人工智能

Lecture 8: 群体智能 (Swarm Intelligence) Part2

赵培海

东华大学
计算机科学与技术学院

2023 年 9 月 6 日

内容组织

1. 粒子群算法
2. 量子粒子群算法
3. 蚁群算法
4. 自适应蚁群算法

内容组织

1. 粒子群算法

2. 量子粒子群算法

3. 蚁群算法

4. 自适应蚁群算法

群体智能

- 群智能算法 (Swarm Intelligence，SI)：受动物群体智能启发的算法.
- 群体智能：由简单个体组成的群落与环境以及个体之间的互动行为.
- 群智能算法包括：粒子群优化算法、蚁群算法、蜂群算法、……

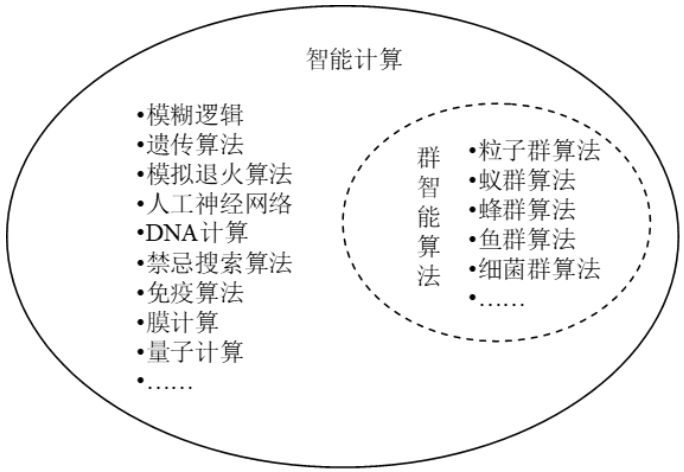


图: SI 概览

粒子群算法 (Particle Swarm Optimization)

- 产生背景
  - 粒子群优化 (Particle Swarm Optimization, PSO) 算法是由美国普渡大学的 Kennedy 和 Eberhart 于 1995 年提出，它的基本概念源于对鸟群觅食行为的研究.
- 设想这样一个场景:
  - 一群鸟在随机搜寻食物，在这个区域里只有一块食物，所有的鸟都不知道食物在哪里，但是它们知道当前的位置离食物还有多远. 那么找到食物的最优策略是什么呢?
- 最简单有效的就是搜寻目前离食物最近的鸟的周围区域.

粒子群算法 (Particle Swarm Optimization)

基本思想

- 将每个个体看作 n 维搜索空间中一个没有体积质量的粒子，在搜索空间中以一定的速度飞行，该速度决定粒子飞行的方向和距离. 所有粒子还有一个由被优化的函数决定的适应值.

基本原理

- PSO 初始化为一群随机粒子，然后通过迭代找到最优解. 在每一次迭代中，粒子通过跟踪两个“极值”来更新自己. 第一个就是粒子本身所找到的最优解，这个解称为个体极值. 另个一是整个种群目前找到的最优解，这个解称为全局极值.

PSO 的基本原理

在 n 维连续搜索空间中，对粒子群中的第 i $(i = 1, 2, \cdots, m)$ 个粒子进行定义：

- $x^i(k) = [x_1^i, x_2^i, \cdots, x_n^i]^T$：表示搜索空间中粒子的当前位置.
- $p^i(k) = [p_1^i, p_2^i, \cdots, p_n^i]^T$：表示该粒子至今所获得的具有最优适应度 $f_p^i(k)$ 的位置.
- $v^i(k) = [v_1^i, v_2^i, \cdots, v_n^i]^T$：表示该粒子的搜索方向.

PSO 的基本原理

- 每个粒子经历过的最优位置（pbest）记为 $p^i(k) = [p_1^i, p_2^i, \cdots, p_n^i]^T$
- 群体经历过的最优位置 (gbest) 记为 $p^g(k) = [p_1^g, p_2^g, \cdots, p_n^g]^T$
- 则基本的 PSO 算法为：

$$v_j^i(k+1) = w(k)v_j^i(k) + \phi_1 rand(0, a_1)\left(p_j^i(k) - x_j^i(k)\right) + \phi_2 rand(0, a_2)\left(p_j^g(k) - x_j^i(k)\right)$$

$$x_j^i(k+1) = x_j^i(k) + v_j^i(k+1)$$

$$i = 1, 2, \cdots, m; \quad j = 1, 2, \cdots, n$$

  - 其中，w 是惯性权重因子
  - ϕ_1, ϕ_2 是加速度常数，均为非负值
  - $rand(0, a_1)$ 和 $rand(0, a_2)$ 为 $[0, a_1]$、$[0, a_2]$ 范围内的具有均匀分布的随机数，a_1 与 a_2 为相应的控制参数.

