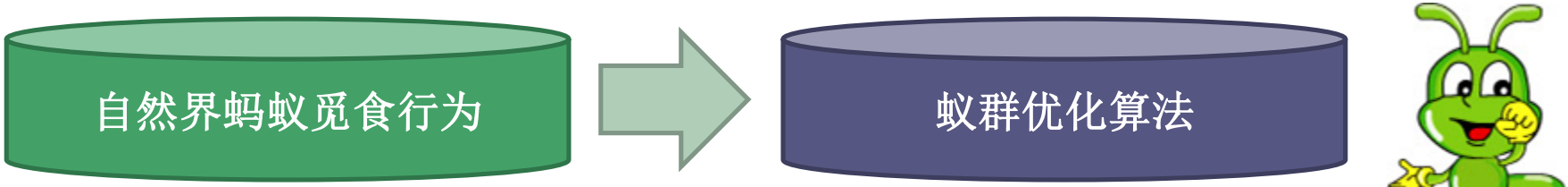
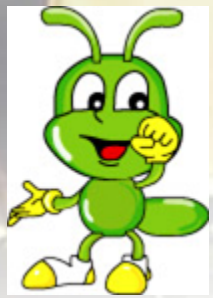
第5章 蚁群优化算法

Contents

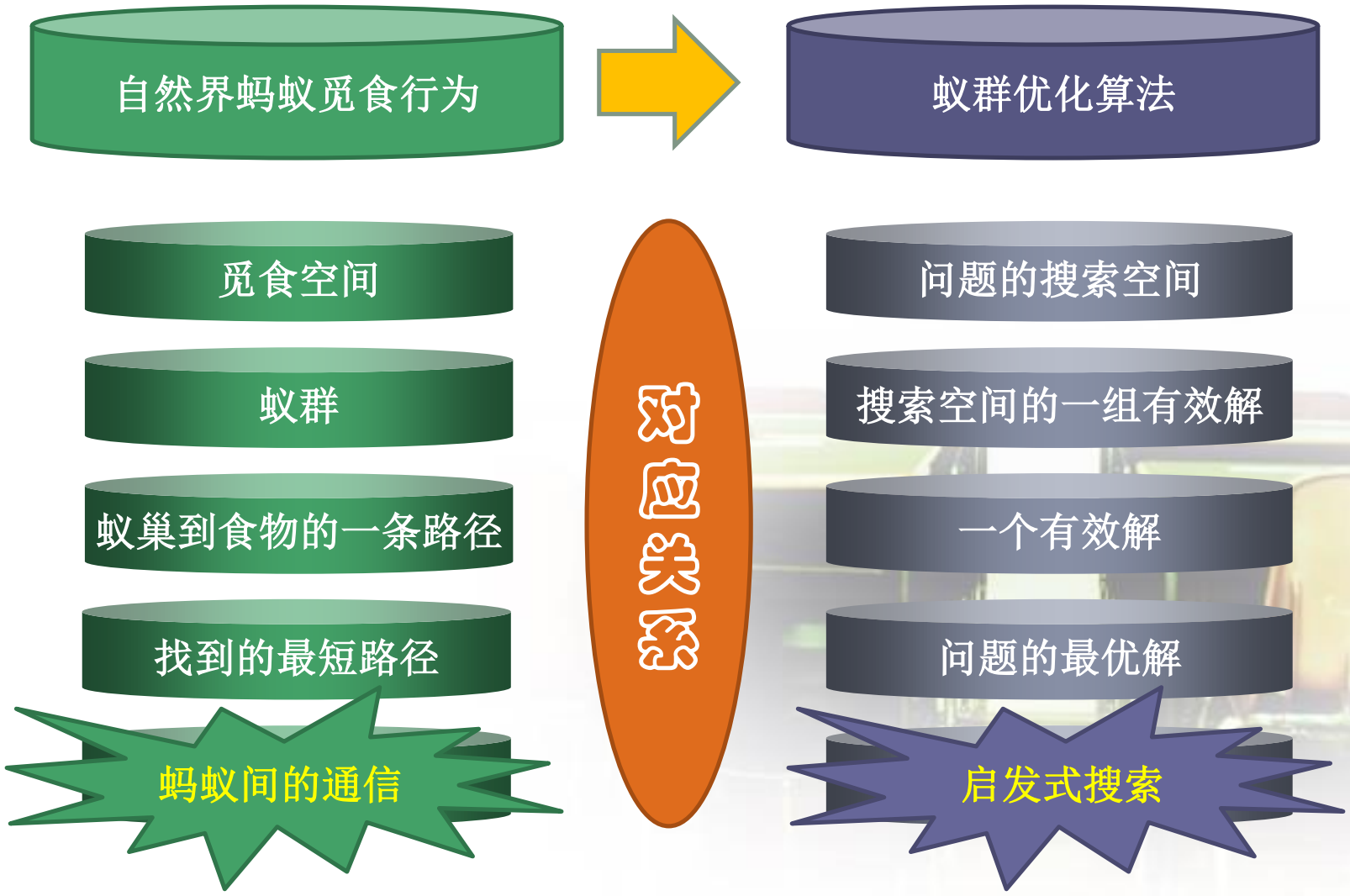
5.1 基本原理



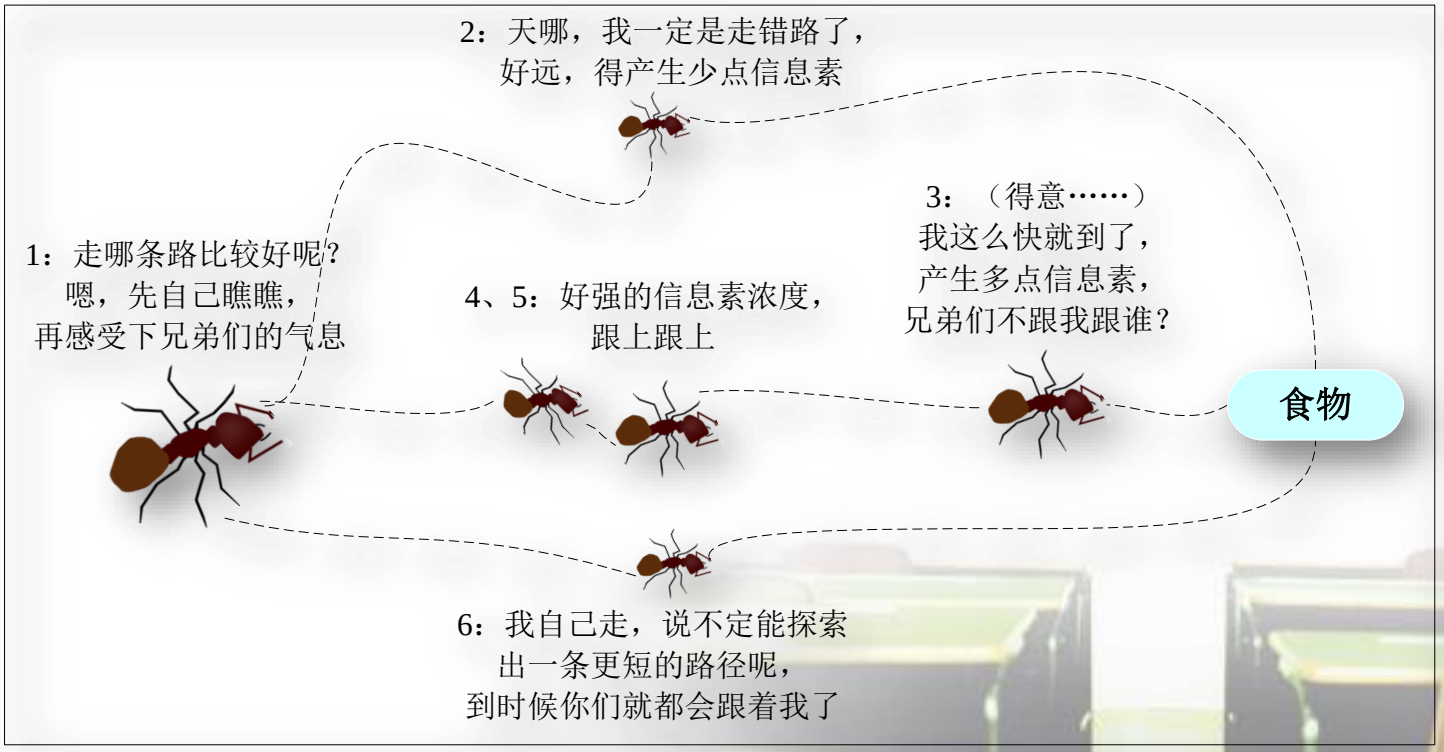
- 自然界蚂蚁群体在寻找食物的过程中，通过一种被称为信息素（**Pheromone**）的物质实现相互的间接通信，从而能够合作发现从蚁穴到食物源的最短路径。
- 通过对这种群体智能行为的抽象建模，研究者提出了蚁群优化算法（**Ant Colony Optimization, ACO**），为最优化问题、尤其是组合优化问题的求解提供了一强有力的手段。



5.1 基本原理

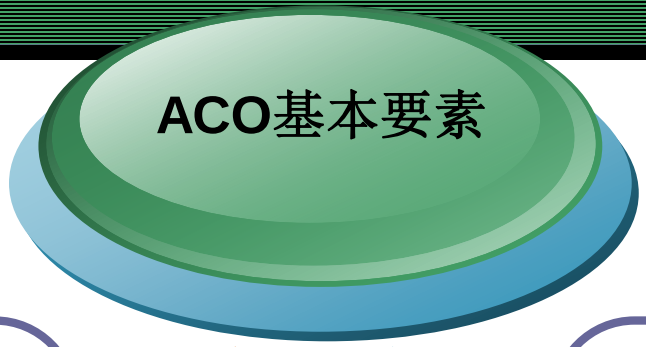

自然界蚂蚁觅食行为
蚁群优化算法
觅食空间
蚁群
蚁巢到食物的一条路径
找到的最短路径
蚂蚁间的通信
对应关系
问题的搜索空间
搜索空间的一组有效解
一个有效解
问题的最优解
启发式搜索

5.1 基本原理



- 蚂蚁在寻找食物的过程中往往是随机选择路径的，但它们能感知当前地面上的信息素浓度，并倾向于往信息素浓度高的方向行进。信息素由蚂蚁自身释放，是实现蚁群内间接通信的物质。由于较短路径上蚂蚁的往返时间比较短，单位时间内经过该路径的蚂蚁多，所以信息素的积累速度比较长路径快。因此，当后续蚂蚁在路口时，就能感知先前蚂蚁留下的信息，并倾向于选择一条较短的路径前行。这种正反馈机制使得越来越多的蚂蚁在巢穴与食物之间的最短路径上行进。由于其他路径上的信息素会随着时间蒸发，最终所有的蚂蚁都在最优路径上行进。

5.2 算法流程



路径构建

每只蚂蚁都随机选择一个城市作为其出发城市，并维护一个路径记忆向量，用来存放该蚂蚁依次经过的城市。蚂蚁在构建路径的每一步中，按照一个随机比例规则选择下一个要到达的城市。

