中国地质大学
China University of Geosciences

第四章

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

Zhaoman, AI,
Department of Computer Science, CS, CUG

中国地质大学 China University of Geosciences

目 录

Zhaoman, AI,
Department of Computer Science, CS, CUG

中国地质大学 China University of Geosciences

目 录

Zhaoman, AI,
Department of Computer Science, 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：
  - 局部搜索和群体智能。

1.2 经典搜索

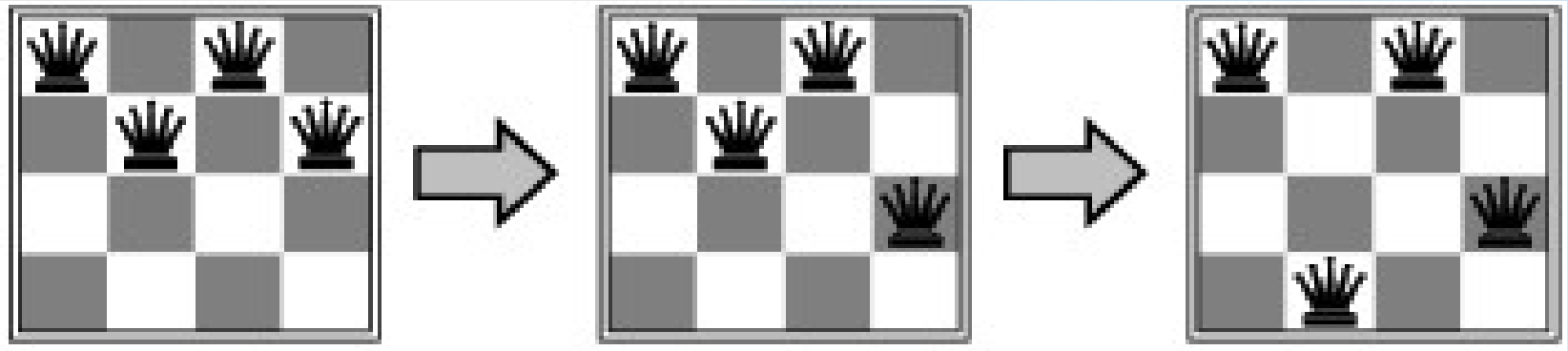
- 具有如下特点：
  - 搜索算法被设成系统地探索问题的空间。
  - 该系统性是由以下方法得到的：在内存中保持一条或多条路径，并且在沿着该路径的每个点上记录哪些已被探索过。
  - 目标被找到时，该路径也就构成问题的一个解。
- 然而，
  - 在许多问题中，到达目标的路径是无关紧要的 。

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* × *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　通讯
  - network optimization　网络优化
  - vehicle routing　车辆路由
  - portfolio management　投资组合管理

1.4群体智能

- 研究来自于“集群智能”（collective intelligence）灵感的计算系统。
- 集群智能是环境中大量的同类智能体通过合作来实现的。

  例如：鱼群、鸟群、以及蚁群。
- 这样的智能是分散式、自组织、并且在一个环境内分布。
- 事实上，这类系统通常用于：

  有效觅食、猎物躲避、群体搬迁、等等。

Algorithms of Swarm Intelligence 群体智能的算法

- Altruism algorithm 利他算法
- Ant colony optimization 蚁群优化
- Bee colony algorithm 蜂群算法
- Artificial immune systems 人工免疫 系统
- Bat algorithm 蝙蝠算法
- Differential evolution 差分进化
- Multi-swarm optimization 多群体优化
- Gravitational search algorithm 引力搜索算法
- Glowworm swarm optimization 萤火虫群优化
- Particle swarm optimization 粒子群优化
- Bacterial colony optimization 细菌群优化
- River formation dynamics 河流形成动力学
- Self-propelled particles 自行式粒子系统
- Stochastic diffusion search 随机扩散搜索

