中山大學 SUN YAT-SEN UNIVERSITY

Lecture 5
Metaheuristic Algorithms (II)

Algorithm

张子臻，中山大学计算机学院

zhangzizhen@gmail.com

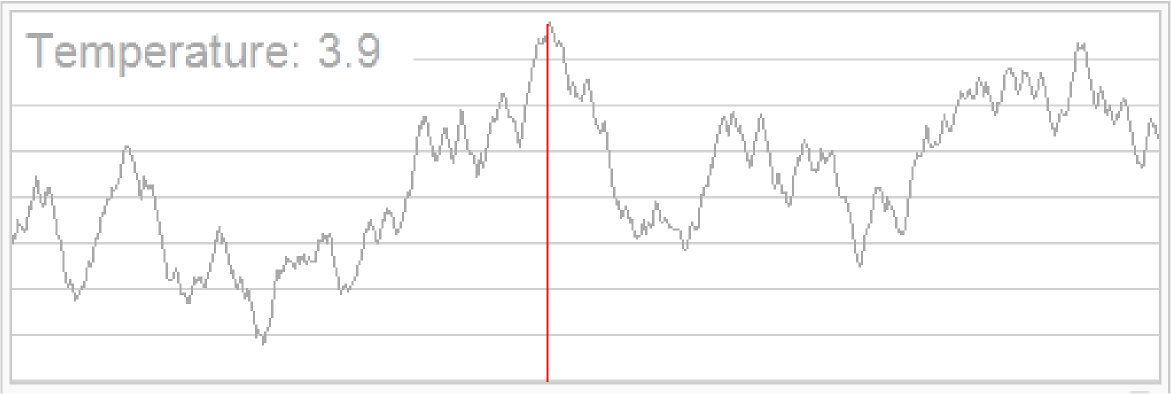
Outline

- Single-Solution Based Metaheuristics
  - Simulated Annealing (模拟退火)
  - Tabu Search (禁忌搜索)
- Population Based Metaheuristics
  - Genetic Algorithm (遗传算法)
  - Ant Colony Optimization (蚁群算法)
  - Particle Swarm Optimization (粒子群算法)
- Summary of Metaheuristics

Simulated Annealing (SA)

模拟退火算法（SA）在图分割和VLSI设计等领域得到了应用。

- In the pioneering works, SA has been applied to graph partitioning and VLSI design.
- Simple and efficient in solving combinatorial optimization problems. 它在解决组合优化问题方面简单高效。
- It has been extended to deal with continuous optimization problems. SA已经被拓展应用于连续优化问题。
- SA is based on the principles of statistical mechanics whereby the annealing process requires heating and then slowly cooling a substance to obtain a strong crystalline structure. SA基于统计力学原理，其退火过程需要将物质加热，然后缓慢冷却，以获取强有力的晶体结构。



Description of SA

- At each iteration, a random neighbor *s'* is generated. 每次迭代时，会生成一个随机的相邻状态 s'
- Moves that improve the cost function are always accepted. 对于改善成本函数的移动，始终会被接受。
- Otherwise, the neighbor is selected with a given probability: (important!)

否则，以一定的概率选择邻居状态：

$$P(\Delta E, T) = e^{-\frac{f(s') - f(s)}{T}}$$

- Temperature *T* determines the probability of accepting non-improving solutions. 温度 T 决定了接受非改进解的概率。
- At a particular level of temperature, many trials are explored. Once an equilibrium state is reached, the temperature is gradually decreased according to a cooling schedule.

在特定温度水平下，会探索许多次试验。一旦达到平衡状态，根据一个冷却时间表逐渐降低温度。

SA Algorithm

Template of simulated annealing algorithm.

Input: Cooling schedule.
$s = s_0$; /* Generation of the initial solution */
$T = T_{max}$; /* Starting temperature */
Repeat
  Repeat /* At a fixed temperature */
    Generate a random neighbor s' ;
    $\Delta E = f(s') - f(s)$;
    If $\Delta E \leq 0$ **Then** $s = s'$ /* Accept the neighbor solution */
    Else Accept s' with a probability $e^{\frac{-\Delta E}{T}}$;
  Until Equilibrium condition
  /* e.g. a given number of iterations executed at each temperature T */
  $T = g(T)$; /* Temperature update */
Until Stopping criteria satisfied /* e.g. $T < T_{min}$ */
Output: Best solution found.

Move Acceptance

- The system can escape from local optima due to the probabilistic acceptance of a non-improving neighbor. 由于对非改进邻居的概率性接受，系统可以从局部最优解中逃脱。
- At high temperature, the probability of accepting worse moves is high (Why?).
- If $T = +\infty$, all moves are accepted, which corresponds to a random walk in the feasible region. 如果 T = +∞，则所有移动都被接受，这对应于在可行区域内进行随机游走。
- If $T = 0$, no worse moves are accepted and the search is equivalent to local search.

如果 T = 0，则不接受更差的移动，搜索等同于局部搜索。

Equilibrium State

- To reach an equilibrium state at each temperature, a number ($N_{nonimprov}$) of non-improving iterations must be performed. 要在每个温度达到平衡状态，必须执行一定数量（Nnonimprov）的非改进迭代。
- This number must be set according to the size of the problem instance and particularly proportional to the neighborhood size $|N(s)|$. 这个数量必须根据问题实例的大小以及特别与邻域大小 |N(s)| 成比例地设置。
- This number may be set by the following two ways:
  - **Static**: this number is determined before the search starts. 在搜索开始之前确定这个数量
  - **Adaptive**: This number will be adjusted during the search process. 这个数量将在搜索过程中进行调整

Cooling

- The temperature is decreased gradually such that

  $T_i > 0, \forall\, i$ and $\lim_{i \to +\infty} T_i = 0$.

- If the temperature is decreased slowly, better solutions are obtained but with more computation time. 如果温度降低得很慢，会获得更好的解，但需要更多的计算时间。
- The temperature T can be updated in different ways:
  - **Linear**: $T = T - \beta$, where β is a specific constant value. Hence, we have $T_i = T_0 - i \times \beta$.
  - **Geometric**: $T = \alpha T$, where $\alpha \in [0, 1]$. It is the most popular cooling function. Experience has shown that α should be between 0.5 and 0.99. 这是最流行的冷却函数。经验表明，α 应该在 0.5 到 0.99 之间。
  - **Logarithmic**: $T = T_0 / \log(i)$. This schedule is too slow to be applied in practice.

    太慢了，不适合在实践中应用
  - **Adaptive**: In an adaptive cooling schedule, the decreasing rate is dynamic and depends on some information obtained during the search.

Stopping Condition

- Reaching a final temperature T_F which is the most popular stopping criteria. This temperature must be low (e.g., $T_{\text{min}} = 0.01$).

  达到最终温度 TF 是最流行的停止准则。这个温度必须很低（例如，Tmin = 0.01）。

- Achieving a predetermined number of iterations without improving the best found solution.

Tabu Search

- TS was invented by Fred Glover (http://leeds-faculty.colorado.edu/glover/).
- TS is one of the most widespread Single-metaheuristics.
- The use of memory, which stores information related to the search process, represents the particular feature of TS. 使用存储器，存储与搜索过程相关的信息，代表了TS的特殊特征。
- TS behaves like a steepest LS algorithm, but it accepts non-improving solutions to escape from local optima when all neighbors are non-improving solutions. TS的行为类似于最陡的局部搜索算法，但当所有邻居都是非改进解时，它接受非改进解以摆脱局部最优解
- TS works in a deterministic manner.

Tabu Search 禁忌搜索

- The best solution in the neighborhood is selected as the new incumbent （现任的）solution; this may generate cycles. 邻居选择最优的
- TS discards some neighbors that have been visited previously to avoid cycles.
- TS manages a memory of the solutions or moves recently applied, which is called the tabu list, which constitutes the short-term memory. 存储最近访问过的解或操作。通过这个列表，禁忌搜索可以避免搜索过程中的循环，并指导搜索朝着更有希望的方向前进
- At each iteration of TS, the short-term memory is updated.

TS丢弃了先前访问过的一些邻居，以避免循环。
TS管理着最近应用的解或移动的内存，称为禁忌列表，它构成了短期记忆。
在TS的每次迭代中，短期记忆都会更新。

TS Algorithm

Template of tabu search algorithm.
$s = s_0$; /* Initial solution */
Initialize the tabu list, medium-term and long-term memories ;
Repeat
Find best admissible neighbor s' ; /* non tabu or aspiration criterion holds */
$s = s'$;
Update tabu list, aspiration conditions, medium and long term memories ;
If intensification_criterion holds **Then** intensification ;
If diversification_criterion holds **Then** diversification ;
Until Stopping criteria satisfied
Output: Best solution found.

