Overview

School of Electronic and Computer Engineering
Peking University

Wang Wenmin

PoAI

Contents

Beyond Classical Search 超越经典搜索

- In previous chapter, we addressed a single category of problems, where the solution is a sequence of actions with following features:

  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：

  - observable,
    可观测、
  - deterministic,
    确定性、
  - known environments.
    已知环境。
- In this chapter, we will discuss:
  - Local search, and Swarm intelligence.

    本章我们将讨论：局部搜索和群体智能。

Classic Search 经典搜索

- The search algorithms we have learned are designed to explore search spaces systematically.

  我们已经学到的搜索算法被设计成系统地探索问题的空间。

  - This systematicity is achieved by keeping one or more paths in memory, and recording which alternatives have been explored at each point along the path.

    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。

  - When a goal is found, the *path* also constitutes a *solution* to the problem.

    目标被找到时，该路径也就构成问题的一个解。

- In many problems, however, the path to the goal is irrelevant.

  然而，在许多问题中，到达目标的路径是无关紧要的。

Local Search 局部搜索

- Local search is a different class of algorithms that do not worry about paths.

  局部搜索是一种不同类型的算法，它不介意什么路径。

- Local search algorithms operate using a single current node (rather than multiple paths), and generally move only to neighbors of that node.

  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。

Local Search 局部搜索

- Typically, the paths followed by the search are not retained.
  通常，搜索后不保留该路径。
- Local search algorithms have two key advantages:
  - use very little memory;
  - can find reasonable solutions in large or infinite (continuous) state spaces.

局部搜索算法有如下两个主要优点：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。

Applications of Local Search 局部搜索的应用

- In many application problems, the path is irrelevant, the goal state itself is the solution, such as:

  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	■	自动规划
telecommunications	■	通讯
network optimization	■	网络优化
vehicle routing	■	车辆路由
portfolio management	■	投资组合管理

Optimization Problem of Local Search 局部搜索的优化问题

- In addition to finding goals, local search algorithms are useful for solving pure optimization problems.

  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。

- The aim in optimization is to find the best state according to an objective function.

  优化的目的是根据一个目标函数找到其最好的状态。

- But many optimization problems do not fit using the search algorithms introduced in previous Chapter.

  但是许多优化问题并不适合采用上一章介绍的搜索算法。

- E.g., Darwinian evolution could be seen as attempting to optimize, but for this problem there is no “goal test”, and no “path cost”.

  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”。

Swarm Intelligence 群体智能

- Study of computational systems inspired by the ‘collective intelligence’.
  研究来自于“集群智能”灵感的计算系统。
- Collective Intelligence emerges through the cooperation of large numbers of homogeneous agents in the environment. Examples: schools of fish, flocks of birds, and colonies of ants.
  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。
- Such intelligence is decentralized, self-organizing and distributed through out an environment.
  这样的智能是分散式、自组织、并且在一个环境内分布。
- In nature, such systems are commonly used to: effective foraging for food, prey evading, colony relocation, etc.
  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

Algorithms of Swarm Intelligence 群体智能的算法

- 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

Typical Algorithms of Swarm Intelligence 群体智能的典型算法

Ant Colony Optimization 蚁群优化

inspired by the behavior of ants, such as:

- stigmergy,
- foraging.

受蚁群的行为所启发，诸如：间接协调、觅食。

Particle Swarm Optimization 粒子群优化

inspired by the social behavior of birds and fishes , such as:

- flocking,
- herding.

受鸟群和鱼群的社会行为所启发，诸如：群集、从众。

Thank you for your attention!

Local Search Algorithms

School of Electronic and Computer Engineering
Peking University

Wang Wenmin

Contents

Hill-Climbing 爬山法

- A mathematical optimization technique which belongs to the family of local search.

  是一种属于局部搜索家族的数学优化方法。

- It is an iterative algorithm:
  - starts with an arbitrary solution to a problem,
  - then incrementally change a single element of the solution,
  - if it's a better solution, the change is made to the new solution,
  - repeating until no further improvements can be found.

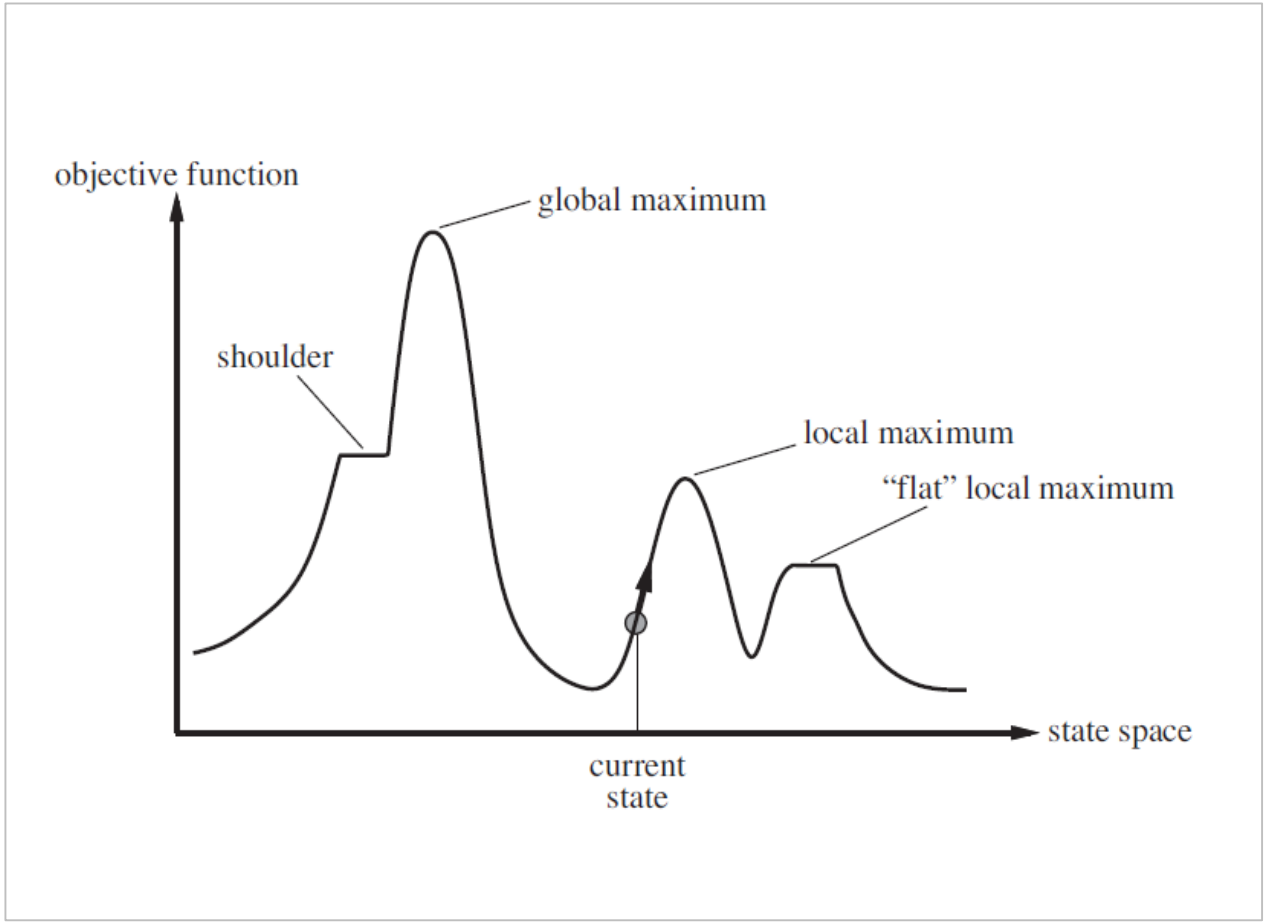
  是一种迭代算法：开始时选择问题的一个任意解，然后递增地修改该解的一个元素，若得到一个更好的解，则将该修改作为新的解；重复直到无法找到进一步的改善。

- Most basic local search algorithm without maintaining a search tree.