群体智能的典型算法：

蚁群优化：受蚁群的行为所启发，诸如：

间接协调（stigmergy）、

觅食（foraging）。

粒子群优化：受鸟群和鱼群的社会行为所启发，诸如：

群集（flocking）、

从众（herding）。

中国地质大学 China University of Geosciences

目 录

Zhaoman, AI,
Department of Computer Science, CS, CUG

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
```

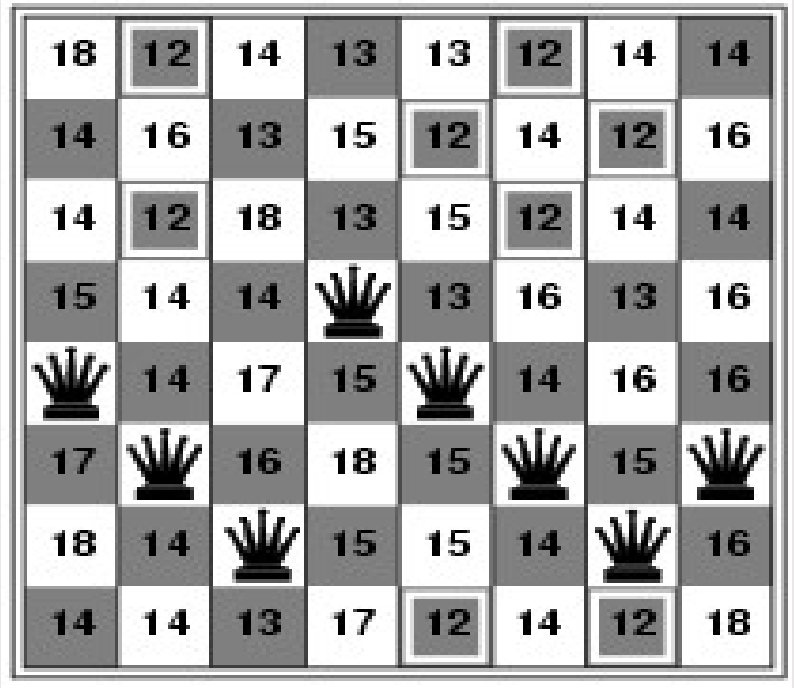
最陡爬坡版（*Steepest-ascent version*）：当前节点每一步都用最佳邻接点（具有最高值的邻接点）替换，否则到达一个“峰值”。

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



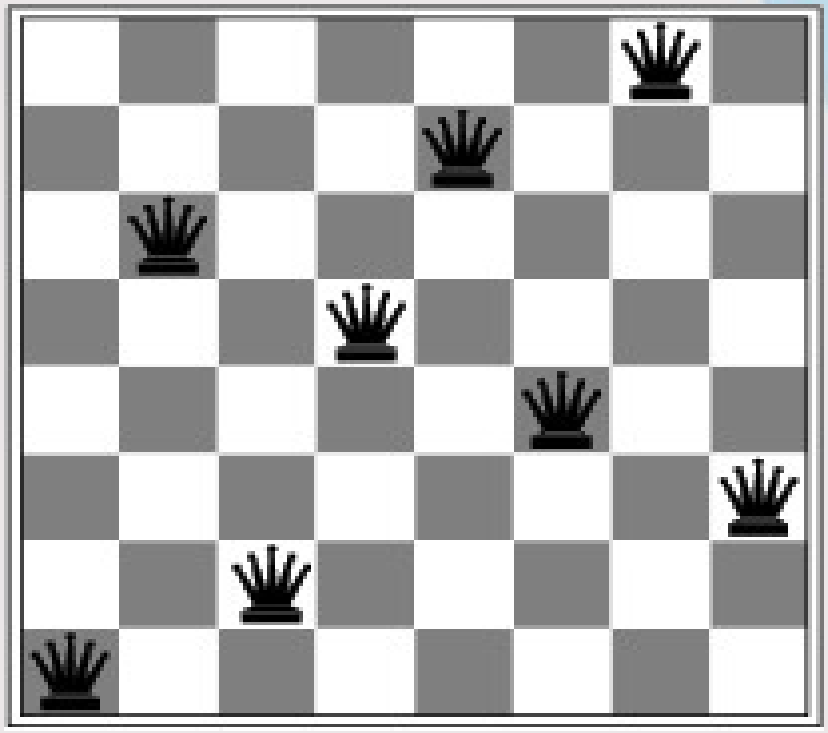
- **问题形式化：** 完全状态形式化。
- **后继函数：** 移动一个皇后到同列另一个方格中的所有可能状态。

  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



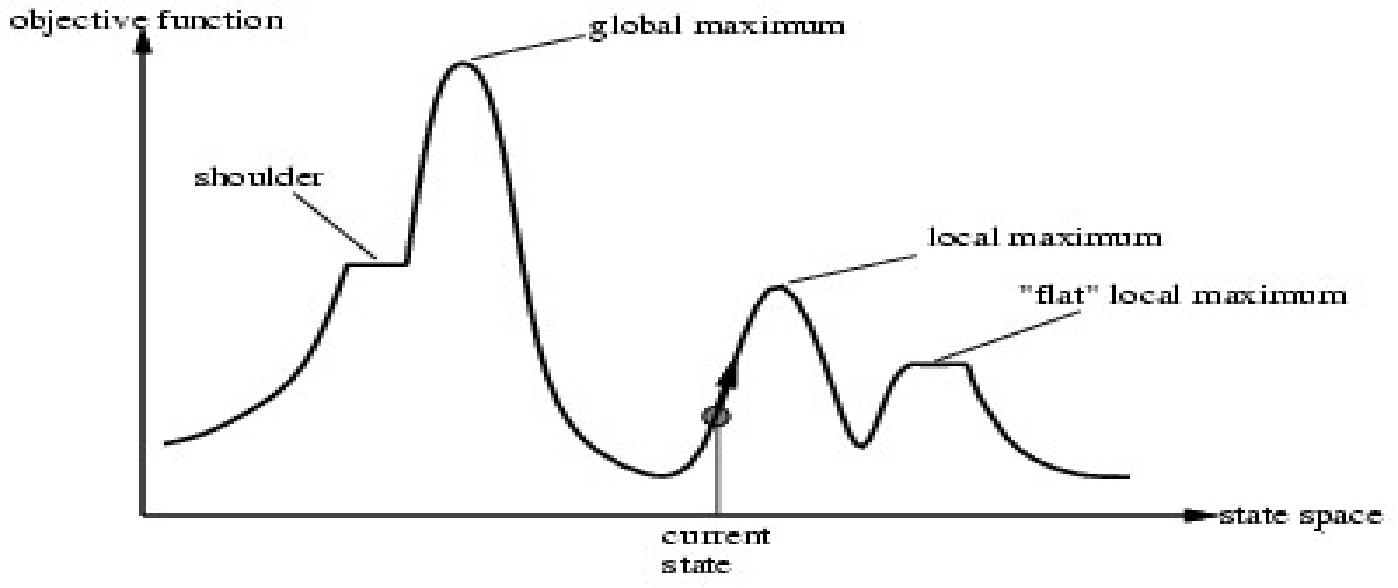
- 一个局部最小值点（ h = 1）
- 依照爬山法该问题求解以失败结束。

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）

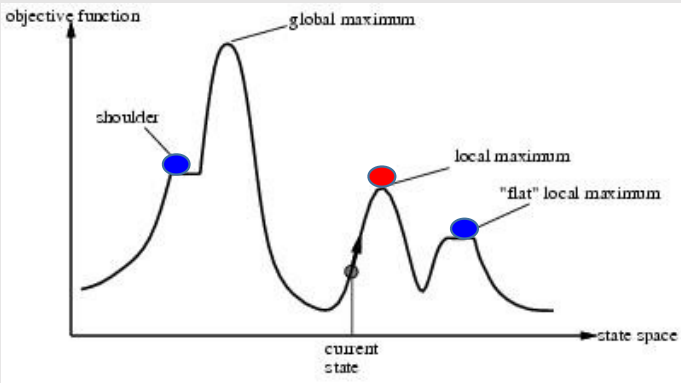
  高于相邻节点但低于全局最大值。

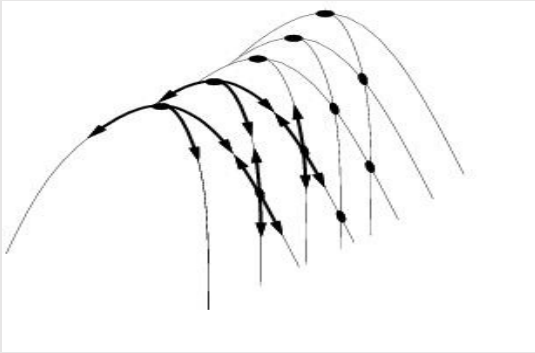
- 高原（Plateaux）：

  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。

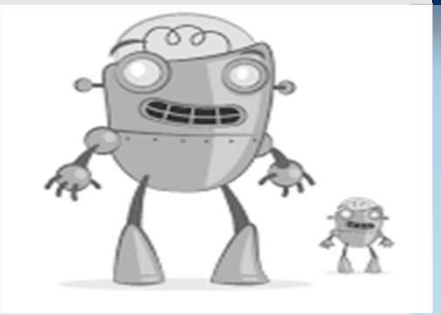
- 随机重启开始爬山法

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 don't 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%。

2.2局部束搜索



- 在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：保持保存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。
- 同随机重新开始的区别：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法还要昂贵。

  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的递增函数，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·格洛弗于1986年提出，1989年加以形式化。
- 它是一种元启发式算法，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 to 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。

TabuSearch Algorithm 禁忌搜索算法

