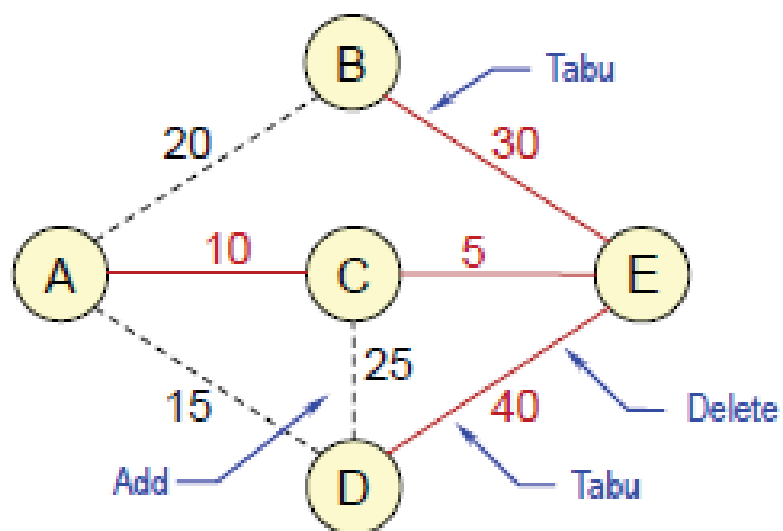
约束1: 仅当包含连接DE时, 才可以包含连接AD。(处罚: 100)

约束2: 至多可以包含三个连接(AD, CD和AB)中的一个。

(处罚: 若选择了三个中的两个则处罚100, 选择了全部三个则罚200)

Minimum Spanning Tree Problem 最小生成树问题

Iteration 3 迭代3



Override tabu status
覆盖禁忌状态

Add	Delete	Cost
AB	BE*	Tabu Move
AB	CE	$100 + 0 = 100$
AB	AC	$95 + 0 = 95$
AD	DE *	$60 + 100 = 160$
AD	CE	$95 + 0 = 95$
AD	AC	$90 + 0 = 90$
CD	DE*	$70 + 0 = 70$
CD	CE	$105 + 0 = 105$

约束1: 仅当包含连接DE时, 才可以包含连接AD。(处罚: 100)

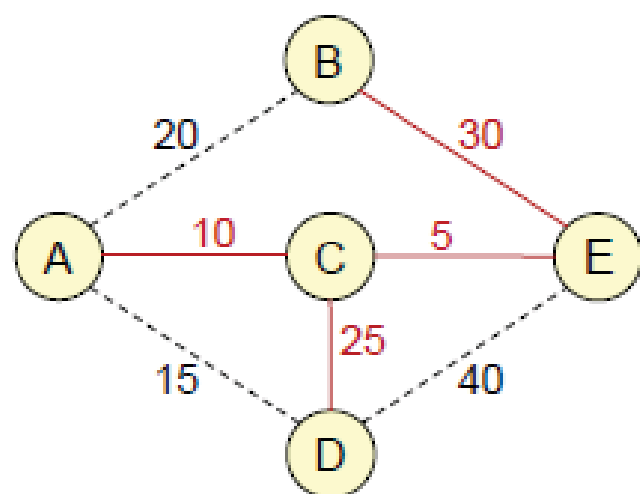
约束2: 至多可以包含三个连接 (AD, CD和AB) 中的一个。

(处罚: 若选择了三个中的两个则处罚100, 选择了全部三个则罚200)

New Cost = 70
新代价 = 70

Minimum Spanning Tree Problem 最小生成树问题

• Iteration 4 迭代4



Cost = 70
代价 = 70

Optimal Solution
最优解

*Additional iterations only find
inferior solutions*

额外的迭代只会找到较差解

约束1: 仅当包含连接DE时, 才可以包含连接AD。(处罚: 100)

约束2: 至多可以包含三个连接 (AD, CD和AB) 中的一个。

(处罚: 若选择了三个中的两个则处罚100, 选择了全部三个则罚200)

禁忌搜索的应用领域

- | | |
|------------------------------|---------|
| • Resource planning | 资源规划 |
| • Telecommunications | 通讯 |
| • VLSI design | VLSI 设计 |
| • Financial analysis | 金融分析 |
| • Scheduling | 调度 |
| • Space planning | 空间规划 |
| • Energy distribution | 能源分配 |
| • Molecular engineering | 分子工程 |
| • Logistics | 后勤保障 |
| • Flexible manufacturing | 柔性生产 |
| • Waste management | 废物管理 |
| • Mineral exploration | 矿产勘探 |
| • Biomedical analysis | 生物医药分析 |
| • Environmental conservation | 环境保护 |