  大多数基本的局部搜索算法都不保持一棵搜索树。

State-Space Landscape 状态空间地形图

- It can be explored by one of local search algorithms.
  可通过局部搜索算法对其进行搜索。
- A complete local search algorithm always finds a goal if one exists.
  一个完备的局部搜索算法总能找到一个存在的目标。
- an optimal local search algorithm always finds a global minimum or maximum.
  一个最优的局部搜索算法总能找到一个全局的最小或最大值。



Hill-Climbing Search Algorithm 爬山搜索算法

```
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
```

Steepest-ascent version: at each step, current node is replaced by the best neighbor (the neighbor with highest value), else it reaches a “peak”.

最陡爬坡版：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”。

Hill-Climbing Search Algorithm 爬山搜索算法

- Hill-Climbing search algorithm is the most basic local search technique.
  爬山搜索算法是最基本的局部搜索方法。
- It often makes rapid progress toward a solution, because it is usually quite easy to improve a bad state.
  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。
- It is sometimes called greedy local search, because it grabs a good neighbor state without thinking ahead about where to go next.
  它往往被称为贪婪局部搜索，因为它只顾抓住一个好的邻接点的状态，而不提前思考下一步该去哪儿。
- Although greed is considered one of the "seven deadly sins", it turns out that greedy algorithms often perform quite well.
  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。

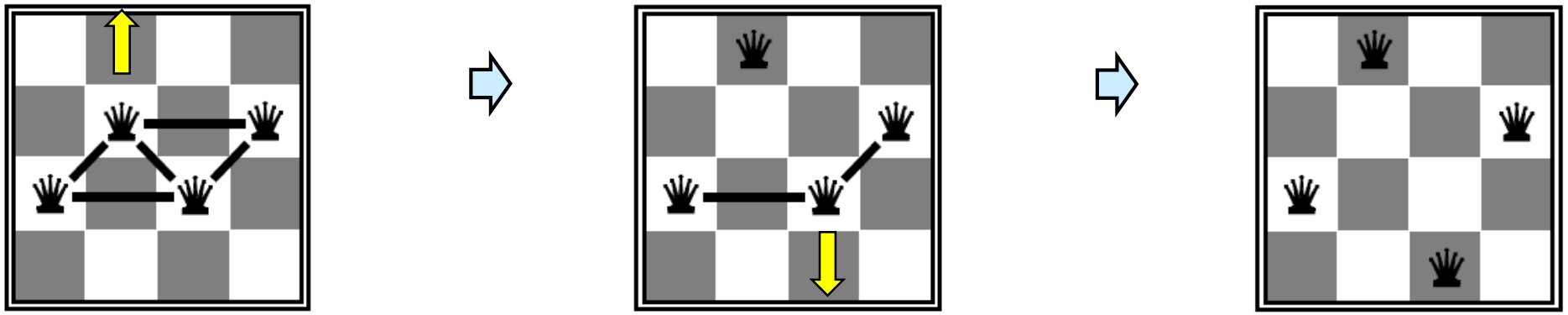
Example: *n*-queens problems *n*皇后问题

- To illustrate hill climbing, we will use the *n*-queens problem. Local search typically use a complete-state formulation.

  为了举例说明爬山法，我们将选用*n*皇后问题。局部搜索通常采用完整状态形式化。

- Put all *n* queens on an *n* x *n* board. Each time move a queen to reduce number of conflicts, to be with no two queens on the same row, column, or diagonal.

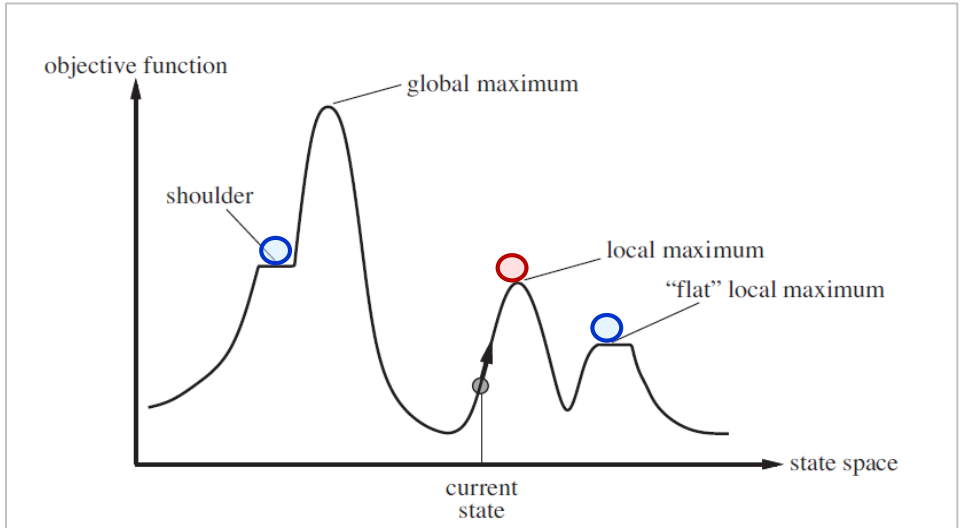
  把n个皇后放在*n* x *n*的棋盘上。每次移动一个皇后来减少冲突数量，使得没有两个皇后在同一行、同一列、或同一对角线上。

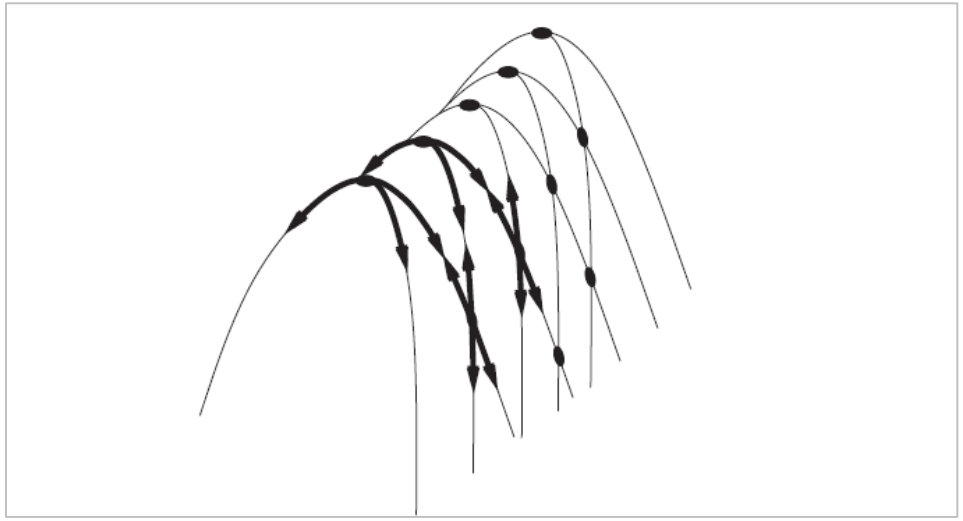


(a) $h = 5$ (b) $h = 2$ (c) $h = 0$

Weaknesses of Hill-Climbing 爬山法的弱点

- It often gets stuck for the three reasons:
  它在如下三种情况下经常被困：
  - Local maxima 局部最大值
    higher than its neighbors but lower than global maximum.
    高于相邻节点但低于全局最大值。
  - Plateaux 高原
    can be a flat local maximum, or a shoulder.
    可能是一个平坦的局部最大值，或山肩。
  - Ridges 山岭
    result in a sequence of local maxima that is very difficult to navigate.
    结果是一系列局部最大值，非常难爬行。





Variants of Hill-Climbing 爬山法的变型

Stochastic hill-climbing 随机爬山法

- It chooses at random among uphill moves; the probability of selection can vary with the steepness of uphill move.
- This usually converges more slowly than steepest ascent.

在向上移动的过程中随机选择；选择的概率随向上移动的斜度而变化。与最陡爬坡相比，收敛速度通常较慢。

First-choice hill-climbing 首选爬山法

- It implements stochastic hill climbing by generating successors randomly until one is generated that is better than the current state.
- This is a good strategy when a state has many of successors.