缓解陷入局部最优解的困境，通过迫使搜索进入以前未探索的搜索空间的区域，

- Intensification (medium-term memory) : explore more thoroughly the portions of the search space that seem promising to make sure that the best solutions in these areas are indeed found.加强对搜索空间中有希望的区域的探索，以确保尽可能找到最优解。
- Diversification (long-term memory) : alleviate trapping in local optima by forcing the search into previously unexplored areas of the search space.

TS Components

- Tabu list（禁忌表）: The goal of using the short-term memory is to avoid cycles. Storing the list of all visited solutions is not practical for efficiency issues. 使用短期记忆来避免循环。由于效率问题，存储所有已访问的解的列表并不实际
- Aspiration criterion（特赦原则）: selecting a tabu move if it generates a solution that is better than the best found solution so far.
- Intensification (medium-term memory): The medium-term memory stores the elite (e.g., best) solutions found during the search.
- Diversification (long-term memory): The long-term memory stores information on the visited solutions along the search. It explores the unvisited areas of the solution space.

Tabu List

- Store the recent history of the search.
- Recording all visited solutions during the search
  - High complexity of data storage and computational time.
  - Data structure: hashing table
- Recording the last *k* visited solutions.
- The most popular way to represent the tabu list is to record the move attributes.
  - Data structure: arrays

表示禁忌列表的最流行方式是记录移动的属性。

Example—Tabu list based on move attributes

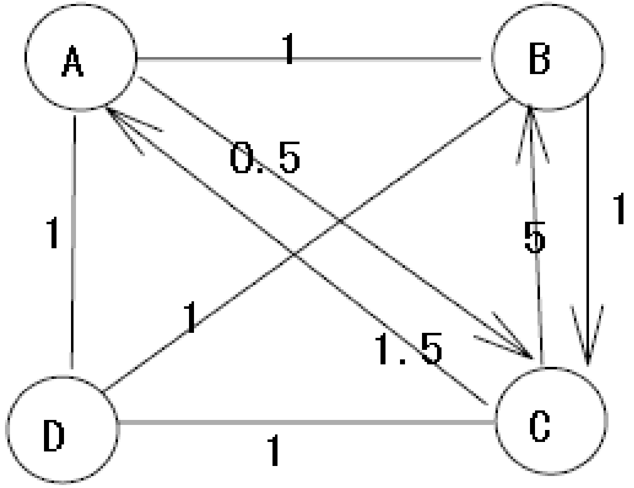
- Let us consider a permutation optimization problem, where the neighborhood is based on exchanging two elements of the permutation.
- Given a permutation π, a move is represented by two indices (i, j). This move generates the neighbor solution π' such that

$$\pi'(k) = \begin{cases} \pi(k) & \text{for } k \neq i \text{ and } k \neq j \\ \pi(j) & \text{for } k = i \\ \pi(i) & \text{for } k = j \end{cases}$$

- The inverse move (j, i) may be stored in the tabu list and is forbidden for a certain number of iterations, called tabu tenure.
- A stronger tabu representation may be related to the indices i and j. This will disallow any move involving the indices i and j.

Example -- TSP

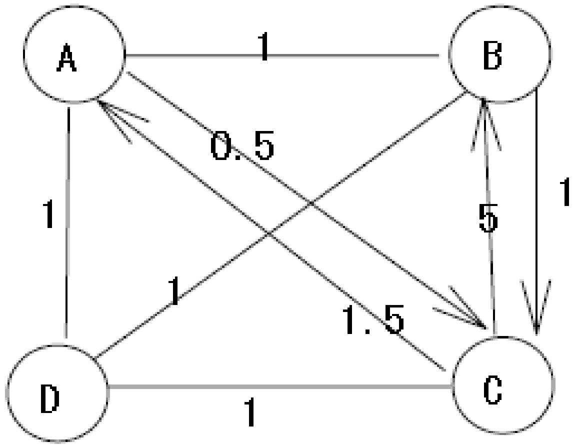
$$D = (d_{ij}) = \begin{bmatrix} 0 & 1 & 0.5 & 1 \\ 1 & 0 & 1 & 1 \\ 1.5 & 5 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$



- Initial solution: x_0=(ABCD)， $f(x_0)$=4
- City A is the starting and ending vertex
- Neighborhood operator: 2-swap (swap a pair of cities)

Example -- TSP

- Step 1:



Solution:

A	B	C	D

$f(x^0)=4$

Tabu list:

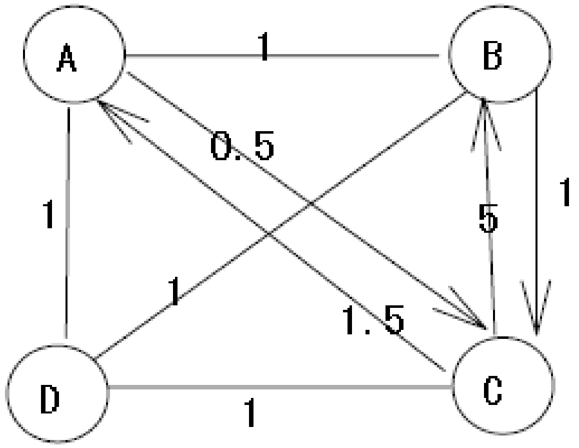
	B	C	D
A			
B			
C			

Candidate solution:

Swap	Fitness
CD	4.5
BC	7.5
BD	8

Example -- TSP

- Step 2:



Solution:

A	B	D	C

$f(x^1)=4.5$

Tabu list:

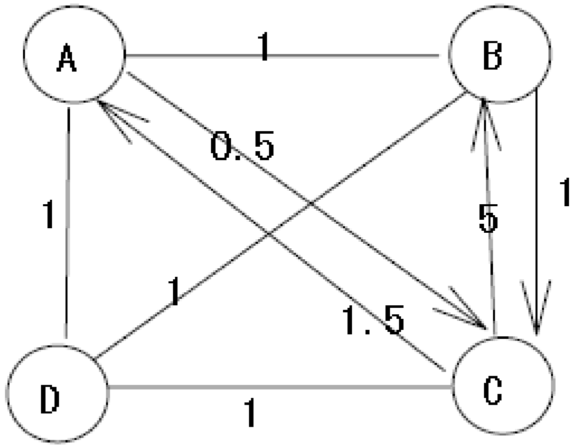
	B	C	D
A			
B			
C			3

Candidate solution:

Swap	Fitness
CD	4.5 T
BC	3.5
BD	4.5

Example -- TSP

- Step 3:



Solution:

A	C	D	B

$f(x^2)$=3.5

Tabu list:

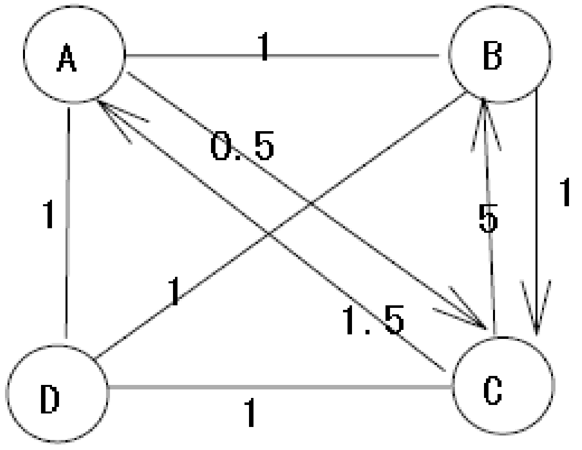
	B	C	D
A			
B		3	
C			2

Candidate solution:

Swap	Fitness
CD	8 T
BC	4.5 T
BD	7.5

Example -- TSP

- Step 4:



Note the length of the Tabu list

Solution:

A	C	B	D

$f(x^3)=7.5$

Tabu list:

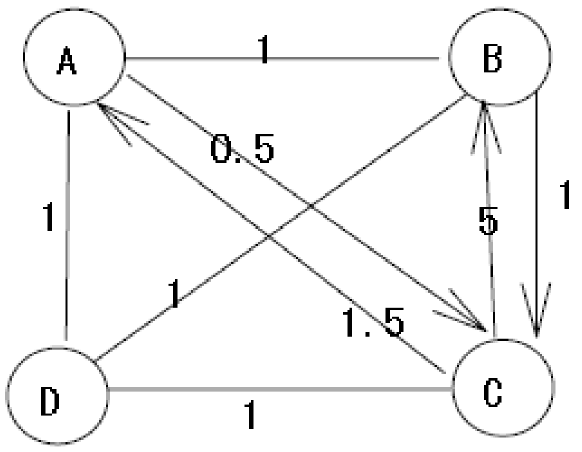
	B	C	D
A			
B		2	3
C			1

Candidate solution:

Swap	Fitness
CD	4.5 T
BC	4.5 T
BD	3.5 T

Example -- TSP

- Step 4:



Solution:

A	C	B	D

$f(x^3)=7.5$

Tabu list:

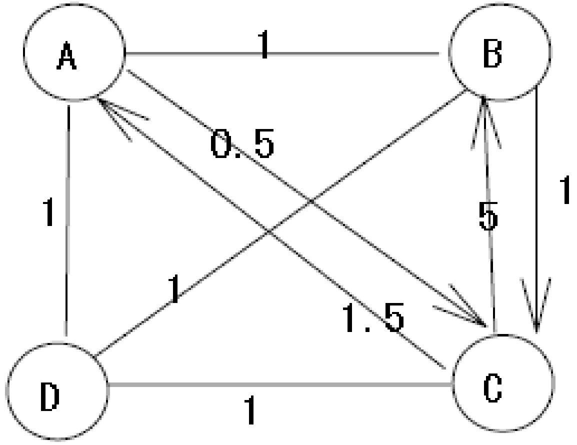
	B	C	D
A			
B		1	2
C			0

Candidate solution:

Swap	Fitness
CD	4.5 ☻
BC	4.5 T
BD	3.5 T

Example -- TSP

- Step 5:



Solution:

A	D	B	C

$f(x^4)=4.5$

Tabu list:

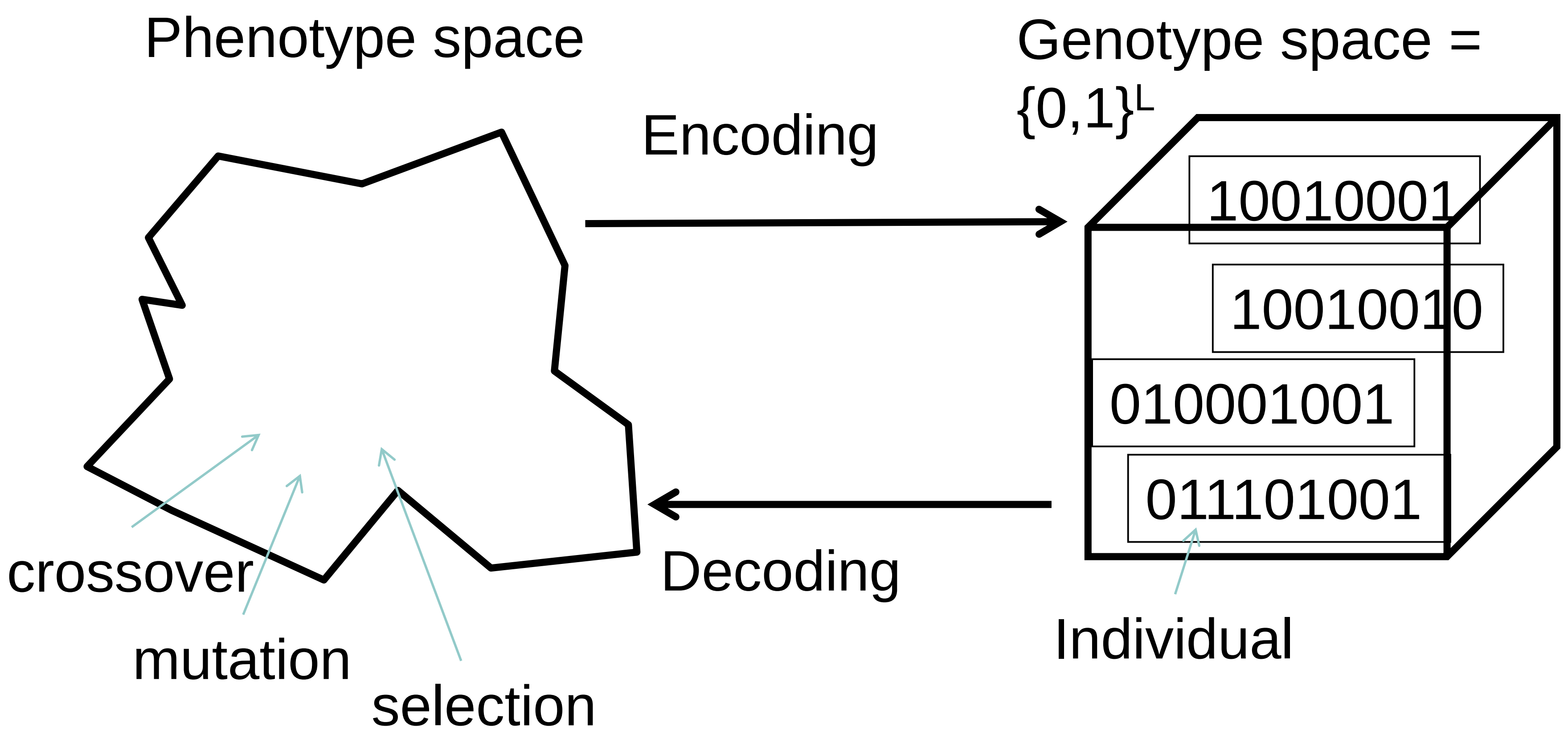
	B	C	D
A			
B		0	1
C			2

Candidate solution:

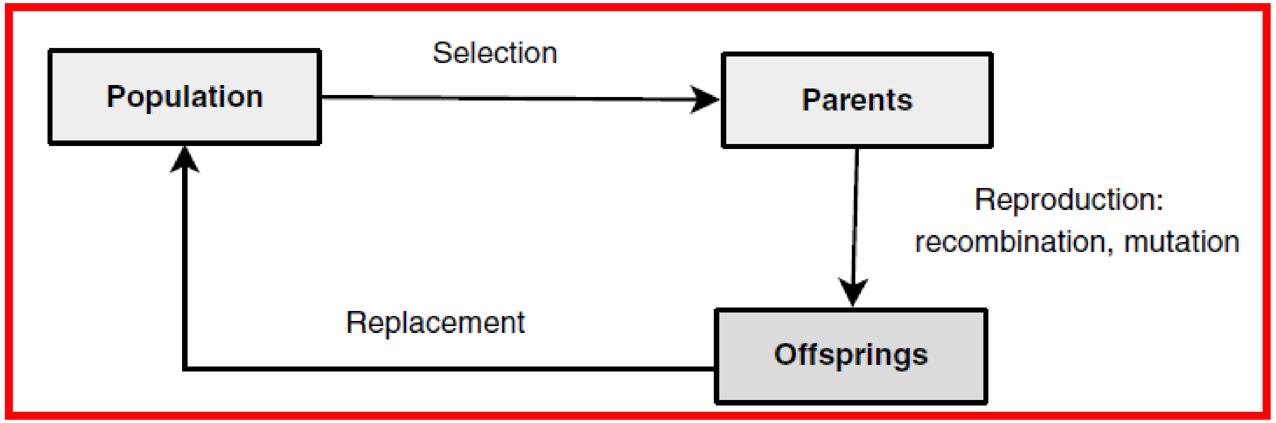
Swap	Fitness
CD	7.5 T
BC	8 ☺
BD	4.5 T

Genetic Algorithm (GA)

- A genetic algorithm (GA) is a search heuristic that mimics the process of natural selection. GA通过模拟自然选择、交叉和变异等过程来搜索最优解

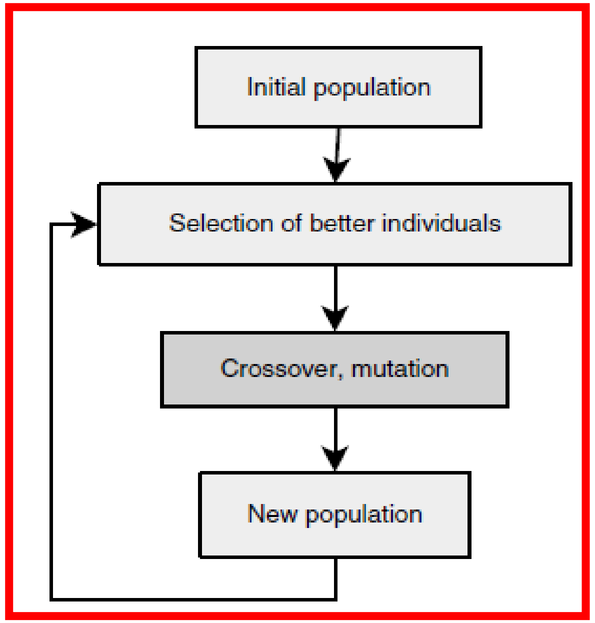


Genetic Algorithm (GA)



Template of an evolutionary algorithm.