信息素更新

当所有蚂蚁构建完路径后，算法将会对所有的路径进行全局信息素的更新。注意，我们所描述的是AS的ant-cycle版本，更新是在全部蚂蚁均完成了路径的构造后才进行的，信息素的浓度变化与蚂蚁在这一轮中构建的路径长度相关。

- 蚂蚁系统（**Ant System，AS**）是最基本的**ACO**算法，是以**TSP**作为应用实例提出的。

5.2 算法流程

- 路径构建——
  伪随机比例选择规则（**random proportional**）

$$p_k(i,j)=\begin{cases}\dfrac{[\tau(i,j)]^{\alpha}[\eta(i,j)]^{\beta}}{\sum_{u\in J_k(i)}[\tau(i,u)]^{\alpha}[\eta(i,u)]^{\beta}}, & \text{if } j\in J_k(i)\\ 0, & \text{otherwise}\end{cases}$$

- 对于每只蚂蚁k，路径记忆向量R^k按照访问顺序记录了所有k已经经过的城市序号。设蚂蚁k当前所在城市为i，则其选择城市j作为下一个访问对象的概率如上式。$J_k(i)$表示从城市i可以直接到达的、且又不在蚂蚁访问过的城市序列R^k中的城市集合。$\eta(i,j)$是一个启发式信息，通常由$\eta(i,j)=1/d_{ij}$直接计算。$\tau(i,j)$表示边(i,j)上的信息素量。

5.2 算法流程

Company Logo

❖路径构建——伪随机比例选择规则（random proportional）

$$p_k(i,j)=\begin{cases}\dfrac{[\tau(i,j)]^{\alpha}[\eta(i,j)]^{\beta}}{\sum_{u\in J_k(i)}[\tau(i,u)]^{\alpha}[\eta(i,u)]^{\beta}}, & \text{if } j\in J_k(i)\\ 0, & \text{otherwise}\end{cases}$$