它通过随机生成后继点来实现随机爬山法，直到生成一个比当前状态好的点。当某个状态有许多后继时，用此策略为好。

Variants of Hill-Climbing 爬山法的变型

Random-restart hill-climbing 随机重启爬山法

- It conducts a series of hill-climbing searches from randomly generated initial states, until a goal is found.
- It is trivially complete with probability approaching 1, because it will eventually generate a goal state as the initial state.
- If each hill-climbing search has a probability p of success, then the expected number of restarts required is $1/p$.

它好于其它爬山搜索方法，从随机生成的初始状态直到找到目标。它十分完备，概率逼近1，因为最终它将生成一个目标状态作为初始状态。如果每次爬山搜索成功的概率为p，则重启需要的期望值是1/p。

It adopts the well-known adage: "If at first you don't succeed, try, try again."

Thank you for your attention!

Local Search Algorithms

School of Electronic and Computer Engineering
Peking University

Wang Wenmin

Contents

Local Beam Search 局部束搜索

- Keeping just one node in memory might seem to be an extreme reaction to the problem of memory limitations.

  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。

- Local beam search keeps track of *k* states rather than just 1.

  而局部束搜索保持*k*个状态而不仅仅为1。

  - It begins with *k* randomly generated states.
  - At each step, all the successors of all *k* states are generated.
  - **If** any one is a goal, the algorithm halts, **else**
    it selects the *k* best successors from the complete list, and repeats.

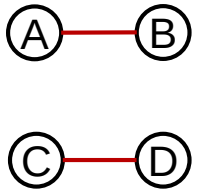
In a local beam search, useful information can be passed among the parallel search threads.

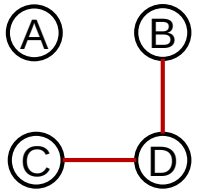
在局部束搜索中，有用的信息能够在并行搜索线程间传递。

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

Keeps track of k states rather than just 1. Start with k randomly generated states. k=2 in this example.

保持k个状态而不仅仅为1。从k个随机生成的状态开始。本例中k=2。

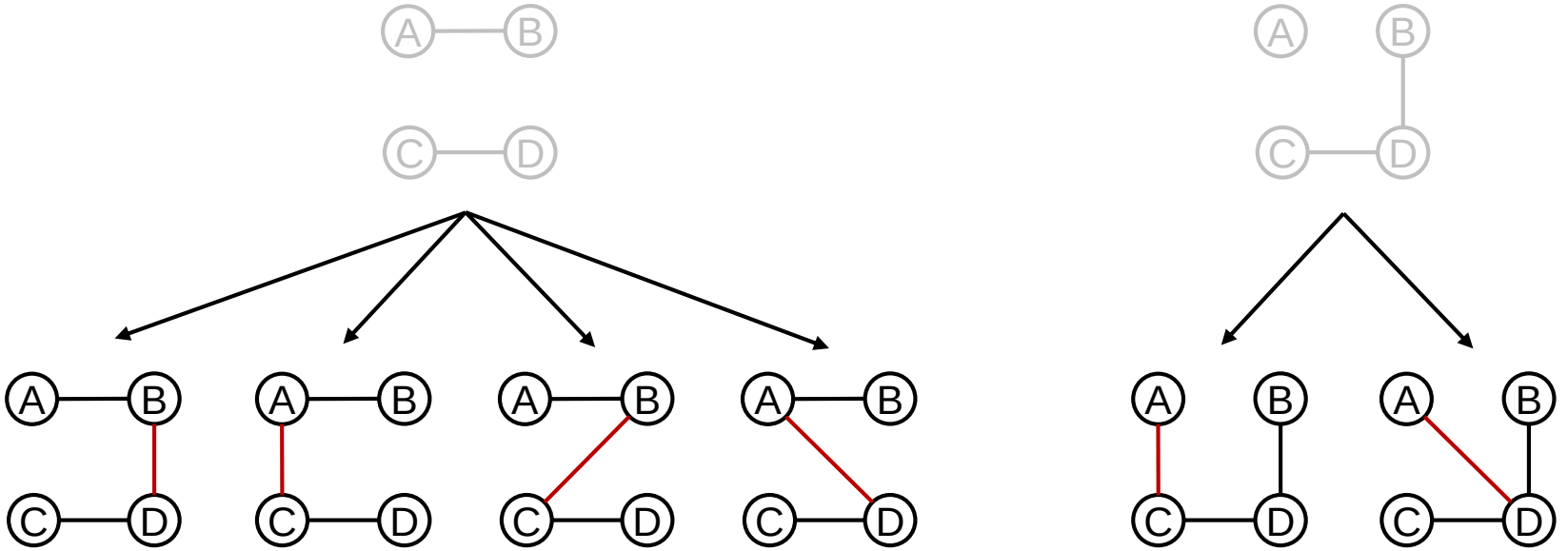




Example: Travelling Salesperson Problem (TSP) 旅行推销员问题

Keeps track of k states rather than just 1. Start with k randomly generated states. k=2 in this example.

保持k个状态而不仅仅为1。从k个随机生成的状态开始。本例中k=2。

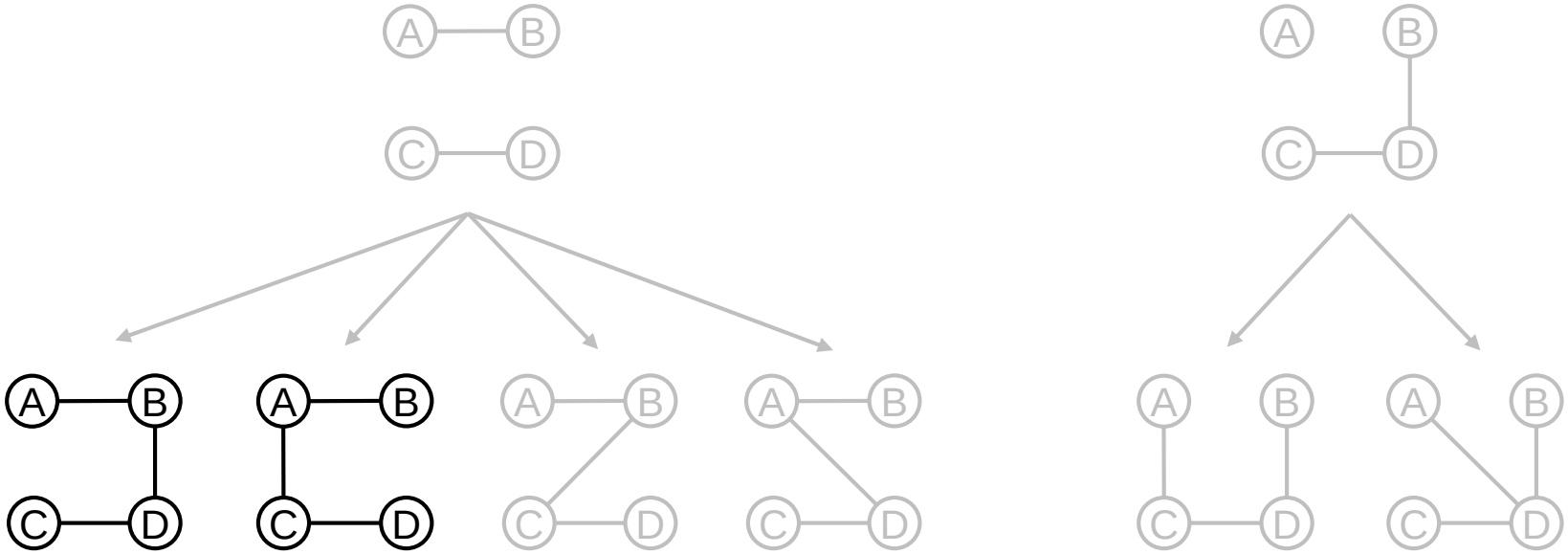


Generate all successors of all the k states. None of these is a goal state so we continue.