```
function TABU-SEARCH(s') return a best candidate
        sBest← s← s'
        tabuList← null list
        while(not STOPPING-CONDITION())
                candidateList← null list
                bestCandidate← null
                for(sCandidate in sNeighborhood)
                        if((not tabuList.CONTAINS(sCandidate))
                            and (FITNESS(sCandidate) > FITNESS(bestCandidate)))
                        Then bestCandidate← sCandidate
                s← bestCandidate
                if(FITNESS(bestCandidate) > FITNESS(sBest)) then sBest← 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'*。

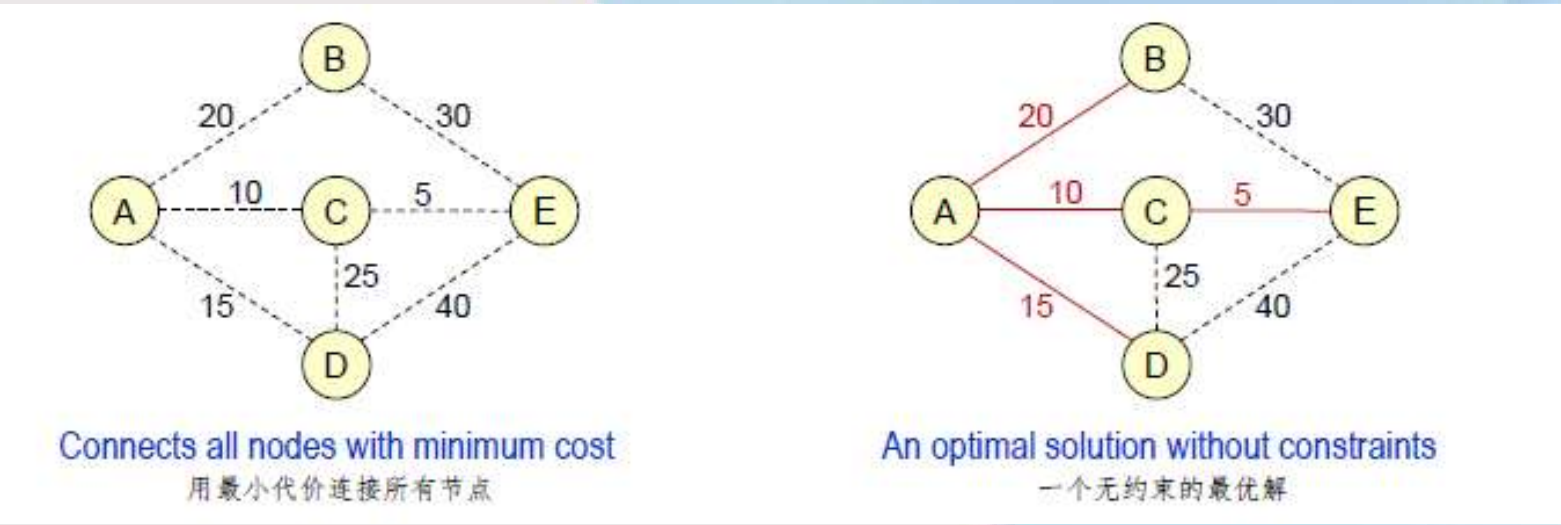
（2）如果满足停止规则，则停止计算，输出结果；否则，在*s*的邻域*sNeighborhood*中选出满足不受禁忌的候选集*sCandidate*。在*sCandidate*中选择一个评价值最低的解*bestCandidate*，*bestCandidate*：＝*sCandidate*；更新历史记录*tabuList*。重复步骤（2）.

可用禁忌搜索 解决的问题

- Travelling Salesperson Problem 旅行推销员问题
- Traveling Tournament Problem 旅行比赛问题
- Job-shop Scheduling Problem 车间作业调度问题
- Network Loading Problem 网络加载问题
- The Graph Coloring Problem 图着色问题
- Hardware/Software Partitioning 硬件/软件划分
- Minimum Spanning Tree Problem 最小生成树 问题