PSO 的基本原理

$$v_j^i(k+1) = w(k)v_j^i(k) + \phi_1 rand(0, a_1)\left(p_j^i(k) - x_j^i(k)\right) + \phi_2 rand(0, a_2)\left(p_j^g(k) - x_j^i(k)\right)$$

- 第 1 部分是粒子在前一时刻的速度;
- 第 2 部分为个体“认知”分量，表示粒子本身的思考，将现有的位置和曾经经历过的最优位置相比.
- 第 3 部分是群体“社会 (social)”分量，表示粒子间的信息共享与相互合作.
- ϕ_1, ϕ_2 分别控制个体认知分量和群体社会分量相对贡献的学习率.
- 随机系数增加搜索方向的随机性和算法多样性.

PSO 的基本原理

基于学习率 ϕ_1, ϕ_2，Kennedy 给出以下 4 种类型的 PSO 模型：

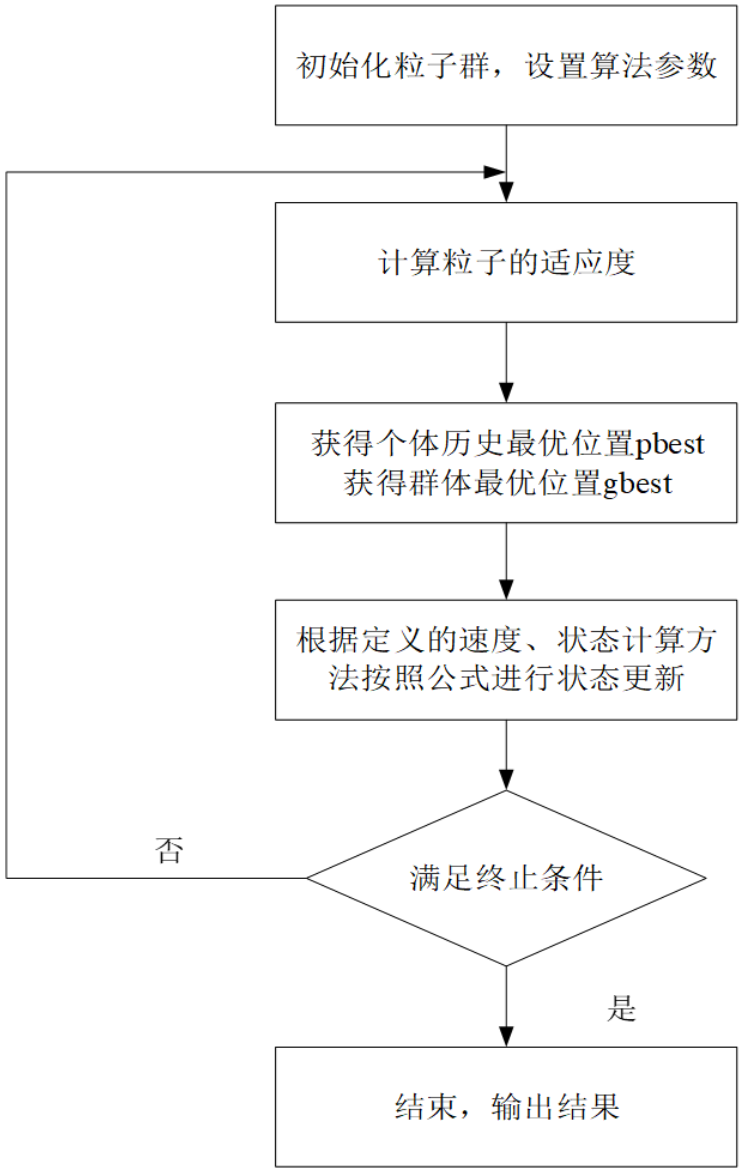
- 若 $\phi_1 > 0,\ \phi_2 > 0$，则称该算法为 PSO 全模型.
- 若 $\phi_1 > 0,\ \phi_2 = 0$，则称该算法为 PSO 认知模型.
- 若 $\phi_1 = 0,\ \phi_2 > 0$，则称该算法为 PSO 社会模型.
- 若 $\phi_1 = 0,\ \phi_2 = 0\ and\ g \neq i$，则称该算法为 PSO 无私模型.