生成所有k个状态的全部后继节点。这些后继节点中没有目标状态，故继续下一步。

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

Keeps track of k states rather than just 1.
k=2 in this example. Start with k randomly generated states.



Generate all successors of all the k states. None of these is a goal state so we continue.
生成所有k个状态的全部后继节点。这些后继节点中没有目标状态，故继续下一步。

Select the best k successors from the complete list. Repeat the process until goal found.
从完成表中选择最佳k个后继节点。重复上述过程，直到找到目标。

Variant of Local Beam Search 局部束搜索的变型

Stochastic Beam Search 随机束搜索

- Local beam search may quickly become concentrated in a small region of state space, making the search little more than an expensive version of *hill climbing*.
  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。
- It analogous to *stochastic hill climbing*, helps alleviate this problem.
  它模仿随机爬山法，有助于缓解这个问题。

Variant of Local Beam Search 局部束搜索的变型

Stochastic Beam Search 随机束搜索

- Instead of choosing best k successors, it chooses k successors randomly, with the probability of choosing a successor being an increasing function of its value.

  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。

- Stochastic beam search bears some similar to the process of natural selection.

  随机束搜索有些类似于自然选择的过程。

Thank you for your attention!

4.2. Local Search Algorithms

Contents:

Tabu Search 禁忌搜索

- Tabu, indicates things that cannot be touched.
  禁忌，指的是不能触及的事物。
- Tabu search is created by Fred Glover in 1986 and formalized in 1989.
  禁忌搜索是由弗雷德·格洛弗于1986年提出，1989年加以形式化。
- It is a *meta-heuristic* algorithm, used for solving combinatorial optimization problems.
  它是一种元启发式算法，用于解决组合优化问题。
- It uses a local or neighborhood search procedure, to iteratively move from one potential solution x to an improved neighborhood solution x', until some stopping condition has been satisfied.
  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。
- The memory structure to determine the solutions is called tabu list.
  用于确定解的数据结构被称为禁忌表。

Three Strategies of Tabu Search 禁忌搜索的三种策略

- **Forbidding strategy** 禁止策略

  control what enters the tabu list.

  控制何物进入该禁忌表。

- **Freeing strategy** 释放策略

  control what exits the tabu list 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。