Minimum Spanning Tree Problem 最小生成树问题

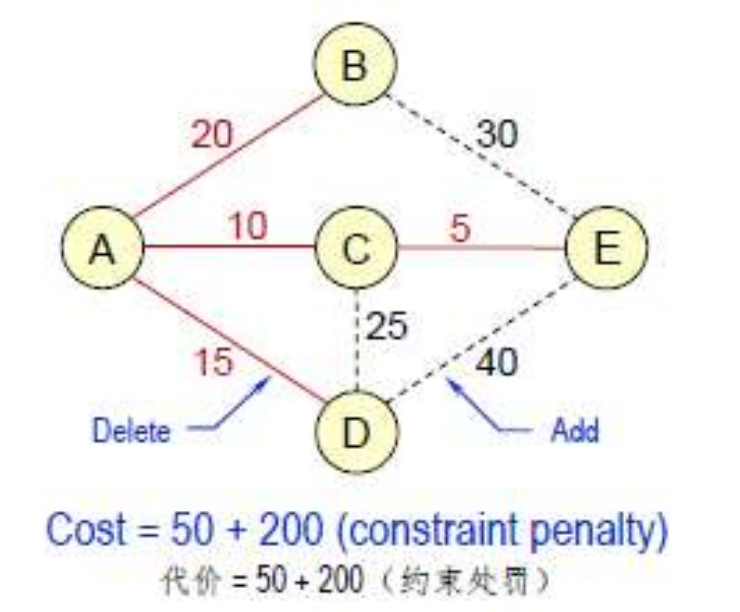
- Objective 目标：用最小代价连接所有节点



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

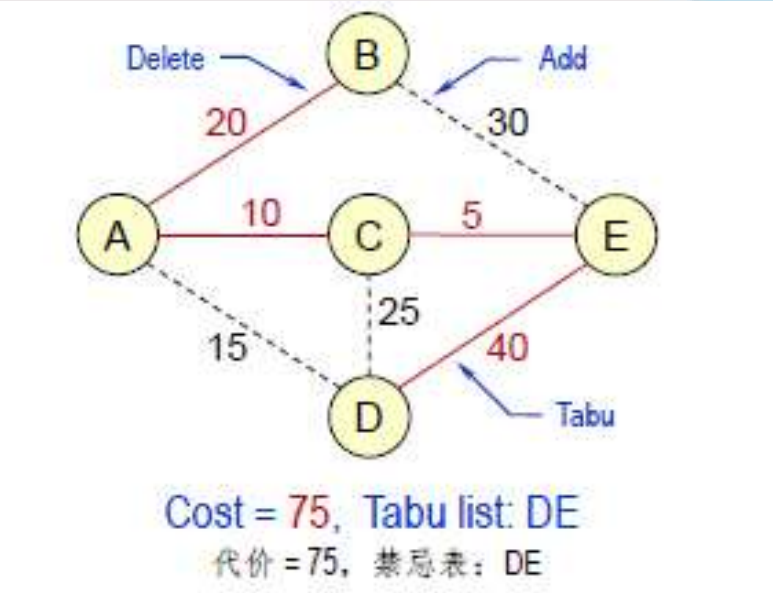
Add	Delete	Cost
BE	CE	75 + 200 = 275
BE	AC	70 + 200 = 270
BE	AB	60 + 100 = 160
CD	AD	60 + 100 = 160
CD	AC	65 + 300 = 365
DE	CE	85 + 100 = 185
DE	AC	80 + 100 = 180
DE	**AD**	**75 + 0 = 75**

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

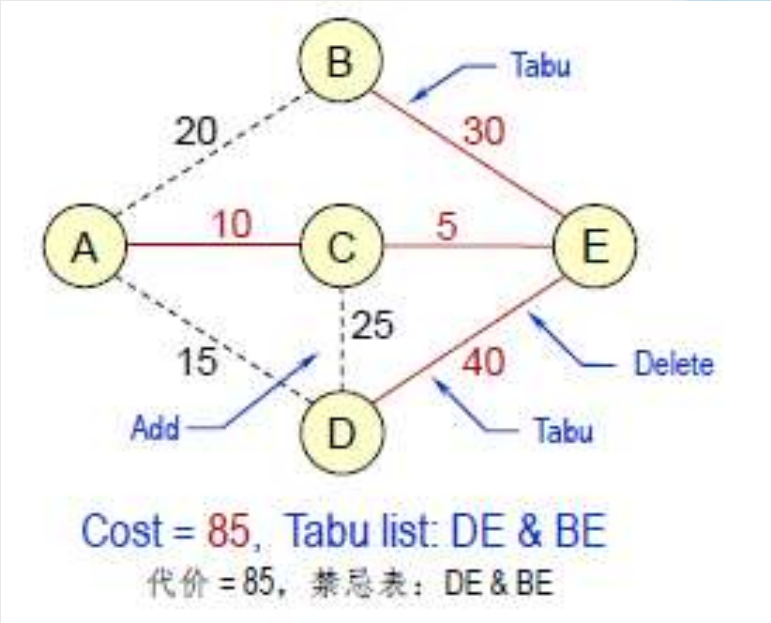
Add	Delete	Cost
AD	DE*	Tabu Move
AD	CE	85 + 100 = 185
AD	AC	80 + 100 = 180
BE	CE	100 + 0 = 100
BE	AC	95 + 0 = 95
BE	AB	85 + 0 = 85
CD	DE*	Tabu Move
CD	CE	95 + 100 = 190

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

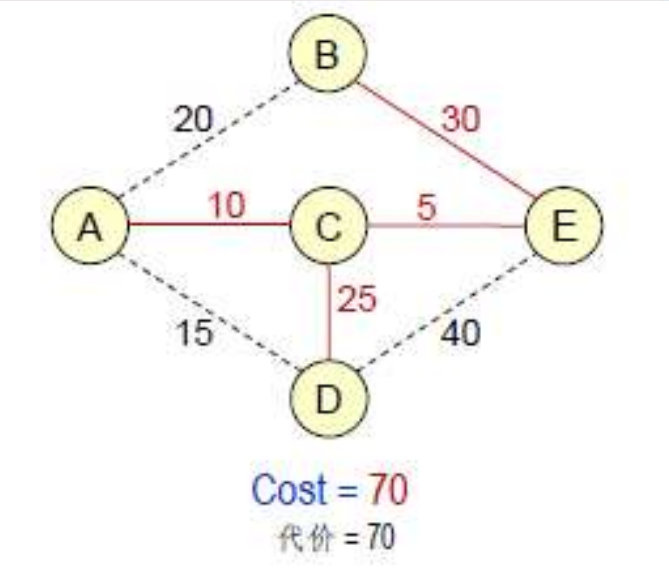
约束1：仅当包含连接DE时，才可以包含连接AD。（处罚：100）
约束2：至多可以包含三个连接（AD，CD和AB）中的一个。
（处罚：若选择了三个中的两个则处罚100，选择了全部三个则罚200）

New Cost = 70
新代价 = 70

Zhaoman, AI,
Department of Computer Science, CS, CUG

Minimum Spanning Tree Problem 最小生成树问题

- Iteration 4 迭代4



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

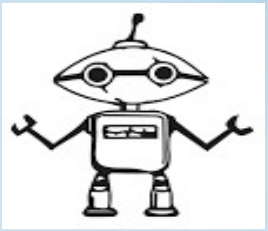
中国地质大学 China University of Geosciences

目 录

Zhaoman, AI,
Department of Computer Science, CS, CUG

3.1 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

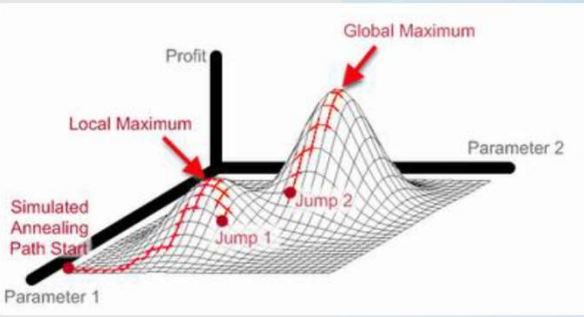
剧烈的摇动（=高温）

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	⇔ Energy level	能量极位
Admissible solution	可接受解	⇔ System state	系统状态
Neighbor solution	相邻解	⇔ Change of state	状态变化
Control parameter	控制参数	⇔ Temperature	温度
Better solution	更优解	⇔ 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← *schedule*(*t*)

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

next← a randomly selected successor of *current*

Δ*E*← *next*.VALUE–*current*.VALUE

if Δ*E*> 0 **then** *current*← *next*

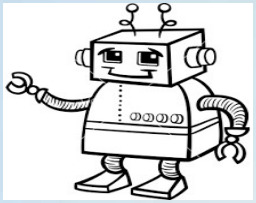
else *current*← *next* only with probability $e^{\Delta E/T}$