- 长度越短、信息素浓度越大的路径被蚂蚁选择的概率越大。α和β是两个预先设置的参数，用来控制启发式信息与信息素浓度作用的权重关系。当α=0时，算法演变成传统的随机贪心算法，最邻近城市被选中的概率最大。当β=0时，蚂蚁完全只根据信息素浓度确定路径，算法将快速收敛，这样构建出的最优路径往往与实际目标有着较大的差异，算法的性能比较糟糕。

5.2 算法流程

❖信息素更新

- （1）在算法初始化时，问题空间中所有的边上的信息素都被初始化为τ_0。
- （2）算法迭代每一轮，问题空间中的所有路径上的信息素都会发生蒸发，我们为所有边上的信息素乘上一个小于1的常数。信息素蒸发是自然界本身固有的特征，在算法中能够帮助避免信息素的无限积累，使得算法可以快速丢弃之前构建过的较差的路径。
- （3）蚂蚁根据自己构建的路径长度在它们本轮经过的边上释放信息素。蚂蚁构建的路径越短、释放的信息素就越多。一条边被蚂蚁爬过的次数越多、它所获得的信息素也越多。
- （4）迭代（2），直至算法终止。

5.2 算法流程

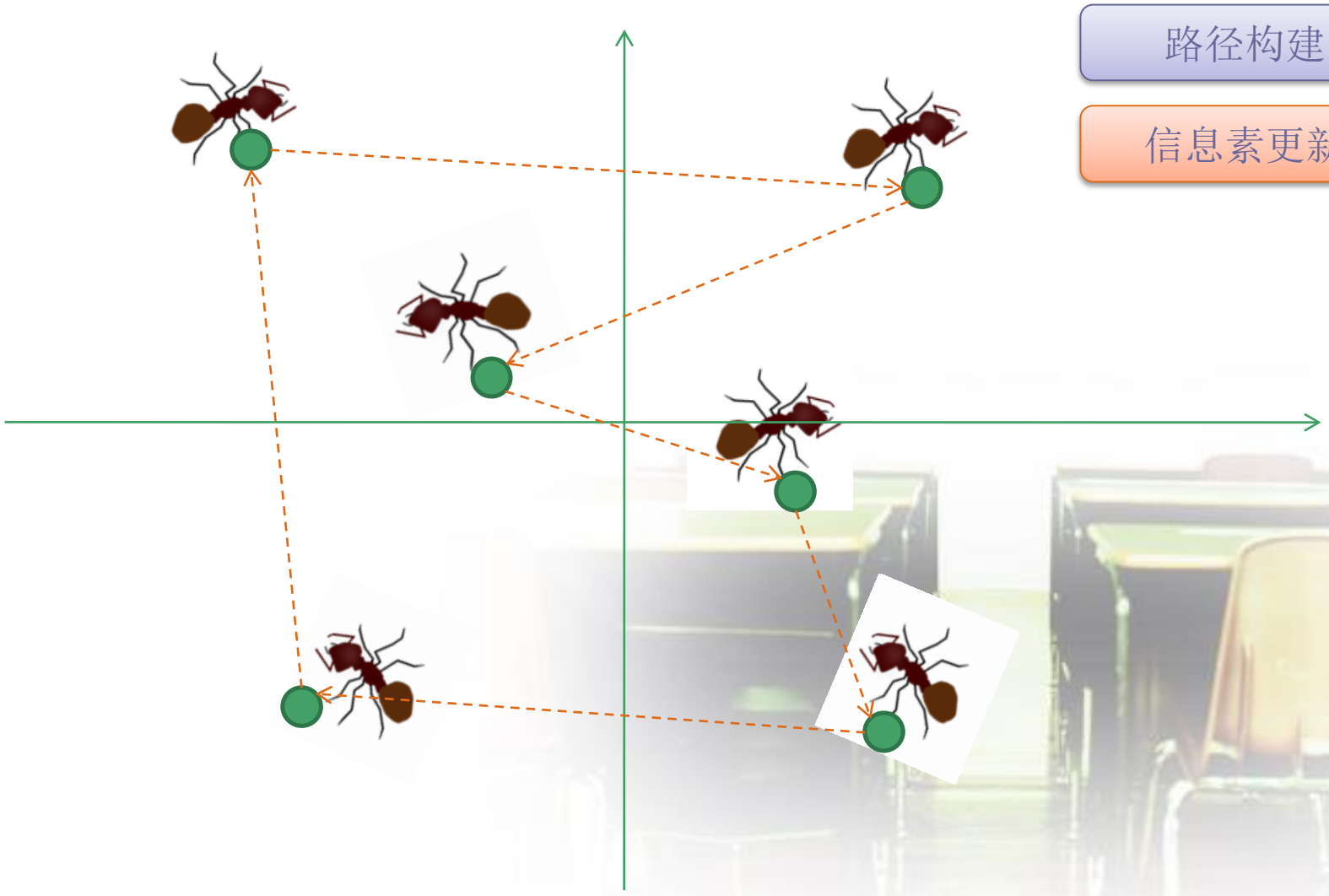
❖信息素更新

$$\tau(i,j) = (1-\rho)\cdot\tau(i,j) + \sum_{k=1}^{m}\Delta\tau_k(i,j),$$

$$\Delta\tau_k(i,j) = \begin{cases} (C_k)^{-1}, & \text{if } (i,j)\in R^k \\ 0, & \text{otherwise} \end{cases}$$

- m是蚂蚁个数；ρ是信息素的蒸发率，规定$0<\rho\leq 1$。$\Delta\tau_k(i,j)$ 是第k只蚂蚁在它经过的边上释放的信息素量，它等于蚂蚁k本轮构建路径长度的倒数。C_k表示路径长度，它是R^k中所有边的长度和。

5.2 算法流程



5.2 算法流程

- 例5.1 给出用蚁群算法求解一个四城市的TSP问题的执行步骤，四个城市A、B、C、D之间的距离矩阵如下

$$W = d_{ij} = \begin{bmatrix} \infty & 3 & 1 & 2 \\ 3 & \infty & 5 & 4 \\ 1 & 5 & \infty & 2 \\ 2 & 4 & 2 & \infty \end{bmatrix}$$

- 假设蚂蚁种群的规模m=3,参数α=1，β=2，ρ=0.5。

5.2 算法流程

❖ 解：

步骤1：初始化。首先使用贪心算法得到路径(*ACDBA*)，则$C^{nn}=f(ACDBA)=1+2+4+3=10$。求得$\tau_0=m/C^{nn}=3/10=0.3$。初始化所有边上的信息素$\tau_{ij}=\tau_0$。