Tabu Search Algorithm 禁忌搜索算法

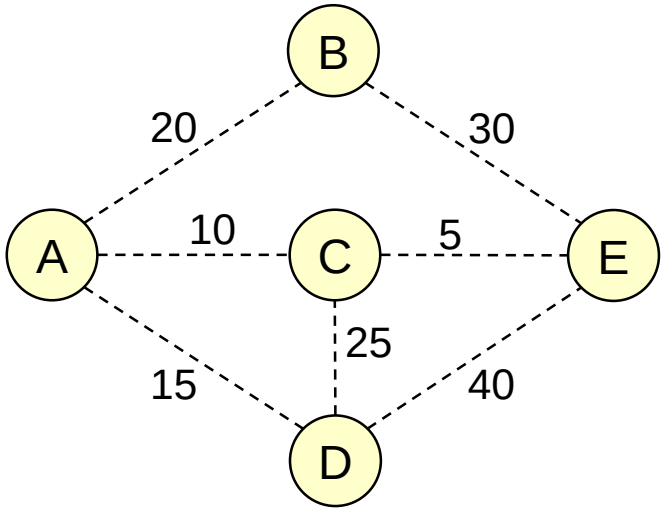
```
function TABU-SEARCH (s') return a best candidate
    sBest ← s ← s'
    tabuList ← null list
    while (not STOPPING-CONDITION())
        candidateList ← null list
        bestCandidate ← null
        for (sCandidate in sNeighborhood)
            if ((not tabuList.CONTAINS(sCandidate))
        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

Problems Can be Solved by Tabu Search 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	■	作业车间调度问题
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

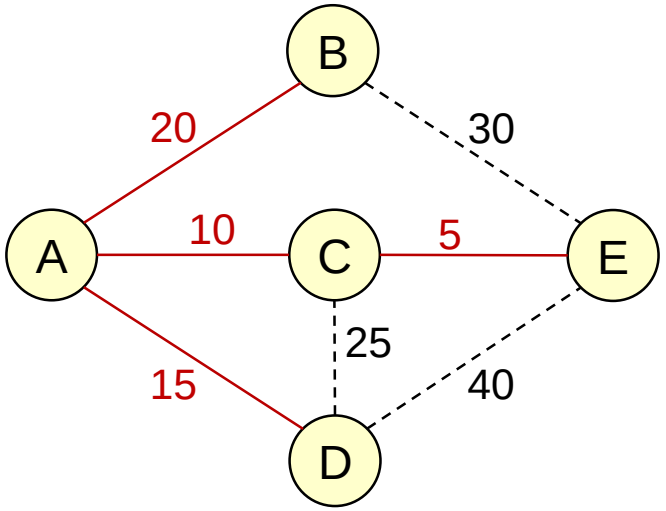
Minimum Spanning Tree Problem 最小生成树问题

Objective 目标



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

Constraints 1: Link AD can be included only if link DE also is included. (Penalty:100)

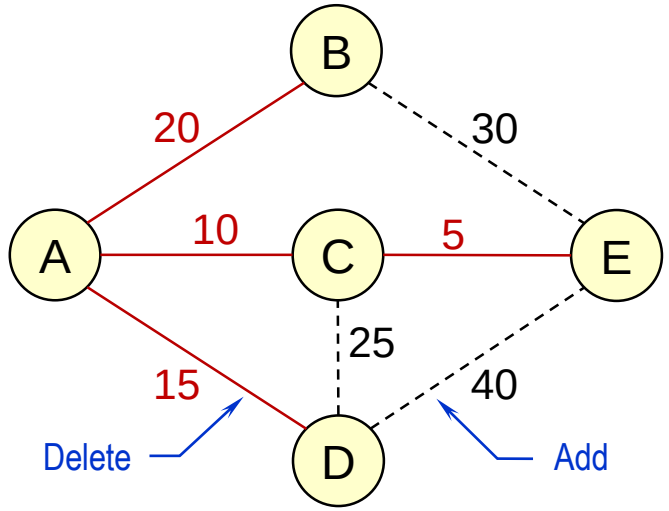
约束1：仅当包含连接DE时，才可以包含连接AD。（处罚：100）

Constraints 2: At most one of the three links (AD, CD, and AB) can be included. (Penalty: 100 if selected two of the three, 200 if selected all three.)

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



Cost = 50 + 200 (constraint penalty)

代价 = 50 + 200（约束处罚）

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

Constraints 1: Link AD can be included only if link DE also is included. (Penalty:100)

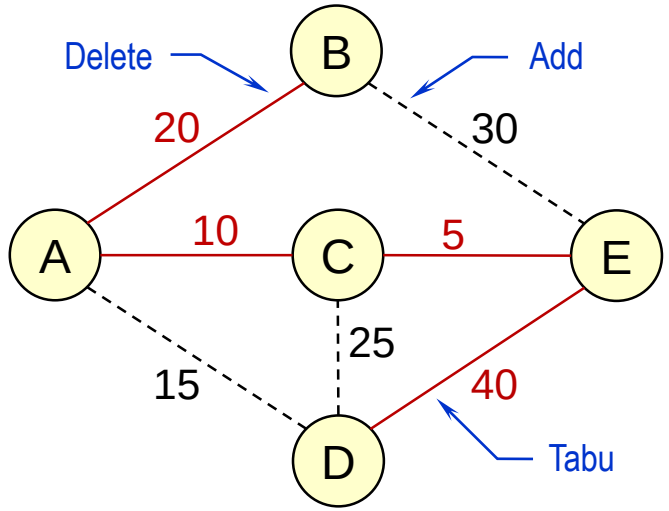
约束1：仅当包含连接DE时，才可以包含连接AD。（处罚：100）

Constraints 2: At most one of the three links (AD, CD, and AB) can be included. (Penalty: 100 if selected two of the three, 200 if selected all three.)

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



Cost = 75, Tabu list: DE

代价 = 75，禁忌表：DE

Escape local optimum

溢出局部最优

Add	Delete	Cost
AD AD AD	DE* CE AC	Tabu move 85 + 100 = 185 80 + 100 = 180
BE BE BE	CE AC AB	100 + 0 = 100 95 + 0 = 95 85 + 0 = 85
CD CD	DE* CE	Tabu move 95 + 100 = 195

New Cost = 85

新代价 = 85

Constraints 1: Link AD can be included only if link DE also is included. (Penalty:100)

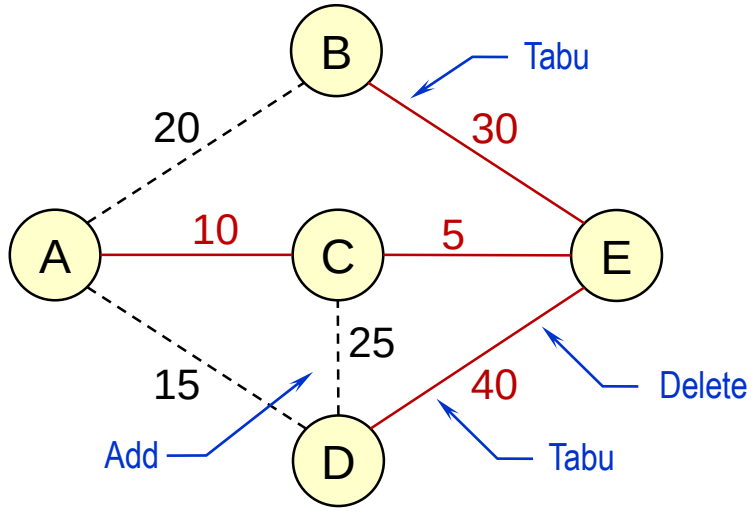
约束1：仅当包含连接DE时，才可以包含连接AD。（处罚：100）

Constraints 2: At most one of the three links (AD, CD, and AB) can be included. (Penalty: 100 if selected two of the three, 200 if selected all three.)

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



Cost = 85, Tabu list: DE & BE

代价 = 85，禁忌表：DE & BE

Override tabu status

覆盖禁忌状态

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

New Cost = 70

新代价 = 70

Constraints 1: Link AD can be included only if link DE also is included. (Penalty:100)

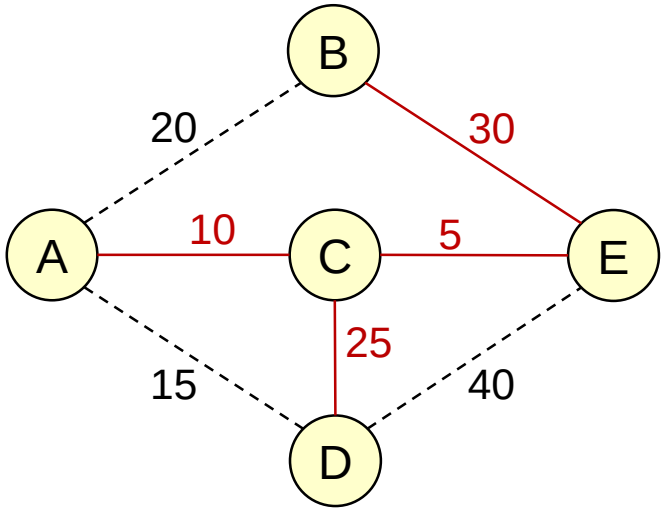
约束1：仅当包含连接DE时，才可以包含连接AD。（处罚：100）

Constraints 2: At most one of the three links (AD, CD, and AB) can be included. (Penalty: 100 if selected two of the three, 200 if selected all three.)

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 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

Constraints 1: Link AD can be included only if link DE also is included. (Penalty:100)

约束1：仅当包含连接DE时，才可以包含连接AD。（处罚：100）

Constraints 2: At most one of the three links (AD, CD, and AB) can be included. (Penalty: 100 if selected two of the three, 200 if selected all three.)

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）

Application Fields of Tabu Search 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	物流
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

Optimization and Evolutionary Algorithms

School of Electronic and Computer Engineering
Peking University

Wang Wenmin

PoAI

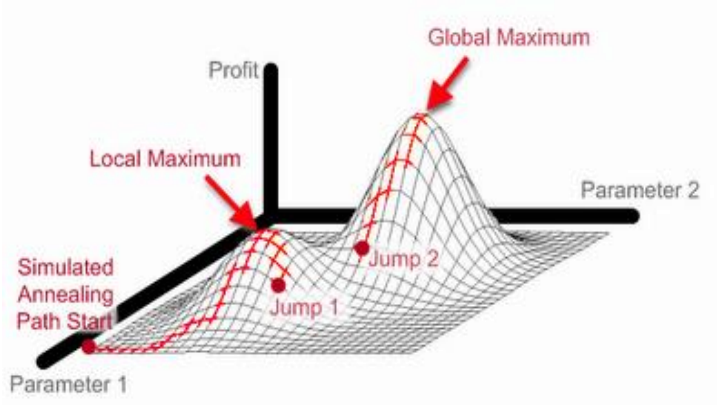
Contents

Annealing 退火