PSO 的基本原理

粒子群优化算法的流程：

1. 初始化每个粒子，即在允许范围内随机设置每个粒子的初始位置和速度.
2. 评价每个粒子的适应度，计算每个粒子的目标函数.
3. 设置每个粒子的 P_i，对每个粒子，将其适应度与其经历过的最好位置 P_i 进行比较，如果优于 P_i，则将其作为该粒子的最好位置 P_i.
4. 设置全局最优值 P_g，对每个粒子，将其适应度与群体经历过的最好位置 P_g 进行比较，如果优于 P_g，则将其作为当前群体的最好位置 P_g.
5. 更新粒子的速度和位置.
6. 检查终止条件. 如果未达到设定条件（预设误差或者迭代的次数），则返回第 2 步.

PSO 算法流程图


初始化粒子群，设置算法参数
计算粒子的适应度
获得个体历史最优位置pbest
获得群体最优位置gbest
根据定义的速度、状态计算方法按照公式进行状态更新
满足终止条件
否
是
结束，输出结果

PSO 算法的参数分析

PSO 算法的参数包括：群体规模 m，惯性权重 w，加速度 ϕ_1, ϕ_2，最大速度 V_{max}，最大代数 G_{max}.

1. 最大速度 V_{max}

   对速度 v_i，算法中有最大速度 V_{max} 作为限制，如果当前粒子的某维速度大于最大速度 V_{max}，则该维的速度就被限制为最大速度 V_{max}.

2. 权重因子

   3 个权重因子：惯性权重 w，加速度 ϕ_1, ϕ_2.

PSO 算法的参数分析

$$v_j^i(k+1) = w(k)v_j^i(k) + \phi_1 rand(0, a_1)\left(p_j^i(k) - x_j^i(k)\right) + \phi_2 rand(0, a_2)\left(p_j^g(k) - x_j^i(k)\right)$$

位置更新方程中各部分的影响：

(1) 只有第 1 部分，即 $\phi_1 = \phi_2 = 0$

粒子将一直以当前的速度飞行，直到达边界.

由于它只能搜索有限的区域，所以很难找到好解.

PSO 算法的参数分析

$$v_j^i(k+1) = w(k)v_j^i(k) + \phi_1 rand(0, a_1)\left(p_j^i(k) - x_j^i(k)\right) + \phi_2 rand(0, a_2)\left(p_j^g(k) - x_j^i(k)\right)$$

位置更新方程中各部分的影响：

(1) 只有第 1 部分，即 $\phi_1 = \phi_2 = 0$

粒子将一直以当前的速度飞行，直到达边界.

由于它只能搜索有限的区域，所以很难找到好解.

(2) 没有第 1 部分，即 $w = 0$

速度只取决于粒子当前位置和其历史最好位置 P_i 和 P_g，速度本身没有记忆性.

PSO 算法的参数分析

$$v_j^i(k+1) = w(k)v_j^i(k) + \phi_1 rand(0, a_1)\left(p_j^i(k) - x_j^i(k)\right) + \phi_2 rand(0, a_2)\left(p_j^g(k) - x_j^i(k)\right)$$

位置更新方程中各部分的影响：

(3) 没有第 2 部分，即 $\phi_1 = 0$

粒子没有认知能力，也就是“只有社会模型”.

在粒子的相互作用下，有能力达到新的搜索空间．但对复杂问题，容易陷入局部最优点.

PSO 算法的参数分析

$$v_j^i(k+1) = w(k)v_j^i(k) + \phi_1 rand(0, a_1)\left(p_j^i(k) - x_j^i(k)\right) + \phi_2 rand(0, a_2)\left(p_j^g(k) - x_j^i(k)\right)$$

位置更新方程中各部分的影响：

(3) 没有第 2 部分，即 $\phi_1 = 0$

粒子没有认知能力，也就是“只有社会模型”.