❖ 解：

步骤2.1：为每只蚂蚁随机选择出发城市，假设蚂蚁1选择城市A，蚂蚁2选择城市B，蚂蚁3选择城市D。 多样性

5.2 算法流程

❖ 解：

步骤2.2：为每只蚂蚁选择下城市。我们仅以蚂蚁1为例，当前城市i=A,可访问城市集合$J_1(i)$ ={B, C, D}。计算蚂蚁1选择B,C,D作为下一访问城市的概率：

$$A \Rightarrow \begin{cases} B: \tau_{AB}^{\alpha} \times \eta_{AB}^{\beta} = 0.3^1 \times (1/3)^2 = 0.033 \\ C: \tau_{AC}^{\alpha} \times \eta_{AC}^{\beta} = 0.3^1 \times (1/1)^2 = 0.3 \\ D: \tau_{AD}^{\alpha} \times \eta_{AD}^{\beta} = 0.3^1 \times (1/2)^2 = 0.075 \end{cases}$$

归一化

$$p(B) = 0.033/(0.033+0.3+0.075) = 0.081$$
$$p(C) = 0.3/(0.033+0.3+0.075) = 0.74$$
$$p(D) = 0.075/(0.033+0.3+0.075) = 0.18$$

用轮盘赌法则选择下城市。假设产生的随机数q=random(0,1)=0.05，则蚂蚁1将会选择城市B。

用同样的方法为蚂蚁2和3选择下一访问城市，假设蚂蚁2选择城市D，蚂蚁3选择城市A。

5.2 算法流程

❖ 解：

步骤2.3：当前蚂蚁1所在城市i=B,路径记忆向量R^1=(AB)，可访问城市集合$J_1(i)$ ={C, D}。计算蚂蚁1选择C,D作为下一城市的概率：

$$B \Rightarrow \begin{cases} C: \tau_{BC}^{\alpha} \times \eta_{BC}^{\beta} = 0.3^1 \times (1/5)^2 = 0.012 \\ D: \tau_{BD}^{\alpha} \times \eta_{BD}^{\beta} = 0.3^1 \times (1/4)^2 = 0.019 \end{cases}$$

$$p(C) = 0.012/(0.012+0.019) = 0.39$$

$$p(D) = 0.019/(0.012+0.019) = 0.61$$

用轮盘赌法则选择下城市。假设产生的随机数q=random(0,1)=0.67，则蚂蚁1将会选择城市D。

用同样的方法为蚂蚁2和3选择下一访问城市，假设蚂蚁2选择城市C，蚂蚁3选择城市C。

5.2 算法流程

❖ 解：

步骤2.4：实际上此时路径已经构造完毕，蚂蚁1构建的路径为(*ABDCA*)。蚂蚁2构建的路径为(*BDCAB*)。蚂蚁3构建的路径为(*DACBD*)。

❖ 解：

步骤3：信息素更新。

计算每只蚂蚁构建的路径长度：C_1=3+4+2+1=10，C_2=4+2+1+3=10，C_3=2+1+5+4=12。更新每条边上的信息素：

$$\tau_{AB}=(1-\rho)\times\tau_{AB}+\sum_{k=1}^{3}\Delta\tau_{AB}^{k}=0.5\times0.3+(1/10+1/10)=0.35$$

$$\tau_{AC}=(1-\rho)\times\tau_{AC}+\sum_{k=1}^{3}\Delta\tau_{AC}^{k}=0.5\times0.3+(1/12)=0.16$$