- In metallurgy, annealing is used to temper or harden metals and glass.
  在冶金学中，退火用于对金属和玻璃进行回火或硬化。
- A solid is heated in a hot bath, increasing the temperature up to a maximum value. At this temperature, all material is in liquid state and the particles arrange themselves randomly.
  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。
- As the temperature of the hot bath is cooled gradually, all the particles of this structure will be arranged in the state of lower energy.
  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。

Simulated Annealing 模拟退火

- Simulated annealing is a probabilistic technique for approximating the global optimum of a given function. Proposed in 1953.
  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。
- Specifically, it is a meta-heuristic to approximate global optimization in a large search space.
  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



Source: http://www.maxdama.com

Simulated Annealing Algorithm 模拟退火算法

- **Initial Solution** 初始解

  Generated using an heuristic. Chosen at random.

  使用启发式方法生成。随机选择。

- **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。

Simulated Annealing Algorithm 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**
    $T \leftarrow schedule(t)$
    if $T = 0$ **then return** *current*
    next ← a randomly selected successor of *current*
    $\Delta E \leftarrow next.\text{VALUE} - current.\text{VALUE}$
    if $\Delta E > 0$ **then** *current* ← *next*
    else *current* ← *next* only with probability $e^{\Delta E/T}$

A version of stochastic hill climbing where some downhill moves are allowed.
Downhill moves are accepted readily early in the annealing schedule and then less often as time goes on.
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。

PoAI

Contents

- 4.3.1. Simulated Annealing
- 4.3.2. Genetic Algorithms

Genetic Algorithms 遗传算法

- The elements of genetic algorithms was introduced in 1960s. Became popular through the work of John Holland in early 1970s, and particularly his book *Adaptation in Natural and Artificial Systems* (1975).

  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。

- It is a *search heuristic* that mimics the process of natural selection.

  它是一种模仿自然选择过程的搜索启发式算法。

- The algorithm is a variant of stochastic beam search, in which successor states are generated by combining *two parent* states rather than by modifying a single state. It is dealing with *sexual reproduction* rather than asexual reproduction.

  该算法是随机束搜索的一个变型，其中后继节点是由两个父辈状态的组合而不是修改单一状态生成的。其处理过程是有性繁殖，而不是无性繁殖。

Genetic Algorithms 遗传算法

- Genetic algorithms belong to the larger class of *evolutionary algorithms*.遗传算法属于进化算法这个大分类。
- The algorithms generate solutions to optimization problems using techniques inspired by natural evolution, such as

  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

  - inheritance, mutation, selection, and crossover.

    遗传、突变、选择、以及杂交。
- It begin with a set of k randomly generated states, called the population.

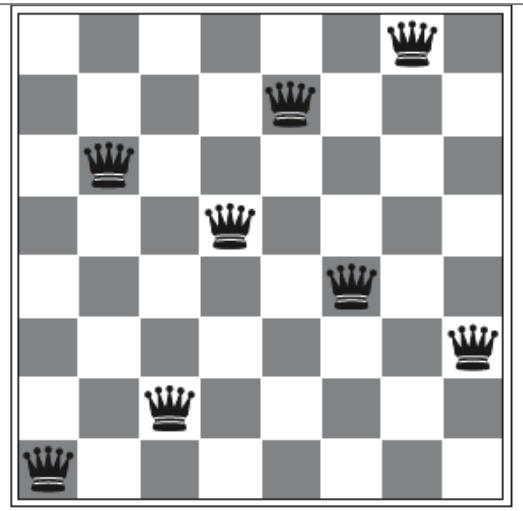
  该算法开始时具有一组k个随机生成的状态，称其为种群。
- Each state (individual) is represented as a string over a finite alphabet, most commonly, a string of 0s and 1s.

  每个状态（个体）表示为有限字母表上的一个字符串，通常是0和1的字符串。

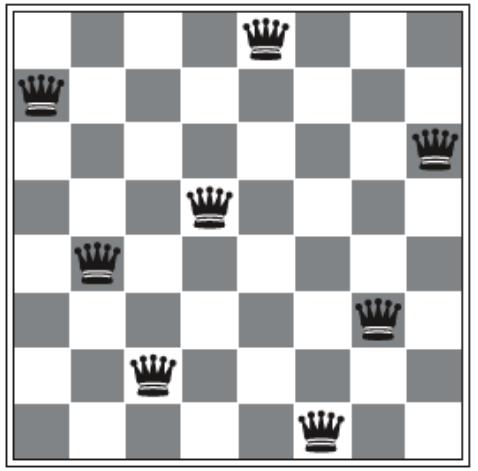
Example: 8-queens problem 8皇后问题

- An 8-queens state must specify the positions of 8 queens, each in a column of 8 squares, the state could be represented as 8 digits, each in the range from 1 to 8.