在粒子的相互作用下，有能力达到新的搜索空间．但对复杂问题，容易陷入局部最优点.

(4) 没有第 3 部分，即 $\phi_2 = 0$

粒子间没有社会共享信息，也就是“只有认知”模型.

因为个体间没有交互，一个规模为 M 的群体等价于 M 个单个粒子的运行，因而得到最优解的机率非常小.

PSO 算法的参数分析

参数设置 (早期的实验)

- w 固定为 1.0
- ϕ_1 和 ϕ_2 固定为 2.0
- V_{max} 成为唯一需要调节的参数，通常设为每维变化范围 $10\% \sim 20\%$
- Suganthan 的实验表明，ϕ_1 和 ϕ_2 为常数时可以得到较好的解，但不一定必须为 2.

内容组织

1. 粒子群算法
2. 量子粒子群算法
3. 蚁群算法
4. 自适应蚁群算法

量子粒子群算法

产生背景

- 在量子力学中是没有确定的轨迹的，因为根据不确定性原理，位置向量 x_i 和速度向量 v_i 是不可能同时确定的.
- J. Sun 受到量子物理学的启发,于 2004 年提出了一种能够保证全局收敛的具有量子行为的量子粒子群优化 (Quantum-behaved particle swarm optimization，QPSO) 算法，并对算法的收敛性进行了分析.

量子粒子群算法

基本量子粒子群优化算法

1. 粒子不再被描述为位置向量 x_i 和速度向量 v_i，而采用波函数表示.

   - 粒子的波函数为

$$\Psi(x) = \frac{1}{\sqrt{L}} \exp\left(-\frac{||p-x||}{L}\right)$$

     其概率密度函数为

$$Q(x) = |\Psi(x)|^2 = \frac{1}{L} \exp\left(-2\frac{||p-x||}{L}\right)$$

     其中，p 为每个粒子历史的最好位置.

   - 参数 L 的取值定义为

$$L_{t+1} = 2\alpha \times |p - x(t)|$$

     L 指出了微粒的搜索空间范围.

量子粒子群算法

基本量子粒子群优化算法

2. 引入了 $mbest$ 为所有微粒的中心

$$\Downarrow$$

$$mbest = \sum_{i=1}^{M} \frac{p_i}{M} = \left(\sum_{i=1}^{M} \frac{p_{i1}}{M}, \sum_{i=1}^{M} \frac{p_{i2}}{M}, \cdots, \sum_{i=1}^{M} \frac{p_{id}}{M} \right)$$

- 其中，M 是种群数目
- p_i 是第 i 个粒子的最好位置
- 通过将所有粒子的中心 $mbest$ 取代每个粒子的最好位置 p，可以有效提高算法的全局搜索能力.

量子粒子群算法

基本量子粒子群优化算法

- 参数 L 表示为

$$L_{t+1} = 2\beta \times |mbest - x(t)|$$

β 为收敛系数，不同的 β 影响算法收敛速度，一般取 β 的值为

$$\beta = \frac{(1.0 - 0.5) \times (MAXITER - T)}{MAXITER} + 0.5$$

$MAXITER$ 为最大迭代次数.

- 由概率密度函数通过 Monte Carlo 算法计算得到

$$x(t) = p \pm \frac{L}{2 \ln\left(\frac{1}{u}\right)}$$

- 代入参数 L，QPSO 算法的进化方程为

$$x(t+1) = p \pm \beta |mbest - x(t)| \ln u$$

量子粒子群算法

基本量子粒子群优化算法

- 量子粒子群优化算法的基本步骤：
  1. 确定种群规模和粒子维数，初始化粒子群体；
  2. 计算个体历史最优值 (pbest)：计算每一个微粒的适应度值，通过和个体的历史最优值比较，如果当前值优于个体历史最优值，则把当前值替换为个体最优值 (pbest)，否则不替换；
  3. 计算所有微粒的适应值，与当前全局最优值 (gbest) 比较，若当前值优于全局最优值，则把当前值替换为全局最优值；
  4. 计算所有粒子的重心（mbest)；更新所有粒子的重心（mbest)；
  5. 根据进化方程更新每个粒子的位置，产生新种群；
  6. 粒子适应度满足收敛条件或者是达到最大迭代次数，则算法结束，否则跳转到步骤 2 继续迭代执行.