……

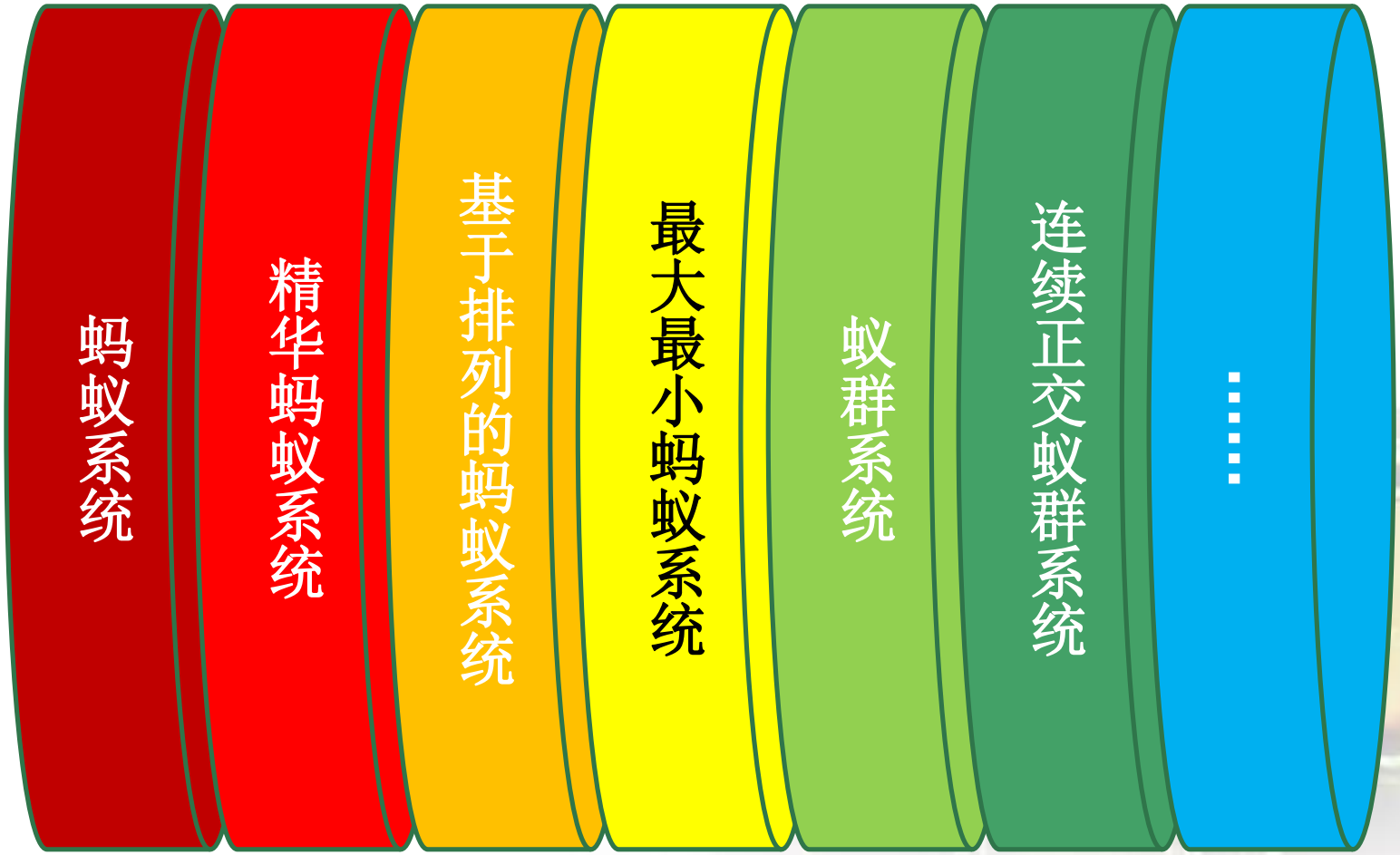
如此，根据公式(5.2)依次计算出问题空间内所有边更新后的信息素量。

5.2 算法流程

❖ 解：

步骤4：
如果满足结束条件，则输出全局最优结果并结束程序，否则，转向步骤2.1继续执行。

5.3 改进版本



5.3.1 精华蚂蚁系统

- 精华蚂蚁系统（Elitist Ant System，EAS）是对基础AS的第一次改进，它在原AS信息素更新原则的基础上增加了一个对至今最优路径的强化手段。

$$\tau(i,j)=(1-\rho)\cdot\tau(i,j)+\sum_{k=1}^{m}\Delta\tau_k(i,j)+e\Delta_b(i,j),$$

$$\Delta\tau_k(i,j)=\begin{cases}(C_k)^{-1}, & \text{if } (i,j)\in R^k\\ 0, & \text{otherwise}\end{cases},\quad \Delta\tau_b(i,j)=\begin{cases}(C_b)^{-1}, & \text{if } (i,j)\text{在路径}T_b\text{上}\\ 0, & \text{otherwise}\end{cases}$$

  - 引入这种额外的信息素强化手段有助于更好地引导蚂蚁搜索的偏向，使算法更快收敛。

5.3.2 基于排列的蚂蚁系统

- 基于排列的蚂蚁系统（**rank-based Ant System，AS_{rank}**）在**AS**的基础上给蚂蚁要释放的信息素大小加上一个权值，进一步加大各边信息素量的差异，以指导搜索。在每一轮所有蚂蚁构建完路径后，它们将按照所得路径的长短进行排名，只有生成了至今最优路径的蚂蚁和排名在前**(ω-1)**的蚂蚁才被允许释放信息素，蚂蚁在边**(i, j)**上释放的信息素的权值由蚂蚁的排名决定。

$$\tau(i,j)=(1-\rho)\cdot\tau(i,j)+\sum_{k=1}^{\omega-1}(\omega-k)\Delta\tau_k(i,j)+\omega\Delta_b(i,j),$$

$$\Delta\tau_k(i,j)=\begin{cases}(C_k)^{-1}, & \text{if } (i,j)\in R^k\\ 0, & \text{otherwise}\end{cases},\quad \Delta\tau_b(i,j)=\begin{cases}(C_b)^{-1}, & \text{if } (i,j)\text{在路径}T_b\text{上}\\ 0, & \text{otherwise}\end{cases}$$

  - 权值**(ω-k)**对不同路径的信息素浓度差异起到了一个放大的作用，AS_{rank}能更有力度地指导蚂蚁搜索。

5.3.3 最大最小蚂蚁系统

- 最大最小蚂蚁系统（**MAX-MIN Ant System，MMAS**）在基本**AS**算法的基础上进行了四项改进：

(1)只允许迭代最优蚂蚁（在本次迭代构建出最短路径的蚂蚁），或者至今最优蚂蚁释放信息素。

(2)信息素量大小的取值范围被限制在一个区间内。

(3)信息素初始值为信息素取值区间的上限，并伴随一个较小的信息素蒸发速率。

(4)每当系统进入停滞状态，问题空间内所有边上的信息素量都会被重新初始化。

5.3.3 最大最小蚂蚁系统

- 最大最小蚂蚁系统（**MAX-MIN Ant System，MMAS**）在基本**AS**算法的基础上进行了四项改进：

(1)只允许迭代最优蚂蚁（在本次迭代构建出最短路径的蚂蚁），或者至今最优蚂蚁释放信息素。（迭代最优更新规则和至今最优更新规则在**MMAS**中会被交替使用。）

- 如果只使用至今最优更新规则进行信息素的更新，搜索的导向性很强，算法会很快收敛到T_b附近；反之，如果只使用迭代最优更新规则，则算法的探索能力会得到增强，但收敛速度会下降。实验结果表明，对于小规模的**TSP**问题，仅仅使用迭代最优信息素更新方式即可。随着问题规模的增大，至今最优信息素规则的使用变得越来越重要。

5.3.3 最大最小蚂蚁系统

- 最大最小蚂蚁系统（**MAX-MIN Ant System，MMAS**）在基本**AS**算法的基础上进行了四项改进：

(2)信息素量大小的取值范围被限制在一个区间$[\tau_{min}, \tau_{max}]$内。

- 当信息素浓度也被限制在一个范围内以后，位于城市*i*的蚂蚁*k*选择城市*j*作为下一城市的概率也将被限制在一个区间内。算法有效避免了陷入停滞状态（所有蚂蚁不断重复搜索同一条路径）的可能性。