Generate($P(0)$) ; /* Initial population */
$t = 0$;
While not Termination_Criterion($P(t)$) **Do**
  Evaluate($P(t)$) ;
  $P'(t)$ = Selection($P(t)$) ;
  $P'(t)$ = Reproduction($P'(t)$); Evaluate($P'(t)$) ;
  $P(t+1)$ = Replace($P(t)$, $P'(t)$) ;
  $t = t + 1$;
End While
Output Best individual or best population found.



Main Components in Genetic Algorithm

- **Representation**: the encoded solution is referred as chromosome while the decision variables within a solution are genes.
- **Population Initialization**: generate a set of initial solutions.
- **Objective Function**: This is a common search component for all heuristics. In GA, the term fitness function generally refers to the objective function.
- **Selection Strategy**: The selection strategy addresses the following question: “Which parents for the next generation are chosen with a bias toward better fitness?”

表示（Representation）：编码的解被称为染色体，而解中的决策变量被称为基因。
种群初始化（Population Initialization）：生成一组初始解。
目标函数（Objective Function）：这是所有启发式算法的常见搜索组件。在遗传算法中，术语适应度函数通常指的是目标函数。
选择策略（Selection Strategy）：选择策略解决以下问题：“如何有偏向性地选择更适应度更好的父代用于下一代？”

Main Components in Genetic Algorithm

- **Reproduction Strategy**: The reproduction strategy consists in designing suitable mutation and crossover operators to generated new individuals (offspring).
- **Replacement Strategy**: The new offsprings compete with old individuals for their place in the next generation.
- **Stopping Criteria**: This is a common search component for all metaheuristics.

繁殖策略（Reproduction Strategy）：繁殖策略包括设计合适的突变和交叉操作符来生成新的个体（后代）。
替换策略（Replacement Strategy）：新的后代与旧个体竞争下一代的位置。
停止准则（Stopping Criteria）：这是所有元启发式算法的共同搜索组件。

Crossover

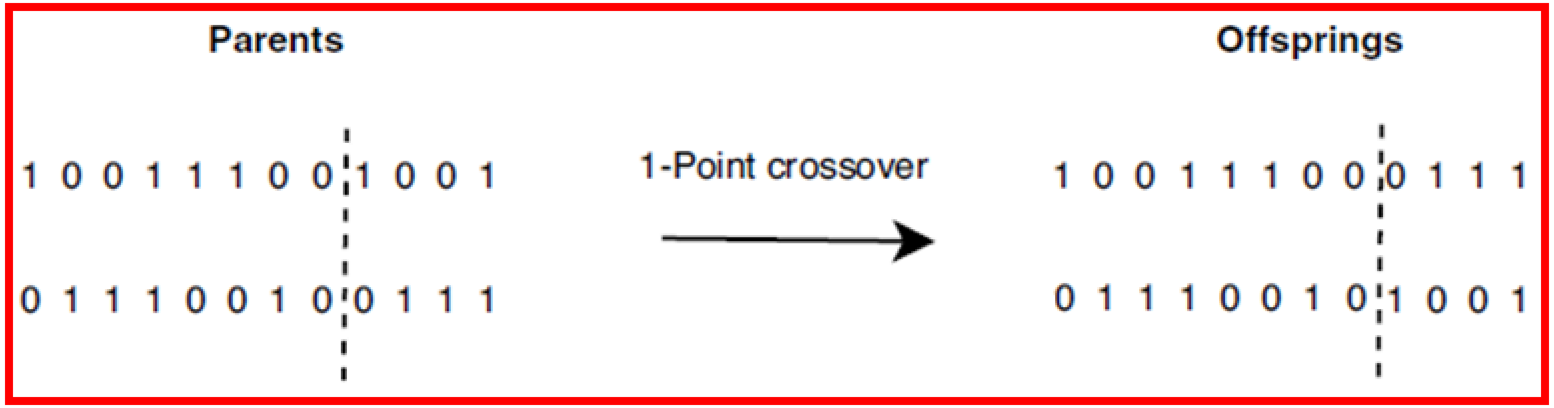
- The role of crossover operators is to inherit some characteristics of the two parents to generate offsprings. 继承两个父代的一些特征来生成后代
- The main characteristic of the crossover operator is heritability. The offsprings should inherit genetic materials from both parents. 遗传性。后代应该从两个父代继承遗传物质
- The crossover operator should produce valid solutions. 应该产生有效的解
- The crossover rate p_c ($p_c \in [0, 1]$) represents the proportion of parents on which a crossover operator will act. 交叉率 pc （pc ∈ [0, 1]）表示交叉操作符作用于父代的比例
  - The most commonly used rates are in the interval [0.45, 0.95].

最常用的交叉率在区间 [0.45, 0.95] 内

1-Point Crossover

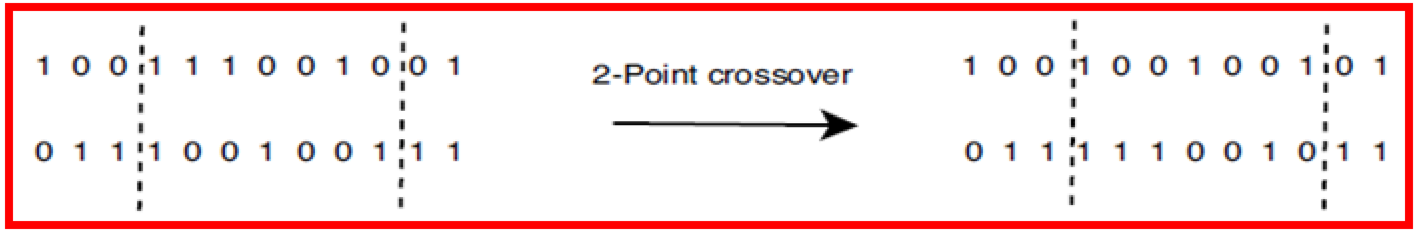
- A crossover site *k* is randomly selected 随机选择一个交叉位点 k
- Two offsprings are created by interchanging the segments of the parents.

通过交换父代的片段来创建两个后代。

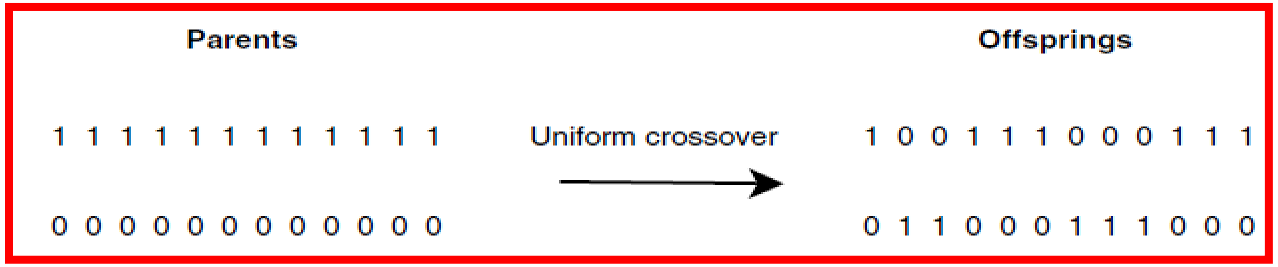


n-Point Crossover

- ***n*-point crossover**
  - *n* crossover sites are randomly selected. 随机选择 n 个交叉位点
  - The individuals *A|BCD|E* and *a|bcd|e* generate two offsprings *A|bcd|E* and *a|BCD|e*.