基本量子粒子群优化算法

- 优点:
  - 相对于粒子群优化算法具有更好的收敛性和全局搜索能力.
- 缺点:
  - 求解约束优化问题的时
    - 会产生大量的不可行解
    - 破坏种群的多样性
    - 导致算法陷入局部极值

粒子群优化算法应用领域

粒子群优化算法已在诸多领域得到应用，归纳如下：

1. 神经网络训练
2. 机器人领域
3. 运筹学领域
4. 图像处理领域
5. 生物信息领域
6. ...

粒子群优化算法应用领域

粒子群优化算法在 PID 参数整定中的应用

- 典型的 PID 控制系统的控制量 u 与偏差 $e = (R - y)$ 之间满足以下差分方程

$$u(n) = K_p \left[e(n) + \frac{1}{T} \sum_{k=0}^{n} e(k)T + T_d \frac{e(n) - e(n-1)}{T} \right]$$

- PID 控制器就是通过调整 K_p, T_i, T_d 这 3 个参数来使系统的控制性能达到给定的要求.
- 从最优控制的角度，就是在 K_p, T_i, T_d 这 3 个变量的参数空间中，寻找最优的值使系统的控制性能达到最优.
- 为优化 PID 参数，可以选取如下函数作为评价控制性能的指标

$$Q = \int_0^{\infty} t|e(t)|dt$$

粒子群优化算法应用领域

粒子群优化算法在 PID 参数整定中的应用

1. 编码与初始种群
   - 实数编码
   - 在初始群体的生成上，首先根据经验估计出 PID 三个参数的取值范围，在此范围内采用随机生成的方式，使粒子群优化算法在整个可行解空间中进行搜索.
2. 适应度函数
   - 由于 PID 参数优化是求目标函数 Q 的极小值问题，因而需要将极小值问题转换为极大值问题，适应度函数可以取为

$$F = \frac{1}{\int_0^\infty t|e(t)|dt}$$

粒子群优化算法应用领域

粒子群优化算法在 PID 参数整定中的应用

- 举例
  - 采用 PID 控制器对被控对象进行控制，假定控制对象具有二阶惯性加延迟的模型，其传递函数为:

$$H(s) = \frac{e^{-0.4s}}{(0.3s+1)^2}$$

  - 假定采样周期选择为 $0.1s$，根据经验 K_p 参数范围为 $(0,4)$，T_i 参数范围为 $(0,1)$，T_d 参数范围为 $(0,1)$.
  - 取粒子群种群规模为 20，迭代次数为 50，c_1 的取值根据迭代的次数线性减小，初始值为 1.5，最终值 0.4，$\phi_1 = \phi_2 = 2$.

粒子群优化算法应用领域

粒子群优化算法在 PID 参数整定中的应用

- 举例

  PID 参数粒子群优化算法寻优结果见表所示. 为了说明粒子群优化算法的有效性，表中同时也给出了用单纯形法的寻优结果.

表: 优化结果及比较

算法	K_p	T_i	T_d	Q
粒子群优化算法	0.62932	0.59349	0.23715	4.84232
单纯形法	0.63057	0.59481	0.23703	4.86818

粒子群优化算法应用领域 2

粒子群优化算法在车辆路径问题中的应用

1. 车辆路径问题（VRP）的模型

- 车辆路径问题：假定配送中心最多可以用 K $(k = 1, 2, \cdots, K)$ 辆车对 L $(i = 1, 2, \cdots, L)$ 个客户进行运输配送，$i = 0$ 表示仓库. 每个车辆载重为 b_k $(k = 1, 2, \cdots, K)$，每个客户的需求为 d_i $(i = 1, 2, \cdots, L)$，客户 i 到客户 j 的运输成本为 c_{ij}（可以是距离，时间，费用等）. 定义如下变量：

$$y_{ik} = \begin{cases} 1, & \text{客户 } i \text{ 由车辆 } k \text{ 配送} \\ 0, & \text{其它} \end{cases}$$

$$x_{ijk} = \begin{cases} 1, & \text{车辆 } k \text{ 从 } i \text{ 访问 } j \\ 0, & \text{其它} \end{cases}$$