(3)信息素初始值为信息素取值区间的上限，并伴随一个较小的信息素蒸发速率。

- 增强算法在初始阶段的探索能力，有助于蚂蚁“视野开阔地”进行全局范围内的搜索。
- 随后蚂蚁逐渐缩小搜索范围。

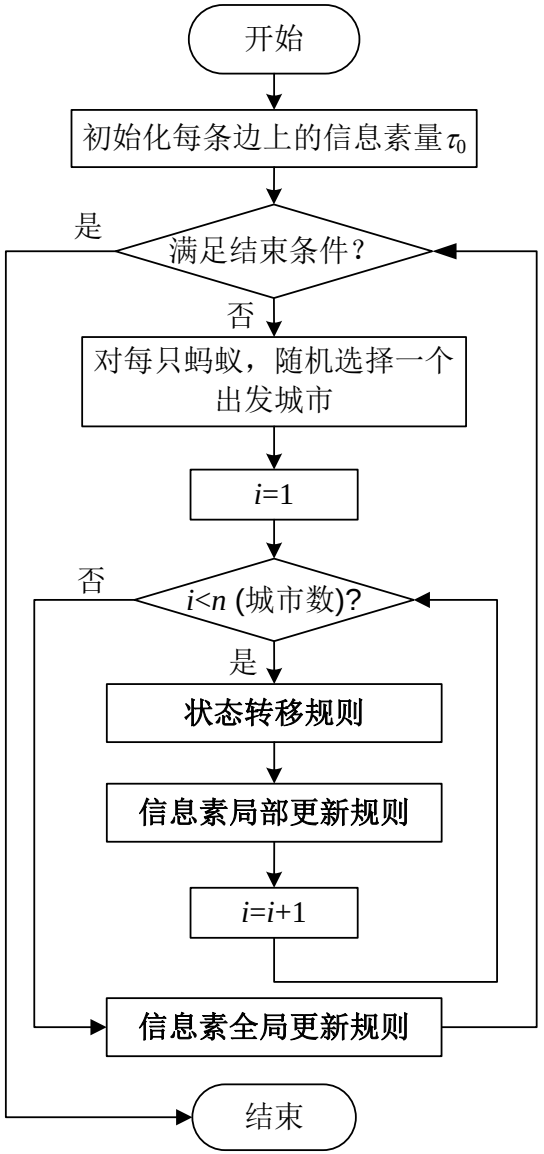
5.3.3 最大最小蚂蚁系统

- 最大最小蚂蚁系统（**MAX-MIN Ant System，MMAS**）在基本**AS**算法的基础上进行了四项改进：

(4)每当系统进入停滞状态，问题空间内所有边上的信息素量都会被重新初始化。（我们通常通过对各条边上信息素量大小的统计或是观察算法在指定次数的迭代内至今最优路径有无被更新来判断算法是否停滞。）

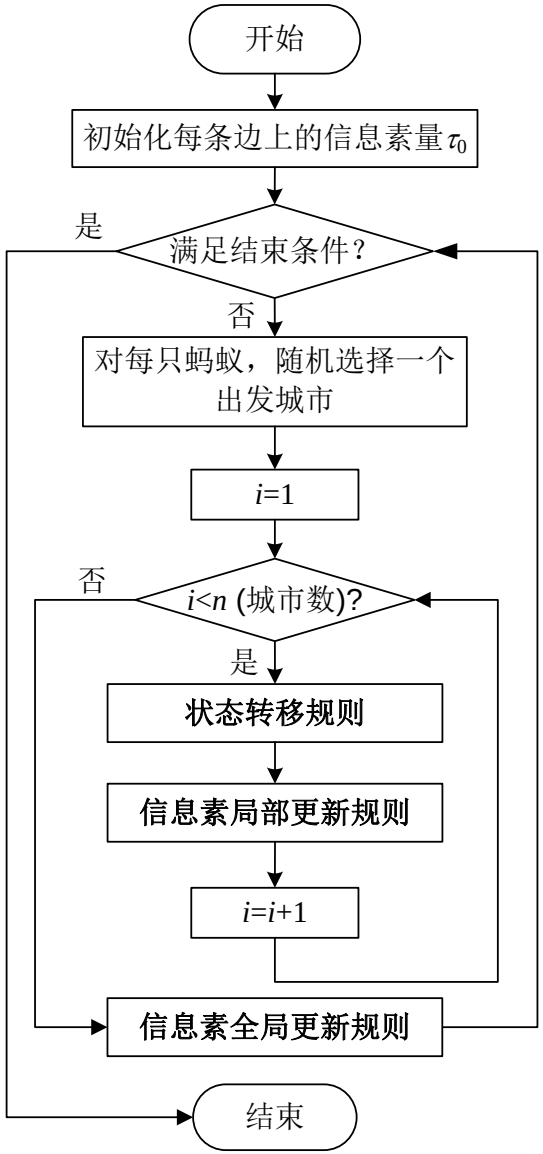
- 有效地利用系统进入停滞状态后的迭代周期继续进行搜索，使算法具有更强的全局寻优能力。

5.3.4 蚁群系统



- 1997年，蚁群算法的创始人Dorigo在“Ant colony system: a cooperative learning approach to the traveling salesman problem”一文中提出了一种具有全新机制的ACO算法——蚁群系统（Ant Colony System，ACS），进一步提高了ACO算法的性能。
- ACS是蚁群算法发展史上的一个里程碑式的作品，

5.3.4 蚁群系统



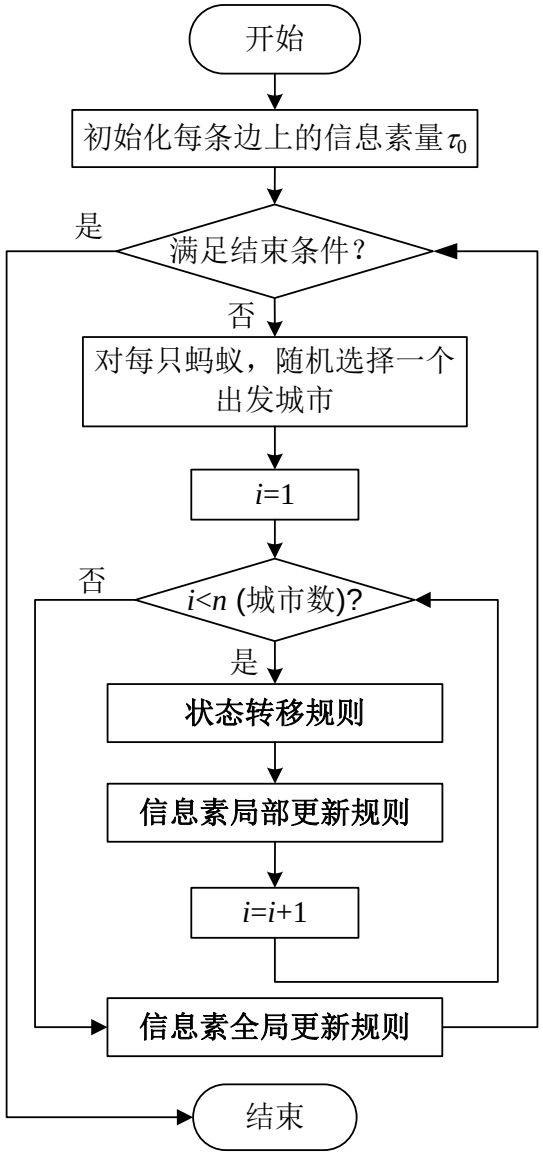
- （1）使用一种伪随机比例规则（pseudorandom proportional）选择下城市节点，建立开发当前路径与探索新路径之间的平衡。

$$j=\begin{cases}\arg\max_{j\in J_k(i)}\left\{[\tau(i,j)],[\eta(i,j)]^{\beta}\right\}, & \text{if } q\le q_0\\ S, & \text{otherwise}\end{cases}$$

q0是一个[0, 1]区间内的参数，当产生的随机数q≤q0时，蚂蚁直接选择使启发式信息与信息素量的指数乘积最大的下城市节点，我们通常称之为开发（exploitation）；反之，当产生的随机数q>q0时ACS将和各种AS算法一样使用轮盘赌选择策略，我们称之为偏向探索（bias exploration）。