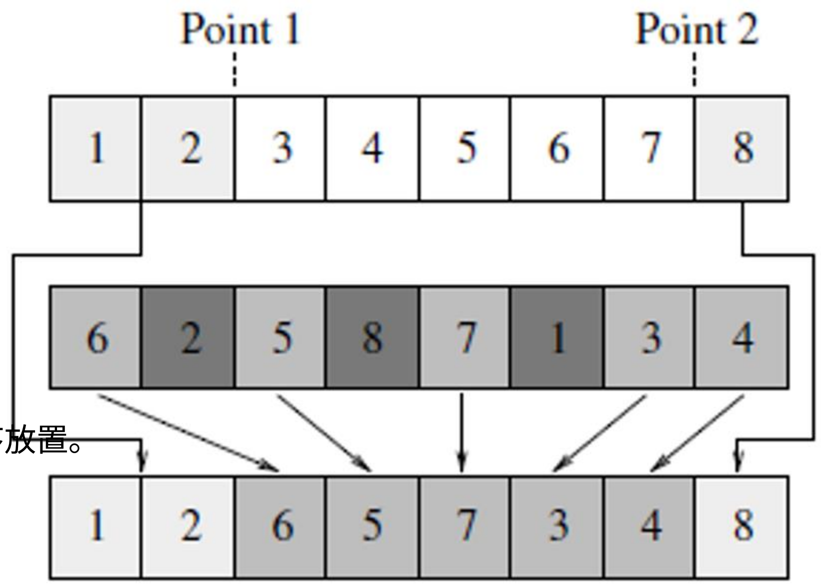
- **Uniform Crossover**
  - Each element of the offspring is selected randomly from either parent. 后代的每个元素都随机从父代中选择。
  - Each parent will contribute equally to generate the offsprings. 每个父代都将平等地为生成后代做出贡献。



Crossover for Permutations

- Applying classical crossover operators to permutations will generate solutions that are not permutations (i.e., infeasible solutions). 会生成非排列的解（即，不可行的解）
- Hence, many permutation crossover operators have been designed as follows.

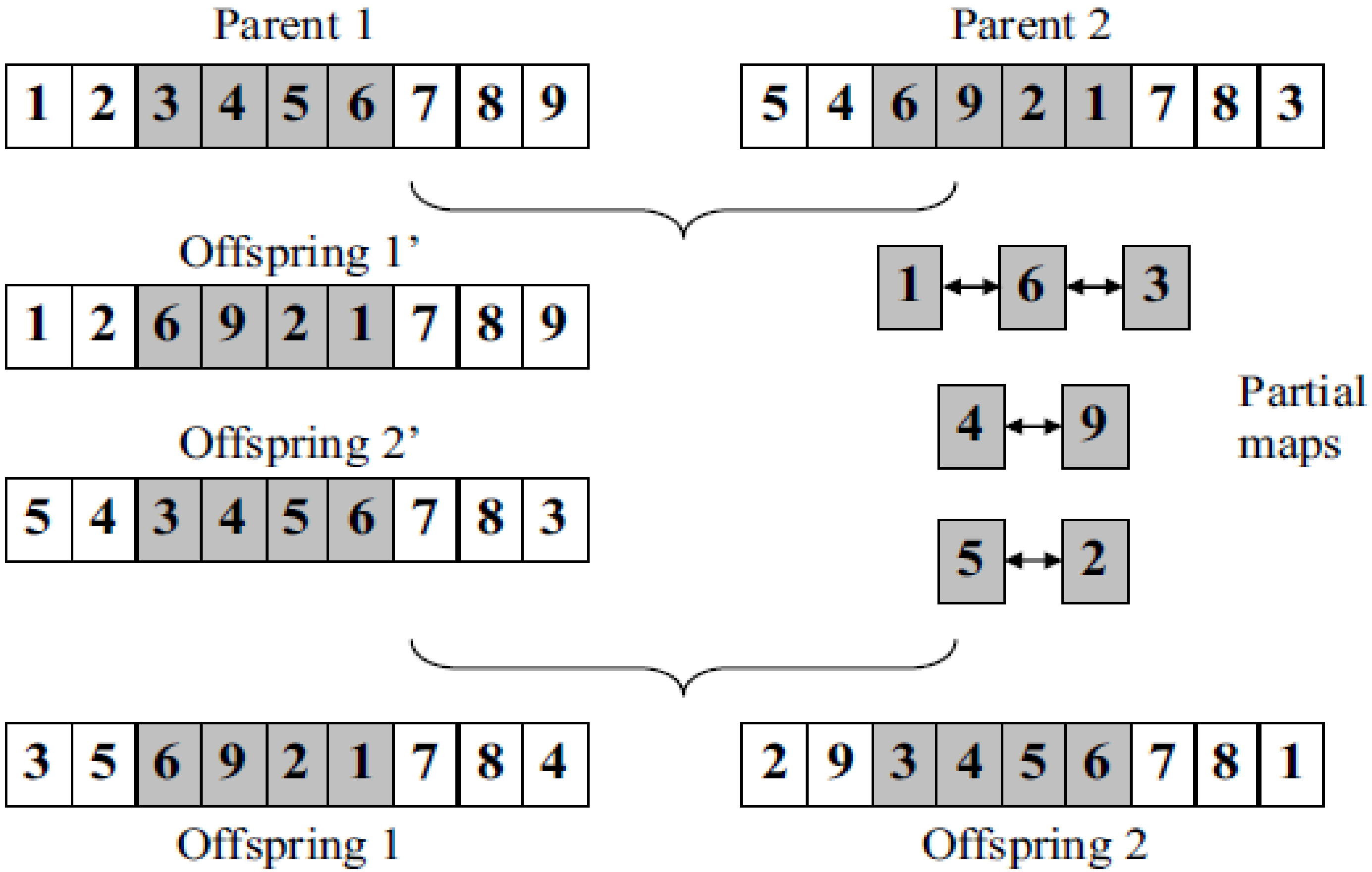
- **Order Crossover (OX)** 随机选择两个交叉点
  - Two crossover points are randomly selected.
  - The elements outside the selected two points are inherited from one parent but the rest of the elements are placed in the order of their appearance in the other parent.



选定两个点外的元素来自于其中一个父代，而其余的元素按照它们在另一个父代中出现的顺序放置。

Crossover for Permutations

- **Partially Mapped Crossover (PMX)**



An example of partially mapped crossover for permutation code

Mutation

突变代表了种群中选择个体的小改变

- Mutations represent small changes of selected individuals of the population.
- Mutation operators are **unary** (一元的) operators acting on a single individual.
  - The commonly used mutation is defined as the flip operator. 翻转
  - Mutation in order-based representation are generally based on the swapping, inversion or the insertion operators. 基于交换、反转或插入操作符
- The probability p_m defines the probability to mutate each element (gene) of the representation. 概率 pm 定义了对表示中每个元素（基因）进行突变的概率。
  - In general, small values are recommended for this probability (e.g., $p_m \in [0.001, 0.01]$).
  - Some strategies initialize mutation probability to 1/k where k is the number of decision variable. 一些策略将突变概率初始化为 1/k，其中 k 是决策变量的数量。
- Mutation introduces some diversification in the individuals by introducing some missing materials in the current individuals.

Selection Methods

- The better an individual is, the higher its chance of being parent. 个体越好，成为父代的机会就越高
- Worst individuals still have some chance to be selected.
- **Roulette Wheel Selection (轮盘赌)**
  - It will assign to each individual a selection probability that is proportional to its relative fitness.为每个个体分配一个与其相对适应度成比例的选择概率
  - Let f_i be the fitness of the individual i in the population P. Its probability to be selected is:

$$p_i = f_i \Big/ \left(\sum_{j=1}^{n} f_j \right)$$

  - A pie graph can be constructed where each individual is assigned a space on the graph that is proportional to its fitness.

可以构建一个饼图，其中每个个体被分配的空间与其适应度成比例。

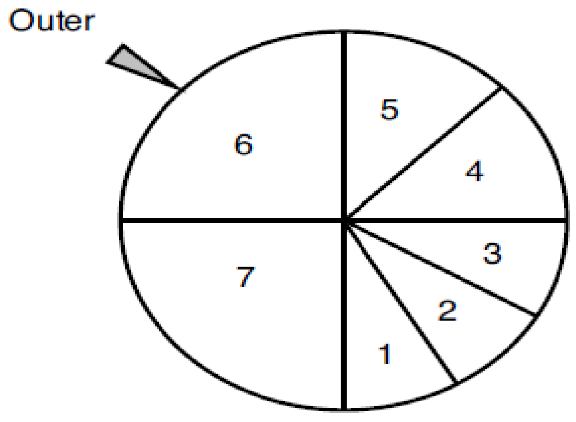
Selection Methods

通过对轮盘进行 μ 次独立旋转来选择 μ 个个体。

- An outer roulette wheel is placed around the pie.
- The selection of μ individuals is performed by μ independent spins (旋转) of the roulette wheel.
- Better individuals have more space and then more chance to be chosen.