粒子群优化算法应用领域 2

粒子群优化算法在车辆路径问题中的应用

1. 车辆路径问题（VRP）的模型

- 则车辆路径问题的数学模型如下表示：

$$\min \sum_{k=1}^{K} \sum_{i=0}^{L} \sum_{j=0}^{L} c_{ij} x_{ijk}$$

$\forall k,\ \sum_{i=1}^{L} d_i y_{ik} \leq b_k$	每辆车的能力约束
$\forall i,\ \sum_{k=1}^{K} y_{ik} = 1$	保证每个客户都被服务
$\forall j,k,\ \sum_{i=1}^{L} x_{ijk} = y_{jk}$	保证客户仅被一辆车访问
$\forall i,k,\ \sum_{j=1}^{L} x_{ijk} = y_{ik}$	保证客户仅被一辆车访问
$\forall k,\ \sum_{i,j \in S \times S} x_{ijk} \leq \lvert S \rvert - 1,\ S \in \{1,2,\cdots,L\}$	消除子回路
$\forall i,j,k,\ x_{ijk} = 0\ or\ 1$	表示变量的取值范围
$\forall i,k,\ y_{ik} = 0\ or\ 1$	表示变量的取值范围

粒子群优化算法应用领域 2

粒子群优化算法在车辆路径问题中的应用

2. 编码与初始种群

- 对这类组合优化问题，编码方式、初始解的设置对问题的求解都有很大的影响.
- 采用常用的自然数编码方式.
- 对于 K 辆车和 L 个客户的问题，用从 1 到 L 的自然数随机排列来产生一组解 $\mathbf{x} = (x_1, x_2, \cdots, x_L)$. 然后分别用节约法或者最近插入法构造初始解.

粒子群优化算法应用领域 2

粒子群优化算法在车辆路径问题中的应用

3. 实验结果

- 粒子群优化算法的各个参数设置如下：种群规模 $p = 50$，迭代次数 $N = 1000$，c_1 的初始值为 1，随着迭代的进行，线性减小到 0，$c_2 = c_3 = 1.4$，$|V_{max}| \leq 100$.

优化结果及其比较

实例	PSO		GA	
	best	dev(%)	best	dev(%)
A-n32-k5	829	5.73	818	4.34
A-n33-k5	705	6.65	674	1.97
A-n34-k5	832	6.94	821	5.52
A-n39-k6	872	6.08	866	5.35
A-n44-k6	1016	8.49	991	5.76
A-n46-k7	977	6.89	957	4.7
A-n54-k7	1205	3.26	1203	3.08
A-n60-k9	1476	9.01	1410	4.13
A-n69-k9	1275	10	1243	7.24
A-n80-k10	1992	12.98	1871	6.12

内容组织

1. 粒子群算法

2. 量子粒子群算法

3. 蚁群算法

4. 自适应蚁群算法

蚁群算法 (Ant Colony Optimization)

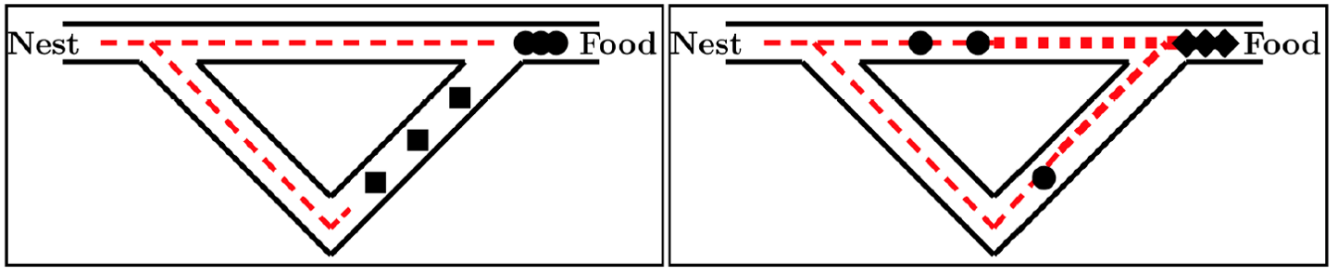
产生背景

- 20 世纪 90 年代初，意大利科学家 Marco Dorigo 等受蚂蚁觅食行为的启发，提出蚁群算法 (Ant Colony Optimization，ACO).
- 一种应用于组合优化问题的启发式搜索算法.
- 在解决离散组合优化方面具有良好的性能.