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容易被接受，随着时间的推移逐步降低。

模拟退火搜索的性质

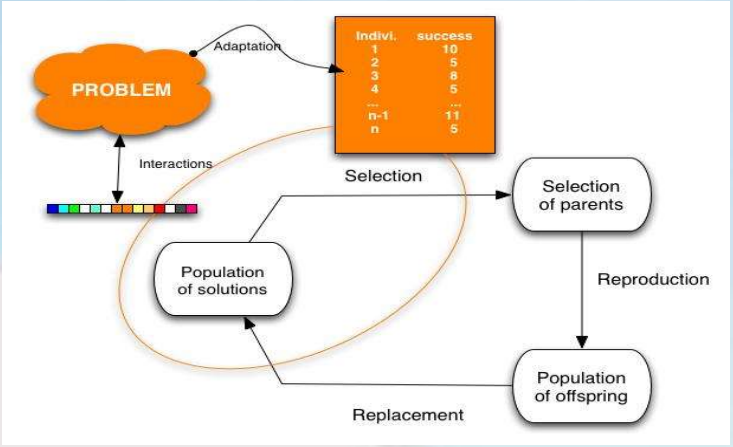
- 可以证明：如果 T 降低的足够慢，则模拟退火能以趋近于1的概率找到最优解。

- 广泛应用于VLSI 布局问题，航空调度等领域



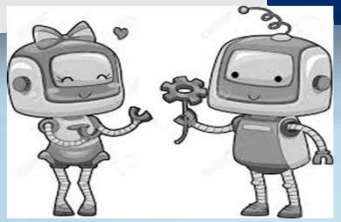
3.2 Genetic Algorithms 遗传算法

- 遗传算法的一些要素是1960年代提出的。通过约翰·霍兰在1970年代早期的工作，尤其是他的《神经元与人工系统的适应性》（1975年）一书，使得遗传算法流行起来。



- **它是一种模仿自然选择过程的搜索启发式算法。**
- **遗传算法**是局部束搜索的变形：与自然选择过程有些相似性，通过把二个父代结合产生后继（有性繁殖），而不是修改单一状态（无性繁殖）。

遗传算法



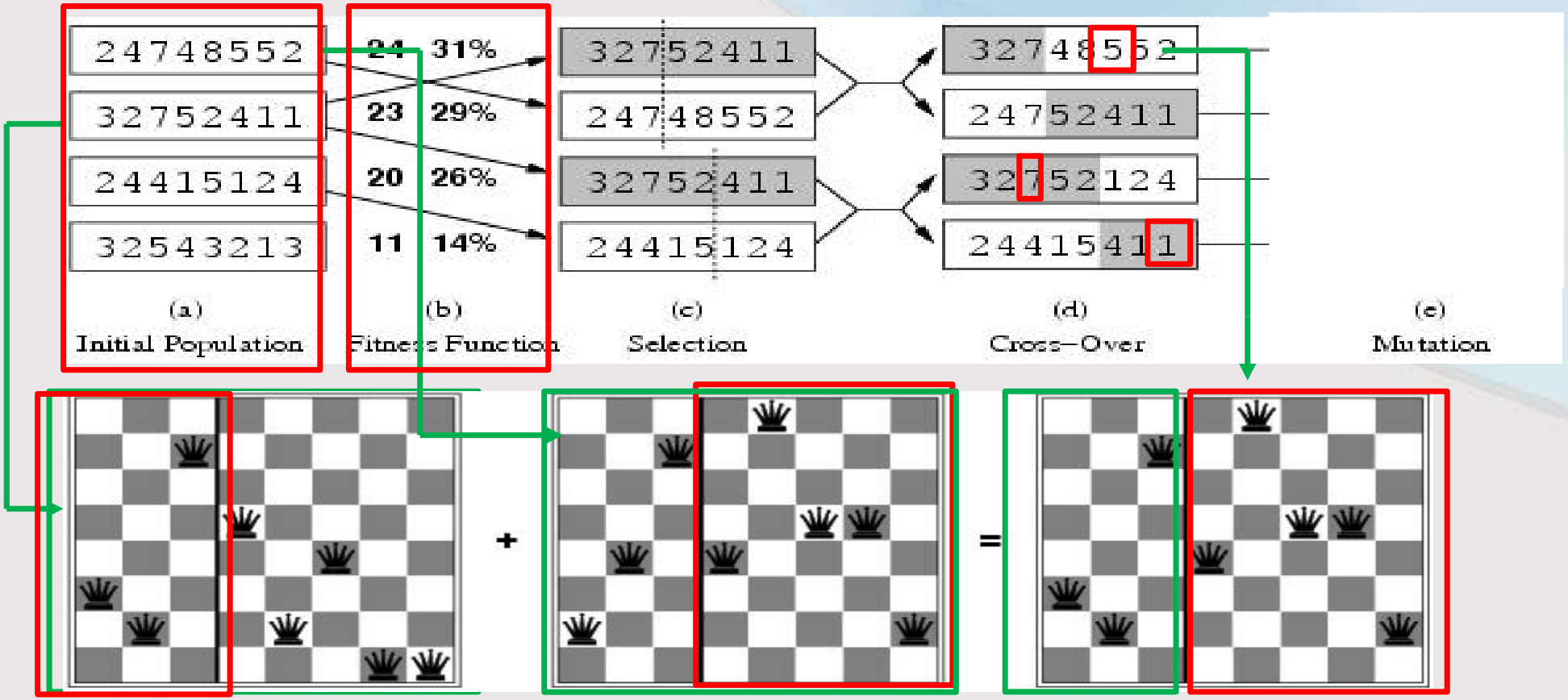
- 遗传算法属于进化算法这个大分类。
- 该算法采用自然进化所派生的技法来生成优化问题的 解，例如

遗传、 突变 、选择、以及杂交(inheritance ， mutation ， selection ， and crossover)

- 通过结合两个状态来产生后续状态
- 该算法开始时从具有一组k 个随机产生的状态开始(种群，population)
- 每个状态（个体）表示成字符串 (染色体)
- 评估函数(适应值函数，fitness function)
- 通过选择，交叉和变异操作来产生新一代种群。

遗传算法思想

- 伴性繁殖的局部束搜索。
- 适应值函数：相互攻击不到的皇后对数 (min = 0, max = 8 × 7/2 = 28)



遗传算法

function GENETIC_ALGORITHM(*population,* FITNESS-FN) **return** an individual

input: *population*, a set of individuals

FITNESS-FN, a function which determines the quality of the individual

repeat

new_population ← empty set

loop for i **from** 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

Applications of Genetic Algorithms 遗传算法的应用

- Bioinformatics 生物信息学
- Computational science 计算科学
- Engineering 工程
- Economics 经济学
- Chemistry 化学
- Manufacturing 制造
- Mathematics 数学
- Physics 物理
- Phylogenetics 种系遗传学
- Pharmacometrics 定量药理学