较好的个体拥有更多的空间，因此被选择的机会更大。

Individuals:	1	2	3	4	5	6	7
Fitness:	1	1	1	1.5	1.5	3	3



Roulette selection

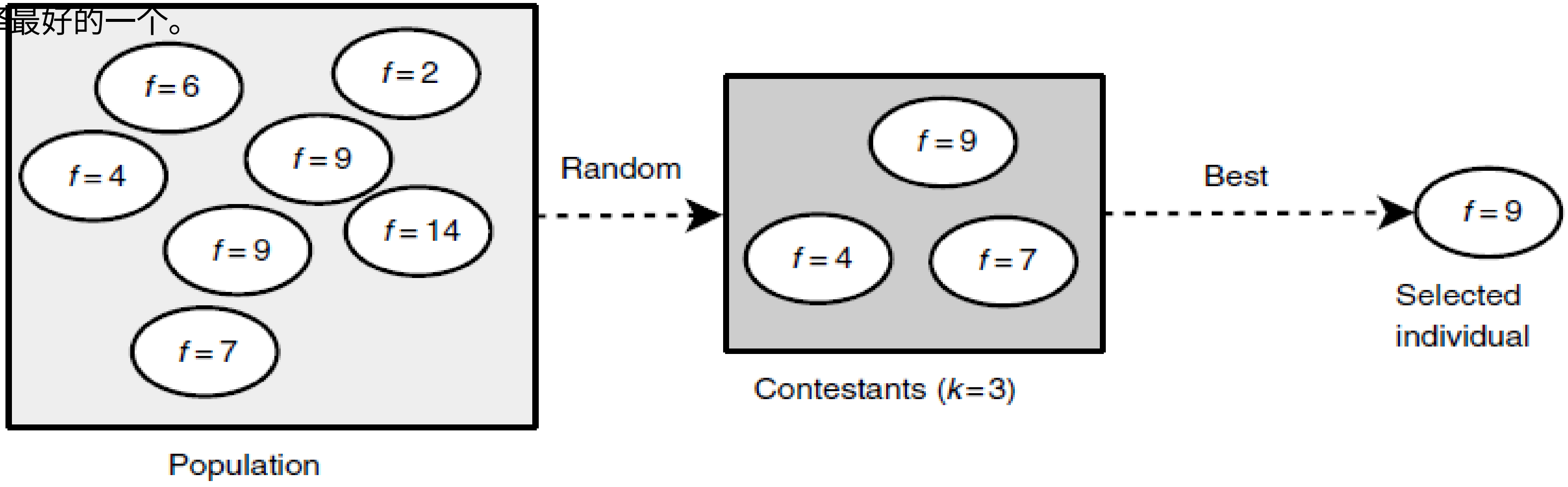
Selection Methods

- **Tournament Selection (锦标赛)**

随机选择 k 个个体；参数 k 被称为锦标赛组的大小

  - Randomly select *k* individuals; the parameter *k* is called the size of the tournament group.
  - Then, select the best one from the selected *k* individuals.

然后，从所选的 k 个个体中选择最好的一个。

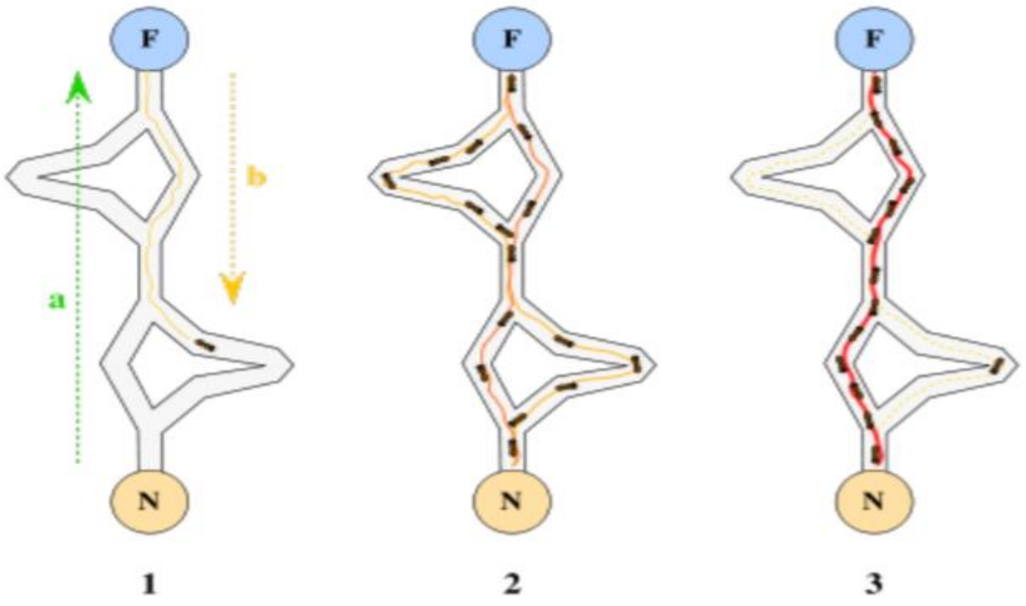


Tournament selection strategy. For instance, a tournament of size 3 is performed. Three solutions are picked randomly from the population. The best solution from the picked individuals is then selected.

Ant Colony Optimization (ACO)

蚁群优化

- History
  - Ant System was developed by Marco Dorigo (Italy) in his PhD thesis in 1992.
  - Max-Min Ant System was developed by Hoos and Stützle in 1996.
  - Ant Colony was developed by Gambardella Dorigo in 1997.



Ant Colony Optimization (ACO)

- Ants navigate from nest to food source.
- Shortest path is discovered via pheromone trails.
- Each ant moves at random.
- Pheromone (信息素) is deposited on path.
- More pheromone on path increases probability of path being followed.

蚂蚁从巢穴到食物源进行导航。

- 最短路径通过信息素路径被发现。
- 每只蚂蚁随机移动。
- 信息素被沿着路径沉积。
- 路径上的信息素越多，路径被跟随的概率就越高。

ACO Framework

```
procedure ACO algorithm for combinatorial optimization problems
    Initialization
    while (termination condition not met) do
        ConstructAntSolutions
        ApplyLocalSearch        % optional
        UpdatePheromones
    end
end ACO algorithm for combinatorial optimization problems
```

ACO – Construct Ant Solutions

- An ant will move from node i to node j with probability

$$p_{i,j} = \frac{(\tau_{i,j}^{\alpha})(\eta_{i,j}^{\beta})}{\sum(\tau_{i,j}^{\alpha})(\eta_{i,j}^{\beta})}$$

- where $\tau_{i,j}$ is the amount of pheromone on edge (i, j) 信息素量
- α is a parameter to control the influence of $\tau_{i,j}$ α 是控制 τ i,j 影响的参数。
- $\eta_{i,j}$ is the desirability of edge (i, j) (typically $1/d_{i,j}$) η i,j 是边 (i, j) 的吸引力（通常为 1/di,j，其中 di,j 是节点 i 到节点 j 的距离）
- β is a parameter to control the influence of $\eta_{i,j}$ β 是控制 η i,j 影响的参数

ACO - Pheromone Update

- The amount of pheromone is updated according to the equation

$$\tau_{i,j} = (1 - \rho)\tau_{i,j} + \Delta\tau_{i,j}$$

- where $\tau_{i,j}$ is the amount of pheromone on a given edge (*i*, *j*)
- ρ is the rate of pheromone evaporation ρ 是信息素蒸发率。
- $\Delta\tau_{i,j}$ is the amount of pheromone deposited, typically given by

$$\Delta\tau^k_{i,j} = \begin{cases} 1/L_k & \text{if ant } k \text{ travels on edge } i,j \\ 0 & \text{otherwise} \end{cases}$$

- where L_k is the cost of the *k*-th ant's tour (typically length).

其中，Lk 是第 k 只蚂蚁的路径成本（通常是长度）。

ACO - Ant Colony System

- First major improvement of Ant System
- Differences with Ant System
  - Pseudo random proportional rule
  - Best only offline Pheromone Update

ACO - Ant Colony System

- Ants in ACS use the pseudo random proportional rule (ε-greedy).
- Probability for an ant to move from node i to node j depends on a random variable q uniformly distributed over [0, 1], and a parameter q_0.
- If $q \leq q_0$, then, among the feasible components, the component that maximizes the product $(\tau_{i,j}^{\alpha})(\eta_{i,j}^{\beta})$ is chosen.
- Otherwise the same equation as in Ant System is used.

$$p_{i,j} = \frac{(\tau_{i,j}^{\alpha})(\eta_{i,j}^{\beta})}{\sum(\tau_{i,j}^{\alpha})(\eta_{i,j}^{\beta})}$$

- This rule favors exploitation of pheromone information.