基本蚁群算法

基本思想

- 信息素跟踪：按照一定的概率沿着信息素较强的路径觅食.
- 信息素遗留：会在走过的路上会释放信息素，使得在一定的范围内的其他蚂蚁能够觉察到并由此影响它们的行为.



基本蚁群算法

- 环境：有障碍物、有其他蚂蚁、有信息素.
- 觅食规则：范围内寻找是否有食物，否则看是否有信息素，每只蚂蚁都会以小概率犯错.
- 移动规则：都朝信息素最多的方向移动，无信息素则继续朝原方向移动，且有随机的小的扰动，有记忆性.
- 避障规则：移动的方向如有障碍物挡住，蚂蚁会随机选择另一个方向.
- 信息素规则：越靠近食物播撒的信息素越多，越离开食物播撒的信息素越少.

基本蚁群算法模型

蚁群优化算法的第一个应用是著名的旅行商问题.

- 旅行商问题（Traveling Salesman Problem，TSP）:

  在寻求单一旅行者由起点出发，通过所有给定的需求点之后，最后再回到原点的最小路径成本.

$$\Downarrow$$

- 蚂蚁搜索食物的过程:

  通过个体之间的信息交流与相互协作最终找到从蚁穴到食物源的最短路径.

基本蚁群算法模型

蚁群系统的模型

m	是蚁群中蚂蚁的数量
$d_{xy}(x, y = 1, \cdots, n)$	表示元素 (城市) 和元素 (城市) 之间的距离
$\eta_{xy}(t)$	表示能见度，称为启发信息函数，等于距离的倒数，即 $\eta_{xy}(t) = \frac{1}{d_{xy}}$
$b_x(t)$	表示 t 时刻位于城市 x 的蚂蚁的个数，$m = \sum_{x=1}^{n} b_x(t)$
$\tau_{xy}(t)$	表示 t 时刻在 xy 连线上残留的信息素，初始时刻，各条路径上的信息素相等即 $\tau_{xy}(0) = C(const)$

蚂蚁 k 在运动过程中，根据各条路径上的信息素决定转移方向.

基本蚁群算法模型

蚁群系统的模型

- $P_{xy}^k(t)$ 表示在 t 时刻蚂蚁 k 选择从元素 (城市)x 转移到元素 (城市)y 的概率，也称为随机比例规则.
- $P_{xy}^k(t)$ 由信息素 $\tau_{xy}(t)$ 和局部启发信息 $\eta_{xy}(t)$ 共同决定.

基本蚁群算法模型

$P_{xy}^k(t)$ 表示如下：

$$P_{xy}^k(t) = \begin{cases} \dfrac{|\tau_{xy}(t)|^\alpha |\eta_{xy}(t)|^\beta}{\sum_{y \in allowed_k(x)} |\tau_{xy}(t)|^\alpha |\eta_{xy}(t)|^\beta}, & if\ y \in allowed_k(x) \\ 0, & others \end{cases}$$

其中：

- $allowed_k(x) = \{0, 1, \cdots, n-1\} - tabu_k(x)$ 表示蚂蚁 k 下一步允许选择的城市
- $tabu_k(x)$ $(k = 1, 2, \cdots, m)$ 记录蚂蚁 k 当前所走过的城市
- α 是信息素启发式因子，表示轨迹的相对重要性

基本蚁群算法模型

$P_{xy}^k(t)$ 表示如下：

$$P_{xy}^k(t) = \begin{cases} \frac{|\tau_{xy}(t)|^\alpha |\eta_{xy}(t)|^\beta}{\sum_{y \in allowed_k(x)} |\tau_{xy}(t)|^\alpha |\eta_{xy}(t)|^\beta}, & if\ y \in allowed_k(x) \\ 0, & others \end{cases}$$

其中：

- α 值越大：该蚂蚁越倾向于选择其它蚂蚁经过的路径，该状态转移概率越接近于贪婪规则.
- 当 $\alpha = 0$ 时：不再考虑信息素水平，算法就成为有多重起点的随机贪婪算法.
- 当 $\beta = 0$ 时：算法就成为纯粹的正反馈的启发式算法.