中国地质大学 China University of Geosciences

目 录

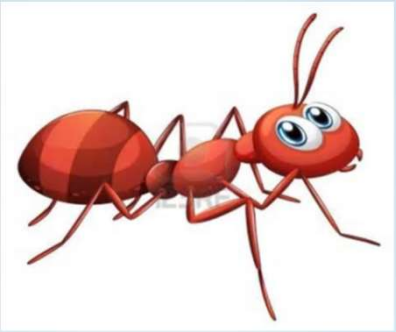
Zhaoman, AI,
Department of Computer Science, CS, CUG

4.1 Ant Colony Optimization (ACO) 蚁群优化

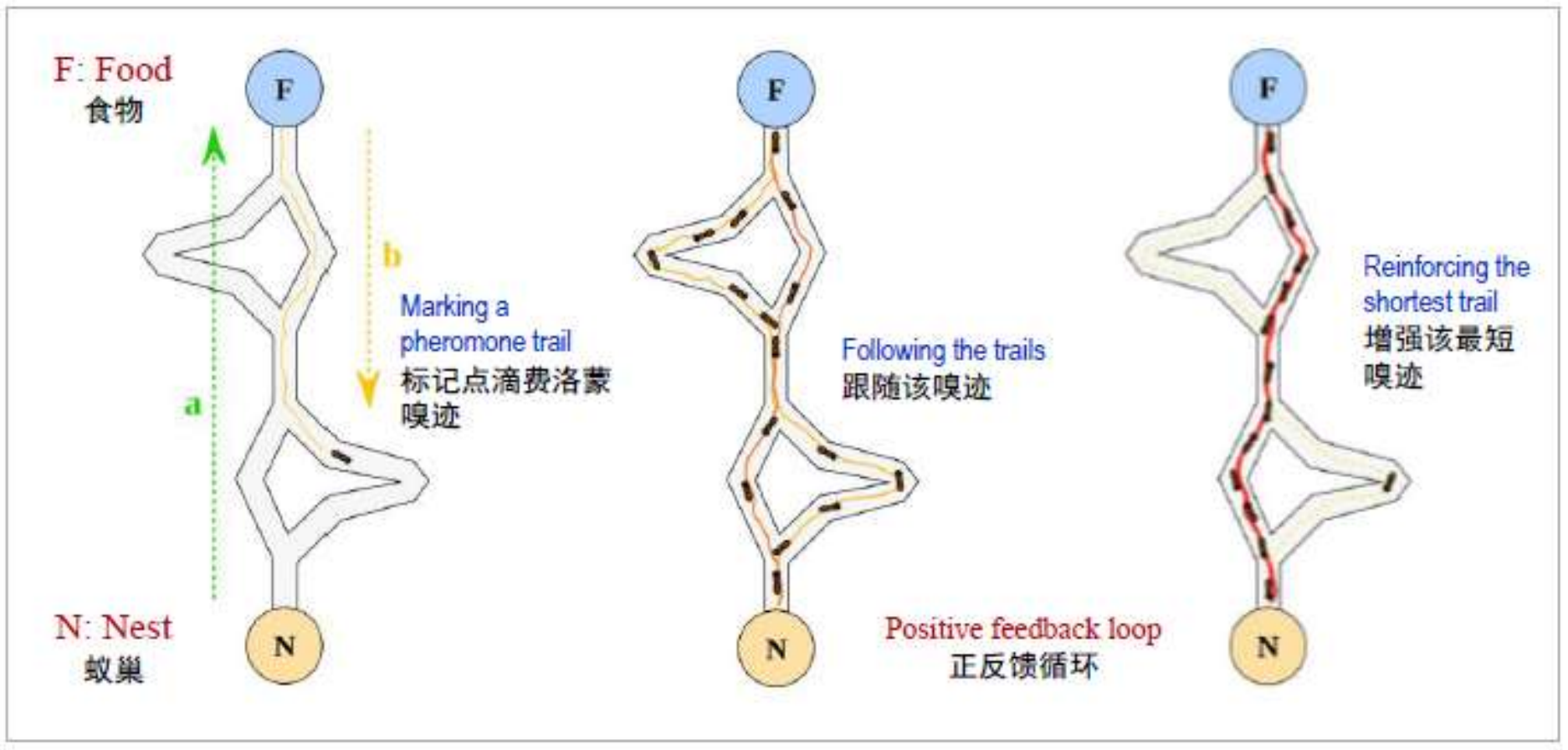
- 它是一种解决计算问题的概率技术，可以用于发现一个图上的最佳路径。
- 最初是由Marco Dorigo于1992年在他的博士论文中提出的。
- 该算法是受蚂蚁在蚁巢和食物源之间寻找路径行为的启发而形成的。

蚁群优化的概念

- 蚂蚁从蚁巢到食物源之间盲目地游荡；
- 最短路径是通过费洛蒙嗅迹(pheromone trails)发现的
- 每个蚂蚁随机地移动
- 费洛蒙就遗留在路径上
- 蚂蚁察觉到前面蚂蚁的路径，跟随而去
- 路径上更多的费洛蒙增加了跟随该路径的概率



蚁群优化的概念

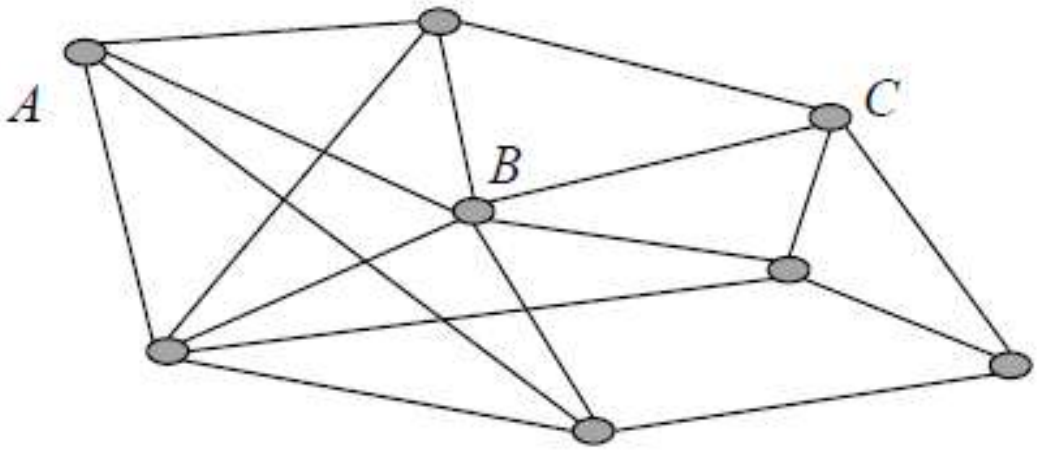

F: Food
食物
F
N
N: Nest
蚁巢
a
b
Marking a
pheromone trail
标记点滴费洛蒙
嗅迹
Following the trails
跟随该嗅迹
Positive feedback loop
正反馈循环
Reinforcing the
shortest trail
增强该最短
嗅迹