1

超越经典概述

2

局部搜索

3

优化算法

3.1 模拟退火 Simulated Annealing

4

CSP问题

5

联机搜索

6

小结



1 Simulated Annealing 模拟退火搜索

引论：爬山法从不下山的策略导致其不完备性，而纯粹的随机行走，又导致效率的低下，把单纯的爬山法和纯粹的随机相结合就是以下的**模拟退火**算法的思想。

Idea：通过允许向不好的状态移动来避开局部最值点，但频率逐渐降低。

起源：冶金中的退火原理。

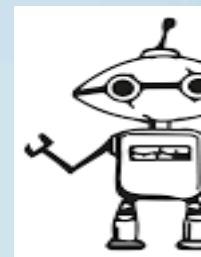
退火用于对金属和玻璃进行回火或硬化。

将一个固体放在高温炉内进行加热，提升温度至最大值。在该温度下，所有的材料都处于液体状态，并且粒子本身随机地排列。随着高温炉内的温度逐渐冷却，该结构的所有粒子将呈现低能状态。

例子：弹球的分析

剧烈的摇动（=高温）

减少晃动（=降温）

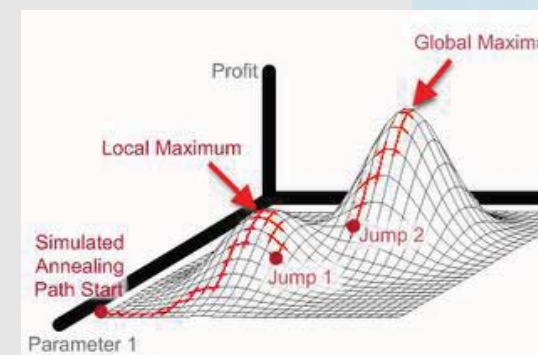


Simulated Annealing 模拟退火搜索

模拟退火是一种给定函数逼近全局最优解的概率方法。发表于 1953 年。
具体来说，是一种在大搜索空间逼近全局最优解的**元启发式**方法。

Optimization and Thermodynamics 优化与热力学

Objective function	目标函数 \Leftrightarrow Energy level	能量极位
Admissible solution	可接受解 \Leftrightarrow System state	系统状态
Neighbor solution	相邻解 \Leftrightarrow Change of state	状态变化
Control parameter	控制参数 \Leftrightarrow Temperature	温度
Better solution	更优解 \Leftrightarrow Solidification state	凝固状态



Simulated Annealing 模拟退火算法

- Initial Solution 初始解

Generated using an heuristic Chosen at random.

使用随机选择启发式方法生成。

- Neighborhood 相邻节点

Generated randomly. Mutating the current solution.

随机生成。当前解的变异。

- Acceptance 接受条件

Neighbor has lower cost value, higher cost value is accepted with the probability p .

相邻节点具有较低代价值，具有较高代价值的相邻节点则以概率 P 接受。

- Stopping Criteria 停止判据

Solution with a lower value than threshold. Maximum total number of iterations.

解具有比阈值低的值。已达到迭代最大总次数。

模拟退火算法

function SIMULATED-ANNEALING(*problem*, *schedule*) **returns** a solution state

inputs: *problem*, a problem

schedule, a mapping from time to “temperature”

current ← MAKE-NODE(*problem*.INITIAL-STATE)

for $t = 1$ **to** ∞ **do**

$T \leftarrow \text{schedule}(t)$

if $T = 0$ **then return** *current* //

next ← a randomly selected successor of *current*

$\Delta E \leftarrow \text{next.V}_{\text{VALUE}} - \text{current.V}_{\text{VALUE}}$

if $\Delta E > 0$ **then** *current* ← *next*

else *current* ← *next* only with probability $e^{\Delta E/T}$

一种允许部分下山移动的随机爬山法的版本。下山移动在退火调度的早期更容易接受，随着时间的推移逐步降低。

模拟退火搜索的性质

- 可以证明: 如果 T 降低的足够慢, 则模拟退火能以趋近于1的概率找到最优解。
- 广泛应用于VLSI 布局问题, 航空调度等领域