  某8皇后状态需要指明8个皇后的位置，每个位于8个方格的一列，其状态可用8个数字表示，每个位于1到8之间。

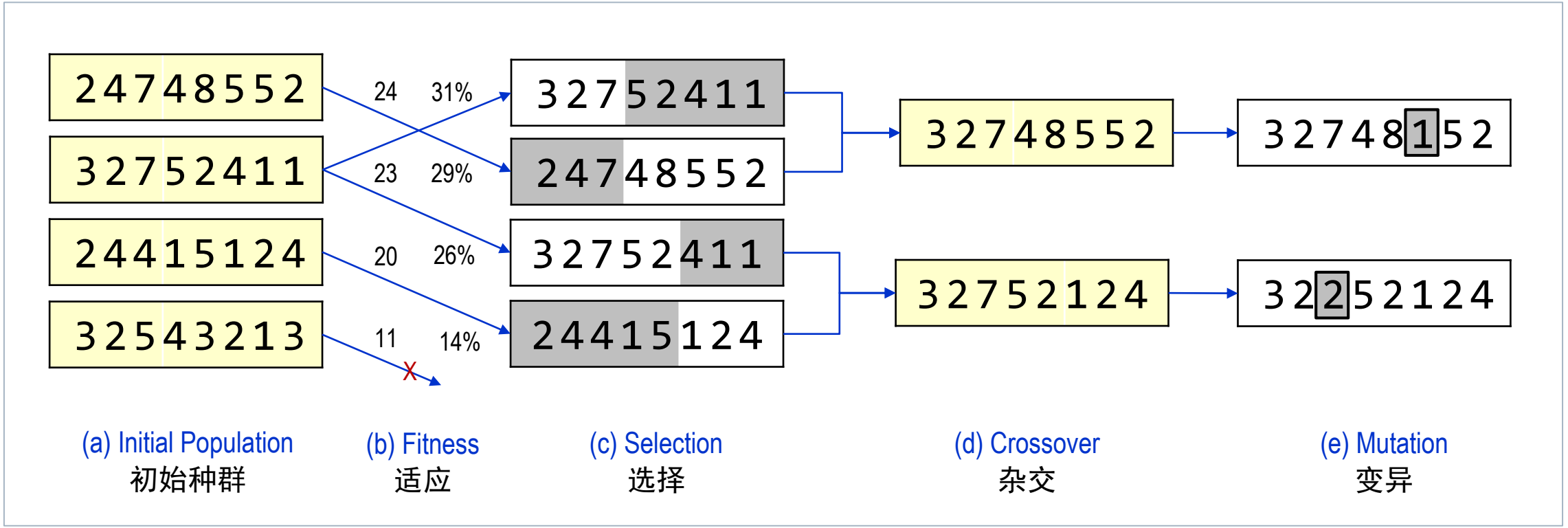


16257483



74258136

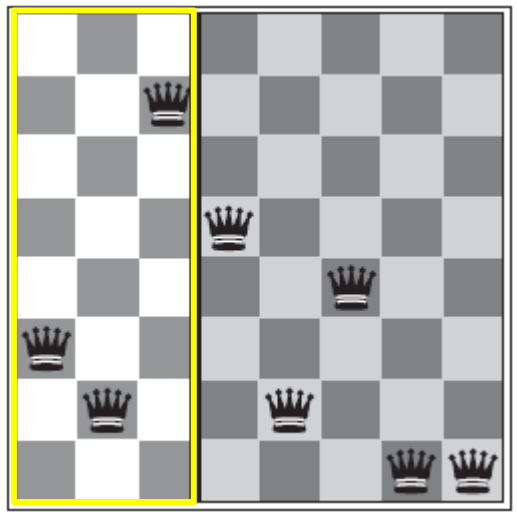
Example: 8-queens problem 8皇后问题



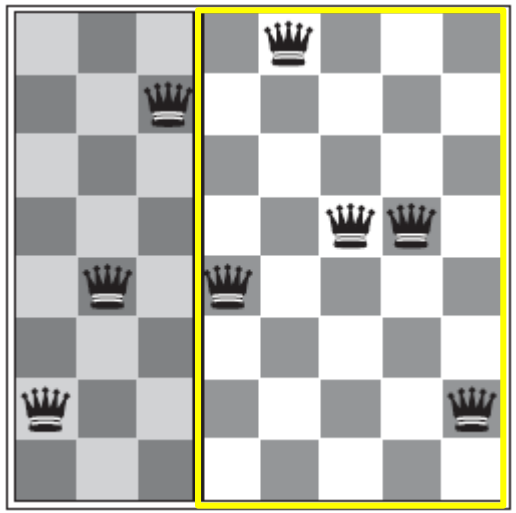
Digit strings representing 8-queens states. (a) the initial population, (b) ranked by the fitness function, (c) resulting in pairs for mating, (d) reproduce child, (e) subject to mutation.

数字串表示8皇后的状态。(a) 为初始种群，(b) 通过适应函数进行分级，(c) 导致交配对产生，(d) 繁殖后代，(e) 取决于突变。

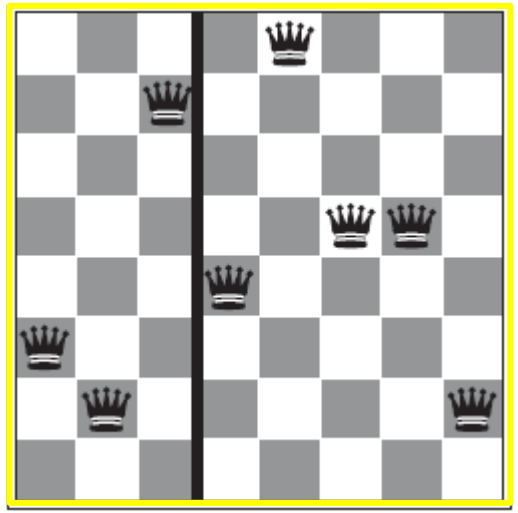
Example: 8-queens problem 8皇后问题



+



=



32752411

24748552

32748552

Those three 8-queens states correspond to the first two parents in “Selection” and their offspring in “Crossover”.
The shaded columns are lost in the crossover step and the unshaded columns are retained.

这三个8皇后的状态分别对应于“选择”中的两个父辈和“杂交”中它们的后代。
阴影的若干列在杂交步骤中被丢掉，而无阴影的若干列则被保留下来。

The Genetic Algorithm 遗传算法

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

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

Thank you for your attention!

Swarm Intelligence

School of Electronic and Computer Engineering
Peking University

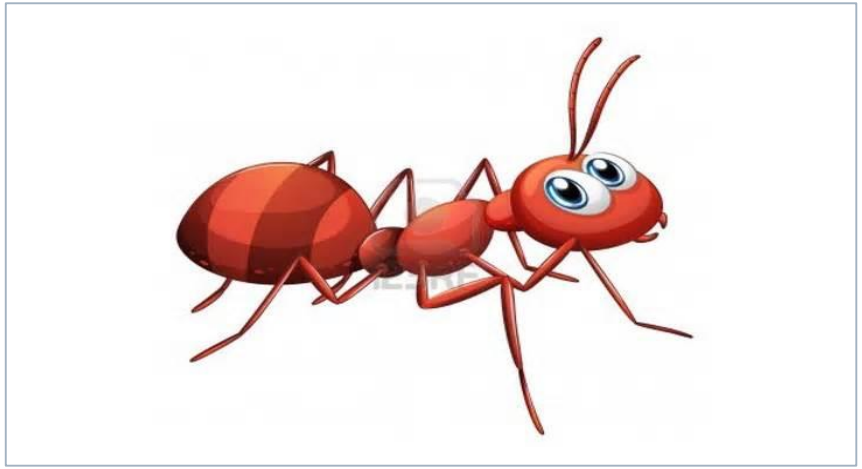
Wang Wenmin

PoAI

Contents

Ant Colony Optimization (ACO) 蚁群优化

- It is a probabilistic technique for solving computational problems which can be used to finding optimal paths in a graph.
  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。
- Initially was proposed by Marco Dorigo in 1992 in his PhD thesis.
  最初是由Marco Dorigo于1992年在他的博士论文中提出的。
- The algorithm was inspired by the behavior of ants seeking a path between their nest and a source of food.
  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。



Concept of Ant Colony Optimization 蚁群优化的概念

- Ants navigate from nest to food source blindly:

  蚂蚁从蚁巢到食物源之间盲目地游荡：

  - Shortest path is discovered via pheromone trails

    最短路径是通过费洛蒙嗅迹发现的

  - Each ant moves at random

    每个蚂蚁随机地移动

  - Pheromone is deposited on path

    费洛蒙就遗留在路径上

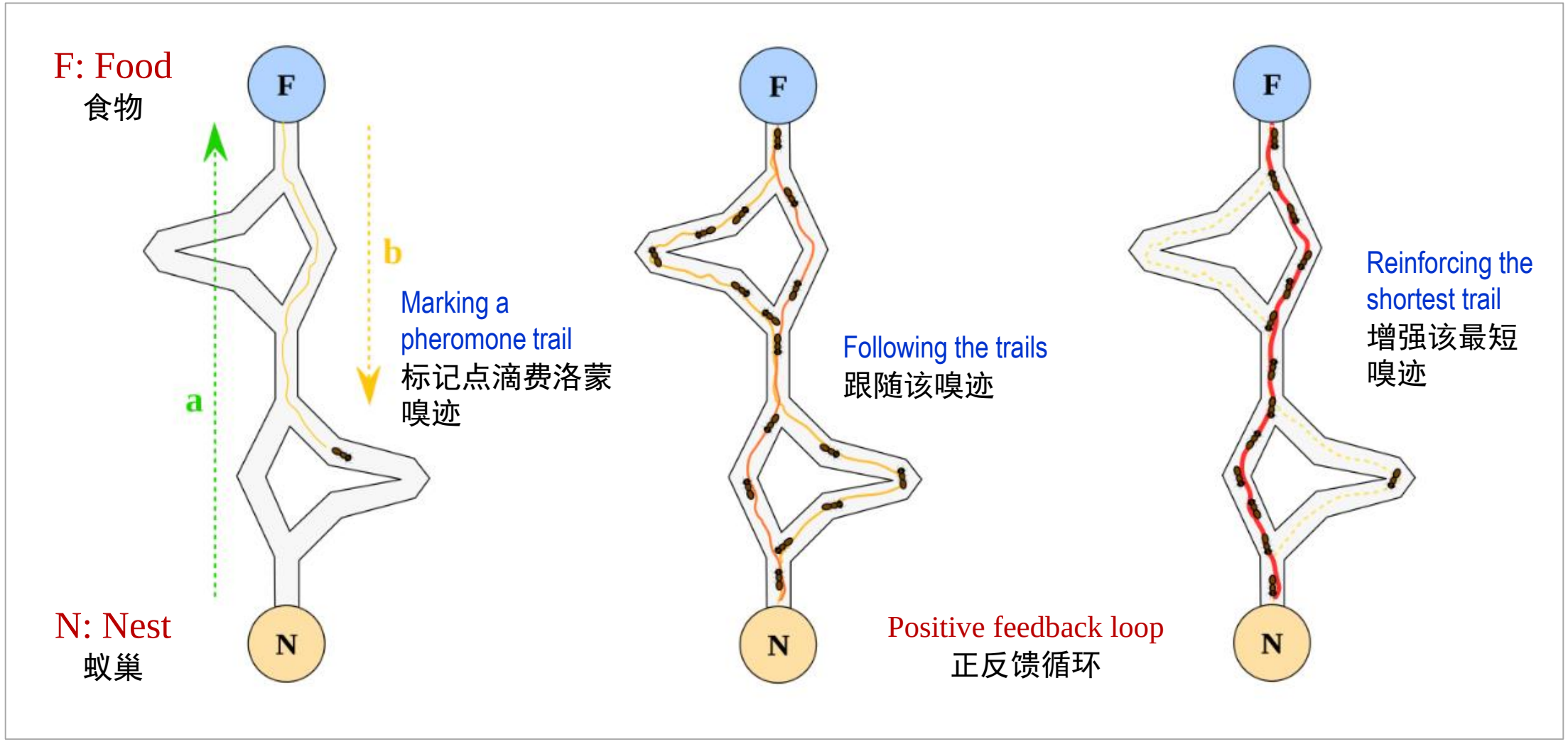
  - Ants detect lead ant's path, inclined to follow

    蚂蚁察觉到前面蚂蚁的路径，跟随而去

  - More pheromone on path increases probability of path being followed

    路径上更多的费洛蒙增加了跟随该路径的概率

Concept of Ant Colony Optimization 蚁群优化的概念

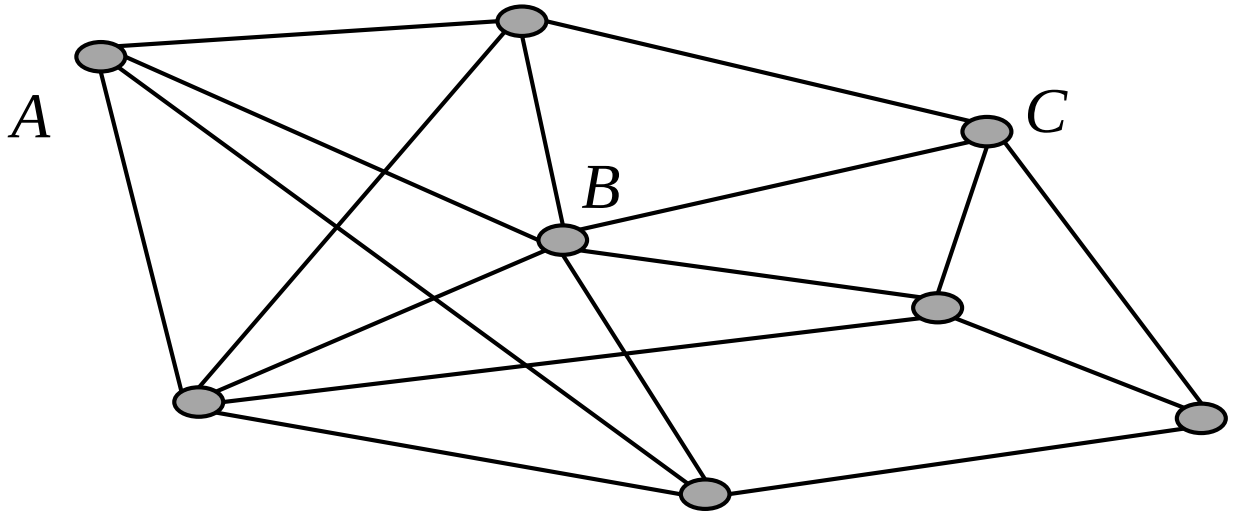


Algorithm of Ant Colony Optimization 蚁群优化算法

- Virtual “trail” accumulated on path segments
  在路径段上积累“虚拟”嗅迹
- Starting a node selected at random
  开始时随机选择某个节点
- The path selected at random: based on amount of “trail” present on possible paths from starting node; higher probability for paths with more “trail”
  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
- Ant reaches next node, selects next path
  蚂蚁到达下一个节点后，再选择下一个路径
- Repeated until most ants select the same path on every cycle
  重复直到更多的蚂蚁在每个循环中都选择同一个路径

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

- A salesman spends his time visiting n cities.
  一个推销员花时间访问n个城市。



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

- He visits each city just once and finishes up where he started.
  他每次仅访问一个城市，最后回到他出发的地方。
- In what order should he visit them to minimize the distance?
  他应该按什么顺序访问这些城市才能使距离最短？

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

- Key points for TSP are as following:

  TSP的要点如下：

  - not a state-space problem

    不是一个状态空间问题

  - "states" = possible tours = $(n\text{-}1)!/2$

    "状态" = 可能的旅行路线 = $(n-1)!/2$

- TSP is an NP-hard problem in combinatorial optimization, important in operations research and theoretical computer science.