通过调整q0，我们能有效调节“开发”与“探索”之间的平衡，以决定算法是集中开发最优路径附近的区域，还是探索其它的区域。

5.3.4 蚁群系统



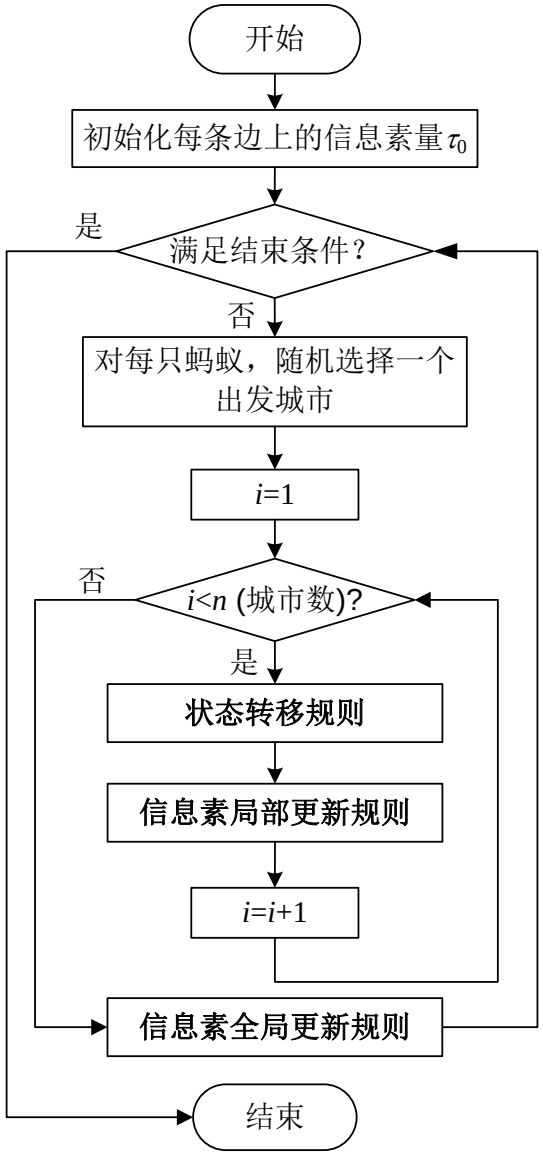
- （2）使用信息素全局更新规则，每轮迭代中所有蚂蚁都已构建完路径后，在属于至今最优路径的边上蒸发和释放信息素。

$$\tau(i,j)=(1-\rho)\cdot\tau(i,j)+\rho\cdot\Delta\tau_b(i,j),\quad \forall(i,j)\in T_b$$

$$\Delta\tau_b(i,j)=1/C_{\mathrm{b}}$$

不论是信息素的蒸发还是释放，都只在属于至今最优路径的边上进行，这里与AS有很大的区别。因为AS算法将信息素的更新应用到了系统的所有边上，信息素更新的计算复杂度为O（n^2），而ACS算法的信息素更新计算复杂度降低为O（n）。参数ρ代表信息素蒸发的速率，新增加的信息素 被乘上系数ρ后，更新后的信息素浓度被控制在旧信息素量与新释放的信息素量之间，用一种隐含的又更简单的方式实现了MMAS算法中对信息素量取值范围的限制。

5.3.4 蚁群系统

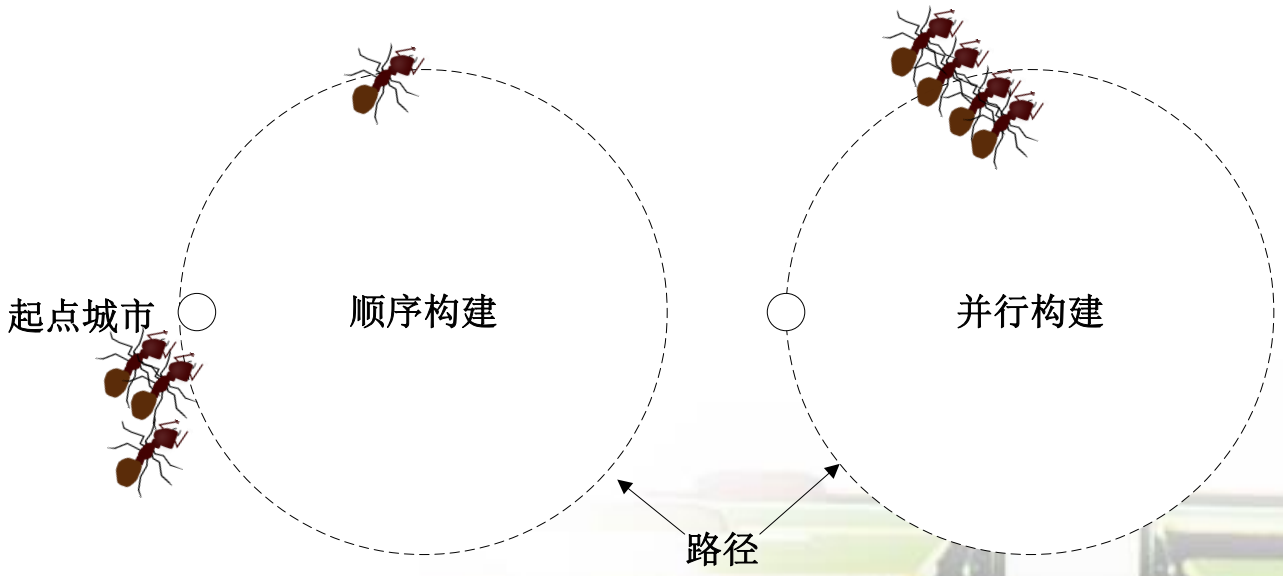


- （3）引入信息素局部更新规则，在路径构建过程中，对每一只蚂蚁，每当其经过一条边(i, j)时，它将立刻对这条边进行信息素的更新。

$$\tau(i,j)=(1-\xi)\cdot\tau(i,j)+\xi\cdot\tau_0$$

信息素局部更新规则作用于某条边上会使得这条边被其他蚂蚁选中的概率减少。这种机制大大增加了算法的探索能力，后续蚂蚁倾向于探索未被使用过的边，有效地避免了算法进入停滞状态。