ACO - Ant Colony System

- Best only offline pheromone update after construction.
- Offline pheromone update equation

$$\tau_{ij} \leftarrow (1-\rho)\cdot\tau_{ij} + \rho\cdot\Delta\tau_{ij}^{best}$$

- where

$$\tau_{ij}^{best} = \begin{cases} 1/L_{best} & \text{if best ant } k \text{ travels on edge } i,j \\ 0 & \text{otherwise} \end{cases}$$

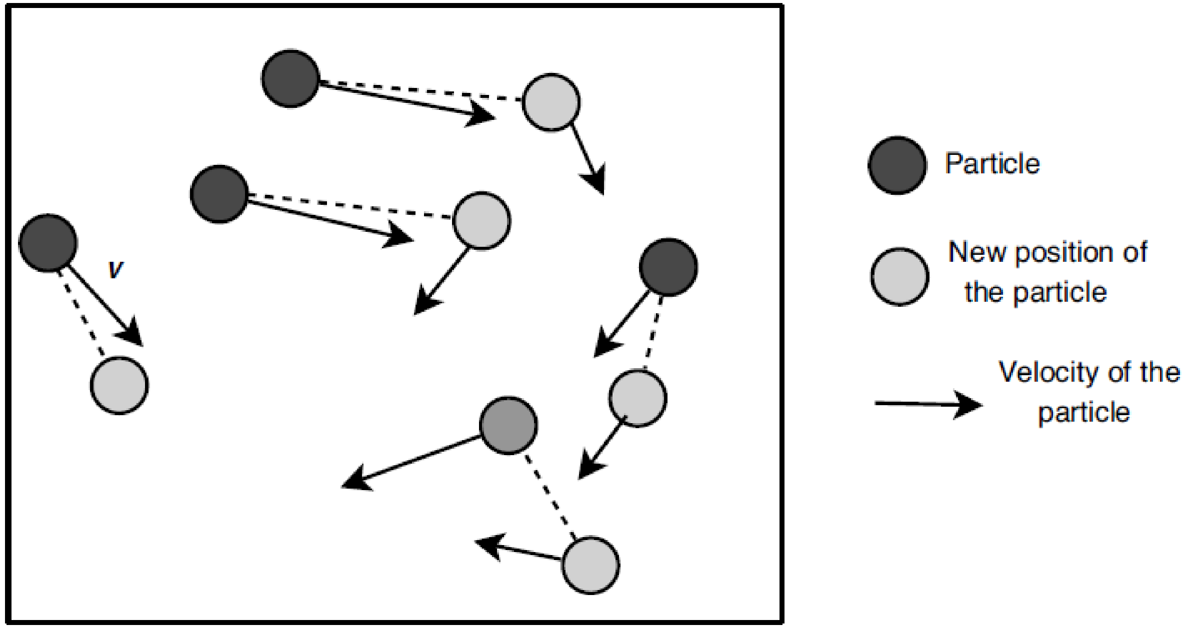
- L_{best} can be set to the length of the best tour found in the current iteration **(iter-best)** or the best solution found since the start of the algorithm **(global-best)**.

ACO - MAX-MIN Ant System

- Min and Max values of the pheromone are explicitly limited.
- τ_{ij} is constrained between τ_{min} and τ_{max} (explicitly set by algorithm designer).
- After pheromone update, τ_{ij} is set to
  - τ_{max}, if $\tau_{ij} > \tau_{max}$
  - τ_{min}, if $\tau_{ij} < \tau_{min}$

Particle Swarm Optimization (PSO)

- PSO is a stochastic population-based metaheuristic inspired from swarm intelligence (群体智能). 随机种群型元启发式算法
- Mimics the social behavior of natural organisms (社会组织) such as bird flock and a school of fish to find a place with enough food. 它模仿自然生物（如鸟群和鱼群）的社会行为，以找到一个有足够食物的地方。



Decision space

Particle Swarm Optimization (PSO)

- A swarm consists of N particles flying around in a D-dimensional search space.
- Each particle i is a candidate solution to the problem, and is represented by the vector x_i in the decision space.
- A particle has its own position and velocity (速度), which indicates the flying direction.
- The success of some particles will influence the behavior of their peers.
- Each particle successively adjusts its position x_i toward the global optimum according to the following two factors: 每个粒子根据以下两个因素连续调整其位置xi，朝着全局最优解调整：
  - The best position visited by itself (pbest$_i$) denoted by $p_i = (p_{i1}, p_{i2}, \ldots, p_{iD})$. 自己访问的最佳位置
  - The best position visited by the whole swarm (gbest) (or lbest, the best position for a given subset of the swarm) denoted by $p_g=(p_{g1}, p_{g2}, \ldots, p_{gD})$.

整个群体访问的最佳位置（gbest）

一个群体由在D维搜索空间中飞行的N个粒子组成。
每个粒子i是问题的一个候选解，并由决策空间中的向量xi表示。
一个粒子有自己的位置和速度，它们表示飞行的方向。
一些粒子的成功会影响它们同伴的行为。

Particle Swarm Optimization (PSO)

- The vector (p_g - x_i) represents the difference between the current position of the particle *i* and the best position of its neighborhood. 粒子 i 的当前位置与其邻域的最佳位置之间的差异
- A particle is composed of three vectors:
  - The *x*-vector records the current position (location) of the particle in the search space.
  - The *p*-vector records the location of the best solution found so far by the particle.
  - The *v*-vector contains a direction for which particle will travel.
  - Two fitness values: the *x*-fitness records the fitness of the *x*-vector, and the *p*-fitness records the fitness of the *p*-vector.
- A particle swarm may be viewed as a *cellular automata* (细胞自动机) where individual cell updates are done in parallel.
- Each new cell value depends only on the old value of the cell and its neighborhood, and all cells are updated using the same rules.

粒子群可以被视为细胞自动机，其中单个单元格的更新是并行进行的。
每个新的单元格值仅依赖于单元格的旧值和它的邻域，并且所有单元格都使用相同的规则进行更新。

Particle Update

- At each iteration, each particle will apply the following operations:

- **Update the velocity**: the amount of change that will be applied to the particle, is defined as:

$$v_i(t) = \omega * v_i(t - 1) + \rho_1 * C_1 * (p_i - x_i(t-1)) + \rho_2 * C_2 * (p_g - x_i(t-1)),$$

where $\omega \in [0, 1]$, ρ_1 and ρ_2 are two random variables in the range [0, 1] and constants C_1 and C_2 represent the learning factors.

其中，ω∈[0，1]，ρ1和ρ2是在范围[0，1]内的两个随机变量，常数C1和C2表示学习因子。

  - The parameter C_1 is the *cognitive* (认知的) learning factor that represents the attraction that a particle has toward ***its own success***.
  - The parameter C_2 is the *social* learning factor that represents the attraction that a particle has toward the ***success of its neighbors***.
  - The velocity defines the direction and the distance the particle should go.

参数 C1 是认知学习因子，代表粒子对自身成功的吸引力。
参数 C2 是社会学习因子，代表粒子对其邻居成功的吸引力。
速度定义了粒子应该前进的方向和距离。

Particle Update

- **Update the position**: Each particle will update its coordinates in the decision space. Then, it moves to the new position.