Algorithm of Ant Colony Optimization 蚁群优化算法

- 在路径段上积累“虚拟”嗅迹
- 开始时随机选择某个节点
- 随机选择一条路径：基于从初始节点至合适路径上出现嗅迹的量；具有较多嗅迹的路径则具有较高的概率
- 蚂蚁到达下一个节点后，再选择下一个路径
- 重复直到更多的蚂蚁在每个循环中都选择同一个路径

例: Travelling Salesperson Problem (TSP) 旅行推销员问题

- 一个推销员花时间访问 n 个城市。
- 他每次仅访问一个城市，最后回到他出发的地方。
- 他应该按什么顺序访问这些城市才能使距离最短？



	A	B	C	…
A	0	12	34	…
B	12	0	76	…
C	34	76	0	…
…	…	…	…	…

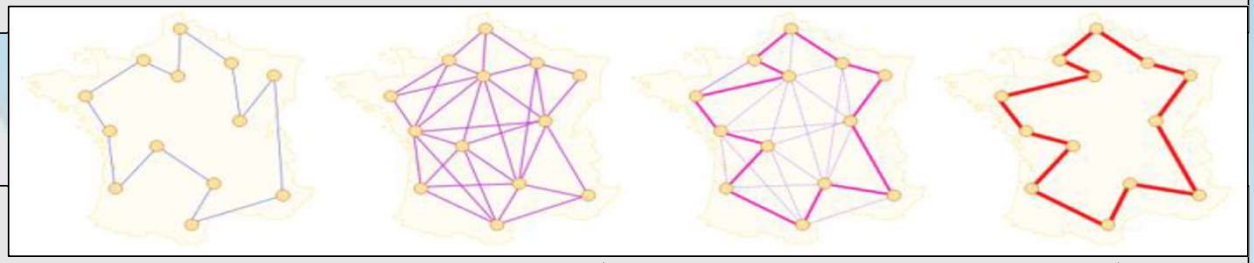
TSP 的要点如下：不是 一个状态空间问题；

“状态 ”= 可能的旅行路线 = (n-1)!/ 2

TSP 是组合优化中的一个**NP** 难问题，在运筹学和理论计算机科学中非常重要。

例: Travelling Salesperson Problem (TSP) 旅行推销员问题

- 最早的蚁群优化算法旨在解决旅行推销员问题，其目标是找到连接所有城市的最短往返旅程。
- 一般的算法相对简单，基于一群蚂蚁，每个都能够沿着这些城市形成一个可能的往返旅程。



算法设计的流程如下:

步骤1：对相关参数进行初始化，包括蚁群规模、信息素因子、启发函数因子、信息素挥发因子、信息素常数、最大迭代次数等，以及将数据读入程序，并进行预处理：比如将城市的坐标信息转换为城市间的距离矩阵。

步骤2：随机将蚂蚁放于不同出发点，对每个蚂蚁计算其下个访问城市，同时进行信息素局部更新，直到有蚂蚁访问完所有城市。

步骤3：计算各蚂蚁经过的路径长度Lk，记录当前迭代次数最优解，同时对路径上的信息素浓度进行全局更新。

步骤4：判断是否达到最大迭代次数，若否，返回步骤2；是，结束程序。

步骤5：输出结果，并根据需要输出寻优过程中的相关指标，如运行时间、收敛迭代次数等。

拓展阅读：https://www.docin.com/p-1448089335.html

Applicationsof Ant Colony Optimization 蚁群优化的应用

- **Scheduling problem** 进度安排问题
- **Vehicle routing problem** 车辆路径问题
- **Assignment problem** 分派问题
- **Device Sizing Problem in Physical Design**
  物理设计中的设备量尺问题
- **Edge Detection in Image Processing Classification**
  图像处理中的边缘检测
- **Data mining** 数据挖掘

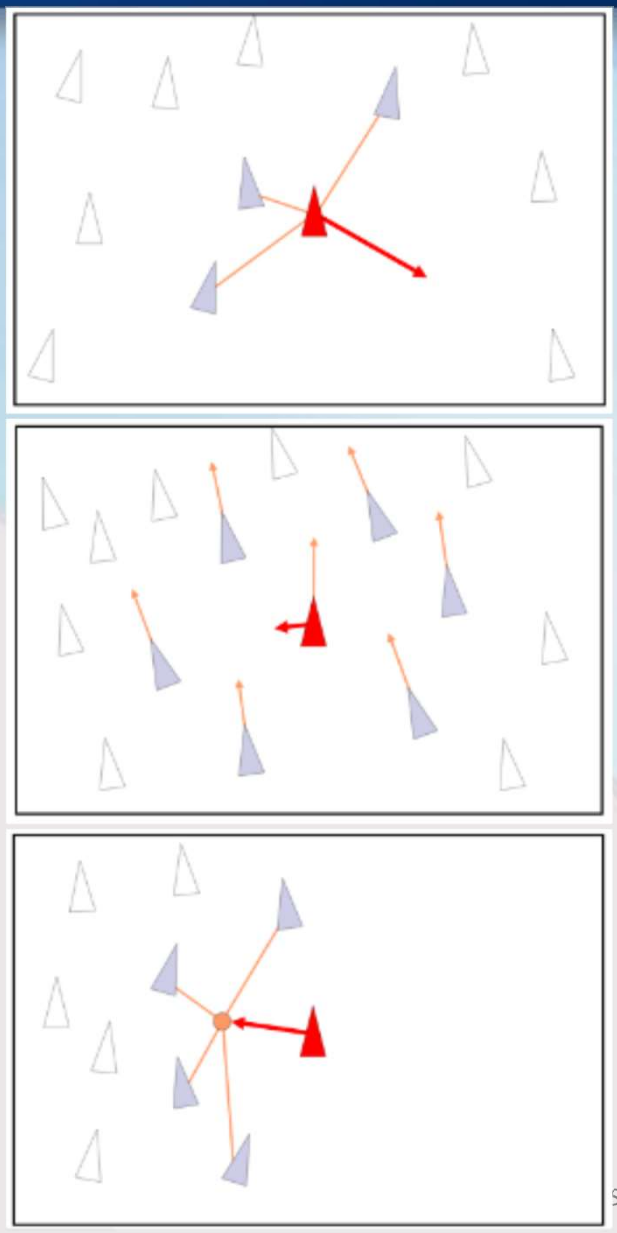
4.2 Particle Swarm Optimization（PSO）粒子群优化