5.3.4 蚁群系统

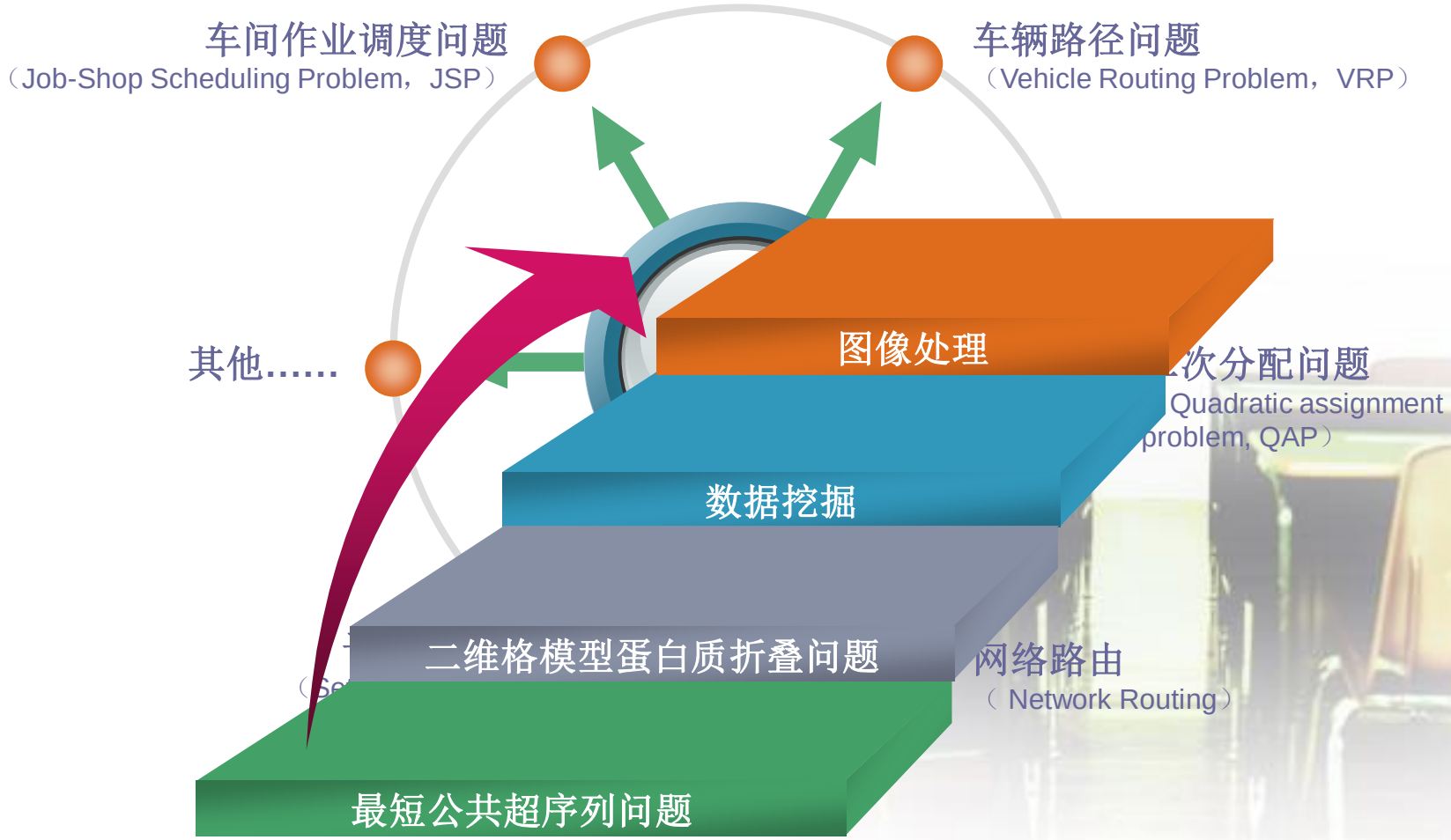


顺序构建和**并行构建**。顺序构建是指当一只蚂蚁完成一轮完整的构建并返回到初始城市之后，下一只蚂蚁才开始构建；并行构建是指所有蚂蚁同时开始构建，每次所有蚂蚁各走一步（从当前城市移动到下一个城市）。对于ACS，要注意到两种路径构建方式会造成算法行为的区别。
在ACS中通常我们选择让所有蚂蚁并行地工作。

5.3.5 连续正交蚁群系统

- 连续正交蚁群算法（**Continuous Orthogonal Ant Colony, COAC**）：近年来，将应用领域扩展到连续空间的蚁群算法也在发展，连续正交蚁群就是其中比较优秀的一种。**COAC**通过在问题空间内自适应地选择和调整一定数量的区域，并利用蚂蚁在这些区域内进行正交搜索、在区域间进行状态转移、并更新各个区域的信息素来搜索问题空间中的最优解。
- **COAC**的基本思想是利用正交试验的方法将连续空间离散化。

5.4 相关应用


车间作业调度问题
（Job-Shop Scheduling Problem，JSP）
车辆路径问题
（Vehicle Routing Problem，VRP）
其他……
图像处理
次分配问题
Quadratic assignment
problem, QAP）
数据挖掘
二维格模型蛋白质折叠问题
网络路由
（ Network Routing）
最短公共超序列问题

5.5 参数设置

参数	参考设置
蚂蚁数目m	在用AS、EAS、AS_{rank}和MMAS求解TSP问题时，m取值等于城市数目n算法有较好性能；而对于ACS，m=10比较合适。
信息素权重α与启发式信息权重β	在各类ACO算法中设置α=1，β=2~5比较合适。
信息素挥发因子ρ	对于AS和EAS，r=0.5；对于AS_{rank}，ρ=0.1；对于MMAS，ρ=0.02；对于ACS，ρ=0.1，算法的综合性能较高。
初始信息素量τ_0	对于AS，$\tau_0=m/C^{nn}$；对于EAS，$\tau_0=(e+m)/rC^{nn}$；对于AS_{rank}，$\tau_0=0.5r(r-1)/rC^{nn}$；对于MMAS，$\tau_0=1/rC^{nn}$；对于ACS，$\tau_0=1/nC^{nn}$。
释放信息素的蚂蚁个数ω	在AS_{rank}中，参数ω设置为ω=6。
进化停滞判定代数rs	在MMAS中，参数rs设置为rs=25。
信息素局部挥发因子ξ	在ACS中，参数ξ设置为ξ=0.1。
伪随机因子q_0	在ACS中，参数q_0设置为q_0=0.1。


Thank You !