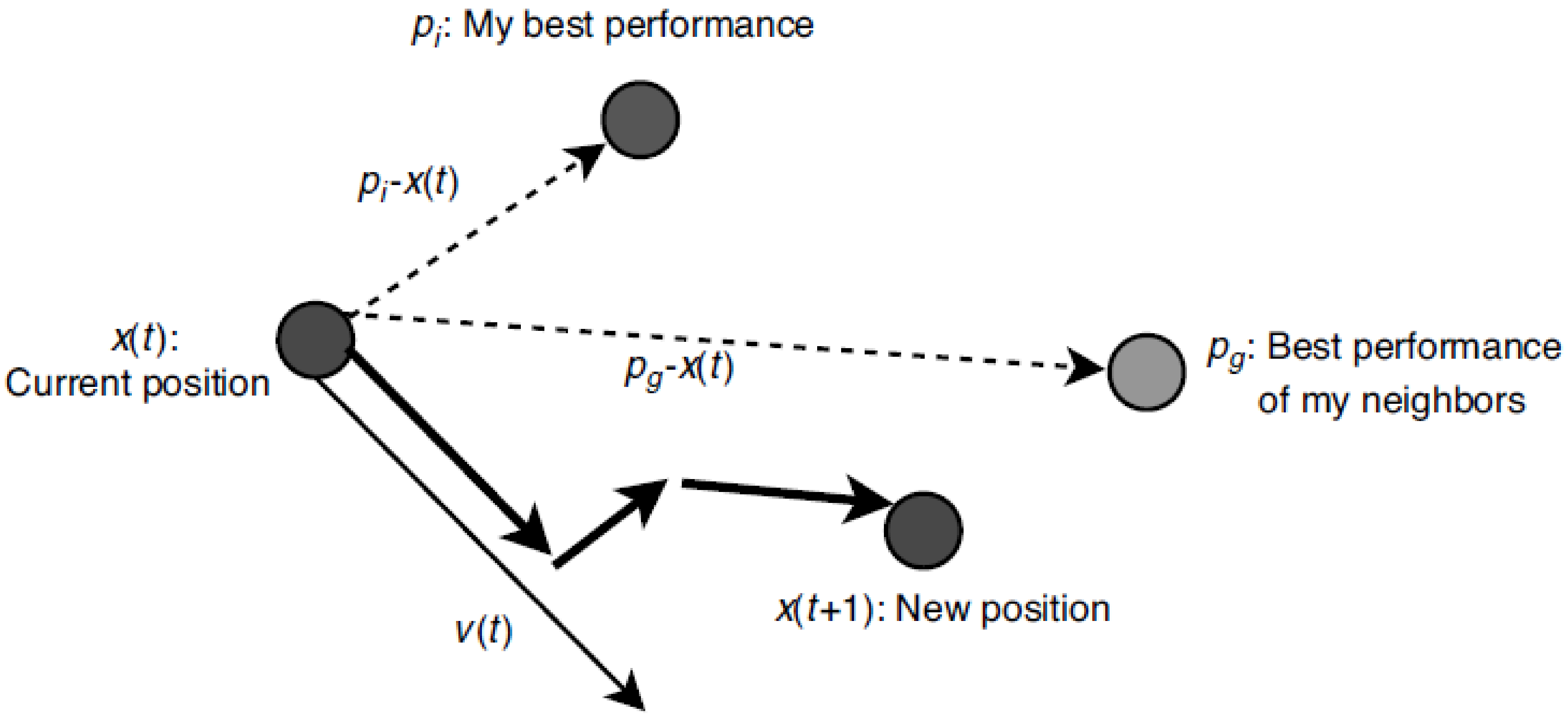
  $$x_i(t) = x_i(t - 1) + v_i(t)$$

- **Update the best found particles**
  - Each particle will update the best local solution, i.e., if $f(x_i) < pbest_i$, then $p_i = x_i$.
  - Moreover, the best global solution of the swarm is updated, i.e., $f(x_i) < gbest$, then $p_g = x_i$.

- Hence, at each iteration, each particle will change its position according to its own experience and that of neighboring particles.

  因此，在每次迭代中，每个粒子都会根据自己的经验和邻近粒子的经验来改变位置。

Particle Update



Movement of a particle and the velocity update.

Template of PSO

Random initialization of the whole swarm;

Repeat

Evaluate $f(x_i)$;

For all particles i

\# Update velocities:

$v_i(t) = \omega * v_i(t\text{-}1) + \rho_1 * C_1 * (p_i - x_i(t\text{-}1)) + \rho_2 * C_2 * (p_g - x_i(t\text{-}1))$;

Move to the new position: $x_i(t) = x_i(t\text{-}1) + v_i(t)$;

If $f(x_i) < f(pbest_i)$ **Then** $pbest_i = x_i$;

If $f(x_i) < f(gbest)$ **Then** $gbest = x_i$;

Update(x_i, v_i);

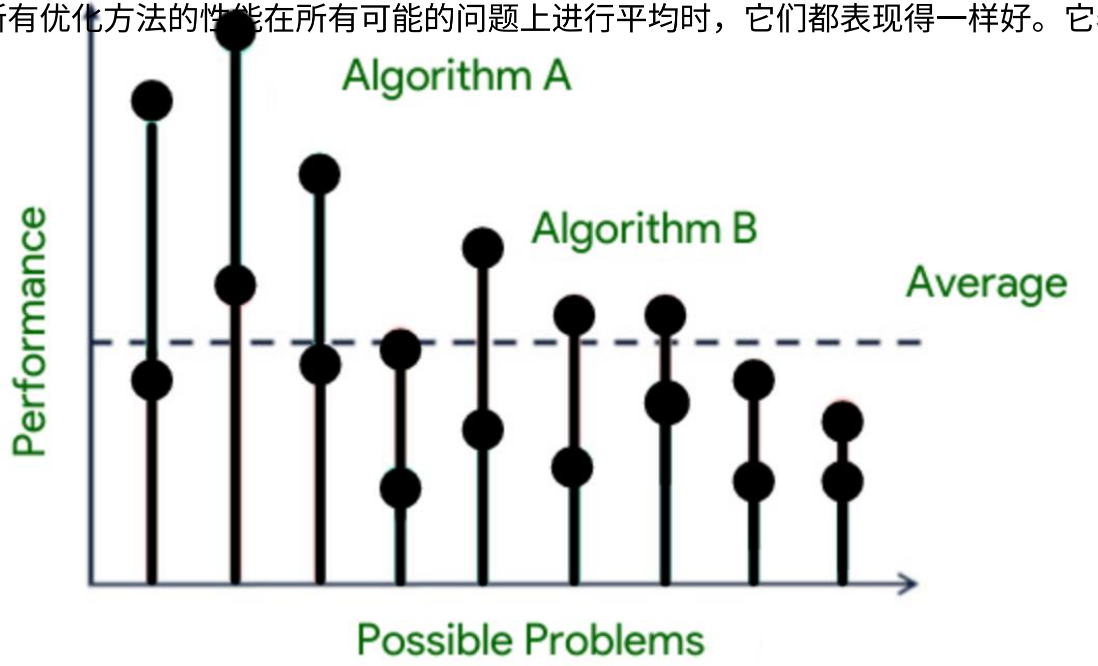
EndFor

Until Stopping criteria

Summary of Metaheuristics

- No Free Lunch Theorem: The theorem asserts that when the performance of all optimization methods is averaged across all conceivable problems, they all perform equally well. It indicates that no one optimum optimization algorithm exists.

"没有免费午餐"定理：该定理断言，当所有优化方法的性能在所有可能的问题上进行平均时，它们都表现得一样好。它表明不存在一个最优的优化算法。



Summary of Metaheuristics

- Escaping Local Optima
  - Restart: re-initialize search whenever a local optimum is encountered. (Often rather ineffective due to cost of initialization.)
  - Allow non-improving steps: in local optima, allow selection of candidate solutions with equal or worse evaluation function value, e.g., using minimally worsening steps. (Can lead to long walks in plateaus, i.e., regions of search positions with identical evaluation function.)
  - Diversify the neighborhood: multiple, variable-size, rich (while still preserving incremental algorithmic insights)

逃离局部最优解

重新开始：在遇到局部最优解时重新初始化搜索。（通常由于初始化的成本而效果不佳。）
允许非改进步骤：在局部最优解中，允许选择候选解，其评估函数值相等或更差，例如，使用最小程度的恶化步骤。（可能导致在高原上长时间行走，即具有相同评估函数的搜索位置的区域。）
扩展邻域：多个、可变大小的、丰富的（同时保留增量算法洞见）

Summary of Metaheuristics

- Enhancing the Quality of the End Result
  - How to balance the Intensification (or exploitation) and Diversification (or exploration)
    - Too much intensification -> local search
    - Too much diversification -> random search
  - Initialization Method
  - Hybrid Method
  - Operator Enhancement
- Reducing the Running Time
  - Parallel Computing
  - Employing Advanced Data Structures
  - Redesigning the procedure of Metaheuristics

提高最终结果的质量

- 如何平衡强化（或开发）和多样化（或探索）
- 过多的强化 -> 局部搜索
- 过多的多样化 -> 随机搜索
- 初始化方法
- 混合方法
- 算子增强

减少运行时间

- 并行计算
- 使用先进的数据结构
- 重新设计元启发式过程

Homework

- Online Judge: Order Crossover, PMX
- http://soj.acmm.club/contest_detail.php?cid=2967

Thank you!