- 由詹姆斯·肯尼迪和拉塞尔·埃伯哈特于1995年提出。受鸟类和鱼类的社会行为的启发。
- 采用若干粒子构成一个围绕搜索空间移动的群体来寻找最优解。
- 搜索空间的每个粒子根据它自己的飞行经验和其它粒子的飞行经验调整它的“飞行”。

Bird Flocking 鸟群

- 一群鸟在一个区域随机地寻找食物。
- 在该被搜索区域仅有一块食物。
- 但它们在经过每次环飞后知道食物有多远。
- 因此发现食物的最好策略是什么？
- 最有效的方法是跟随离食物最近的鸟。

- 仅需三个简单的规则
  - (a) 避免与相邻的鸟碰撞;
  - (b) 保持与相邻的鸟相同的速度 ;
  - (c) 靠近相邻的鸟。



中国地质大学 China University of Geosciences

Algorithm of Particle Swarm Optimization (PSO) 粒子群优化算法

For each *particle*
  Initialize *particle*
Do
  For each *particle*
    Calculate fitness value
    If fitness value > best fitness value (*pBest*) in history //粒子本身所找到最优解
      set fitness value as new *pBest*
  Choose *particle* with best fitness value of all particles as *gBest* //整个种群找到的最优解
  For each *particle*
    Calculate *particle* velocity
    Update *particle* position
While maximum iterations or minimum error criteria is not attained

在找到这两个最优值时,粒子根据如下的公式来更新自己的速度和新的位置：

v[] = v[] + c1 * rand() * (pbest[] - present[]) + c2 * rand() * (gbest[] - present[]) ; present[] = present[] + v[]

v[] 是粒子的速度, present[] 是当前粒子位置. pbest[] and gbest[] 如前定义 rand () 是介于(0, 1)之间的随机数.

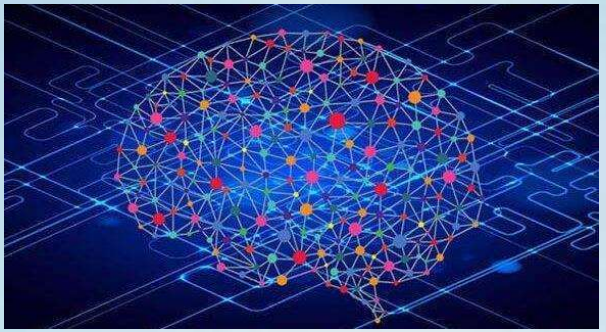
c1, c2 是学习因子. 通常 c1 = c2 =2. (一般c1与c2取值相同，且在0-4之间)

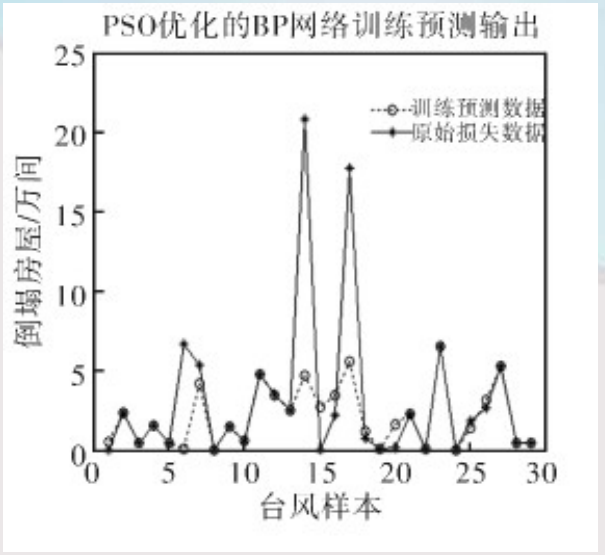
PSO的应用

- Artificial Neural Network (ANN) and PSO

  人工神经网络与PSO

  - ANN 是一种大脑简单模型的计算范型，而反向传播算法是训练 ANN 的最受欢迎的方法之一。
  - 已经有重要的的研究工作，为了改进 ANNs 的一个或多个方面，应用了进化计算计算（evolutionary computation (EC)）。
  - 若干篇论文报告了采用粒子群优化来替代 ANN 中的反向传播学习算法。
  - 论文表明， PSO 是一种训练 ANN 的有前途的方法，它快速并且在多数情况下取得了更好的结果。





中国地质大学 China University of Geosciences

目 录

Zhaoman, AI,
Department of Computer Science, CS, CUG

4.5 小结

- 局部搜索：爬山法在完整状态形式化上进行操作；局部束搜索法保持k个状态的轨迹；禁忌搜索采用一种带约束的邻域搜索。
- 优化与进化算法：模拟退火在大搜索空间逼近全局最优解；遗传算法模仿自然选择的进化过程。
- 群体智能：蚁群优化可以寻找图的最好路径；粒子群优化通过迭代来改善一个候选解。

来自网络：

为了找出地球上最高的山，一群有志气的兔子们开始想办法。

（1）兔子朝着比现在高的地方跳去。他们找到了不远处的最高山峰。但是这座山不一定是珠穆朗玛峰。这就是爬山法，它不能保证局部最优值就是全局最优值。

（2）兔子喝醉了。他随机地跳了很长时间。这期间，它可能走向高处，也可能踏入平地。但是，他渐渐清醒了并朝他踏过的最高方向跳去。这就是模拟退火。

（3）兔子们知道一个兔的力量是渺小的。他们互相转告着，哪里的山已经找过，并且找过的每一座山他们都留下一只兔子做记号。他们制定了下一步去哪里寻找的策略。这就是禁忌搜索。

（4）兔子们吃了失忆药片，并被发射到太空，然后随机落到了地球上的某些地方。他们不知道自己的使命是什么。但是，如果你过几年就杀死一部分海拔低的兔子，多产的兔子们自己就会找到珠穆朗玛峰。这就是遗传算法。

Final Project 参考选题

(1) 求解n皇后问题

- 分别用**爬山法**和**GA算法**求解。
- 要求:

  i . 输入n，并用运行时间比较这**二个算法在相同规模的问题**时的求解效率。

  ii . 比较**同一算法**在n不相同时的运行时间，分析算法的时间复杂性。

(2) 求解TSP问题（采用Romania简化地图，从Bucharest出发并返回）

- 分别用禁忌搜索法和蚁群优化算法求解。
- 要求

  输入地图信息，给出运行结果并分析。