  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。

- TSP is a special case of the travelling purchaser problem (TPP).

  TSP是旅行采购员问题（TPP）的一个特例。

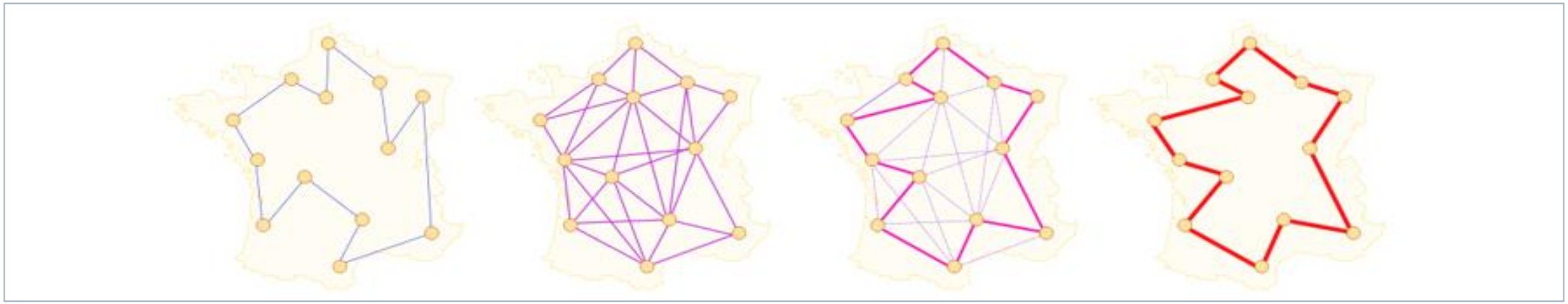
Example: Travelling Salesperson Problem (TSP) 旅行推销员问题

- The first Ant Colony Optimization algorithm was aimed to solve the *Travelling Salesperson Problem*, in which the goal is to find the shortest round-trip to link a series of cities.

  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。

- The general algorithm is relatively simple and based on a set of ants, each making one of the possible round-trips along the cities.

  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。



Applications of Ant Colony Optimization 蚁群优化的应用

- Have been applied to many combinatorial optimization problems:
  已经被用于许多组合优化问题：

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	
	分类
	数据挖掘

PoAI

Contents

Particle Swarm Optimization (PSO) 粒子群优化

- Proposed by James Kennedy & Russell Eberhart in 1995. Inspired by social behavior of birds and fishes.

  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。

- Uses a number of particles that constitute a swarm moving around in the search space looking for the best solution.

  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。

- Each particle in search space adjusts its “flying” according to its own flying experience as well as the flying experience of other particles.

  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

- A group of birds are randomly searching food in an area.
  一群鸟在一个区域随机地寻找食物。
- There is only one piece of food in the area being searched.
  在该被搜索区域仅有一块食物。
- All the birds do not know where the food is.
  所有的鸟都不知道食物在哪儿。
- But they know how far the food is in each iteration.
  但它们在经过每次环飞后知道食物有多远。
- So what's the best strategy to find the food?
  因此发现食物的最好策略是什么？
- The effective one is to follow the bird which is nearest to the food.
  最有效的方法是跟随离食物最近的鸟。

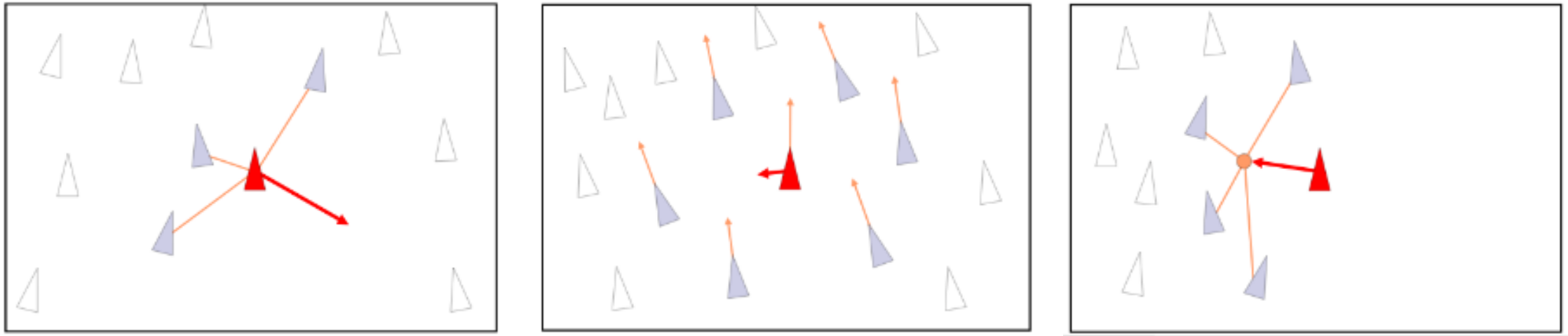
Bird Flocking 鸟群

- Only three simple rules

  仅需三个简单的规则

  - (a) Avoid collision with neighboring birds; (b) Match the velocity of neighboring birds; (c) Stay near neighboring birds.

    （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。



(a) (b) (c)

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
            set fitness value as new *pBest*
    Choose *particle* with best fitness value of all particles as *gBest*
    For each *particle*
        Calculate *particle* velocity
        Update *particle* position
While maximum iterations or minimum error criteria is not attained

Artificial Neural Network (ANN) and PSO

- An ANN is a computing paradigm that is a simple model of the brain, and the back-propagation algorithm is the one of the most popular method to train the ANN.

  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。

- There have been significant research efforts to apply evolutionary computation (EC) techniques for the purposes of evolving one or more aspects of ANNs.

  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（EC）。

Artificial Neural Network (ANN) and PSO

- Several papers reported using PSO to replace the back-propagation learning algorithm in ANN.

  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。

- It showed PSO is a promising method to train ANN. It is faster and gets better results in most cases.

  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。

Thank you for your attention!