基本蚁群算法模型

用参数 $1-\rho$ 表示信息素消逝程度，蚂蚁完成一次循环，各路径上信息素浓度消散规则为：

$$\tau_{xy}(t+1)=\rho\tau_{xy}(t)+\Delta\tau_{xy}(t)$$

蚁群的信息素浓度更新规则为：

$$\Delta\tau_{xy}(t)=\sum_{k=1}^{m}\Delta\tau_{xy}^{k}(t)$$

M. Dorigo 给出 $\Delta\tau_{xy}^{k}(t)$ 的三种不同模型

基本蚁群算法模型

1. 蚂蚁圈系统（Ant-cycle System）

- 单只蚂蚁所访问路径上的信息素浓度更新规则为：

$$\Delta\tau_{xy}^{k}(t)=\begin{cases}\frac{Q}{L_k}, & \text{若第 } k \text{ 只蚂蚁在本次循环中从 } x \text{ 到 } y\\ 0, & others\end{cases}$$

- 其中：
  - $\tau_{xy}(t)$ 为当前路径上的信息素
  - $\Delta\tau_{xy}(t)$ 为路径 (x,y) 上信息素的增量
  - $\Delta\tau_{xy}^{k}(t)$ 为第 k 只蚂蚁留在路径 (x,y) 上的信息素的增量
  - Q 为常数
  - L_k 为优化问题的目标函数值，表示第 k 只蚂蚁在本次循环中所走路径的长度

基本蚁群算法模型

2. 蚂蚁数量系统（Ant-quantity System）

$$\Delta\tau_{xy}^{k}(t)=\begin{cases}\frac{Q}{d_k}, & \text{若第 } k \text{ 只蚂蚁在本次循环中从 } x \text{ 到 } y\\ 0, & others\end{cases}$$

3. 蚂蚁密度系统（Ant-density System）

$$\Delta\tau_{xy}^{k}(t)=\begin{cases}Q, & \text{若第 } k \text{ 只蚂蚁在本次循环中从 } x \text{ 到 } y\\ 0, & others\end{cases}$$

基本蚁群算法模型

三种模型比较：

- 蚂蚁圈系统：利用的是全局信息 $\frac{Q}{L_k}$，即蚂蚁完成一个循环后，更新所有路径上的信息．(效果最好，通常作为蚁群优化算法的基本模型)
- 蚂蚁数量系统：利用的是局部信息 $\frac{Q}{d_{xy}}$，即蚂蚁每走一步都要更新残留信息素的浓度．
- 蚂蚁密度系统：利用的是局部信息 Q，即蚂蚁每走一步都要更新残留信息素的浓度．

基本蚁群算法模型

全局信息更新方法

- 优点
  - 保证了残留信息素不至于无限累积；
  - 如果路径没有被选中，那么上面的残留信息素会随时间的推移而逐渐减弱，这使算法能“忘记”不好的路径；
  - 即使路径经常被访问也不至于因为 $\Delta\tau_{xy}^{k}(t)$ 的累积,而产生 $\Delta\tau_{xy}^{k}(t) \gg \eta_{xy}(t)$ 使期望值的作用无法体现；
  - 充分体现了算法中全局范围内较短路径 (较好解) 的生存能力；
  - 加强了信息正反馈性能；
  - 提高了系统搜索收敛的速度.

蚁群算法的参数选择

信息素启发因子 α

- 反映了蚁群在路径搜索中随机性因素作用的强度；
- α 值越大，蚂蚁选择以前走过的路径的可能性越大，搜索的随机性减弱；
- 当 α 过大时会使蚁群的搜索过早陷于局部最优.

期望值启发式因子 β

- 反映了蚁群在路径搜索中先验性、确定性因素作用的强度；
- β 值越大，蚂蚁在某个局部点上选择局部最短路径的可能性越大；
- 虽然搜索的收敛速度得以加快，但蚁群在最优路径的搜索过程中随机性减弱，易于陷入局部最优.

蚁群算法的参数选择

信息素挥发度 $1-\rho$

- 当要处理的问题规模比较大时，会使那些从来未被搜索到的路径 (可行解) 上的信息量减小到接近于 0，因而降低了算法的全局搜索能力；
- 而且当 ρ 过大时，以前搜索过的路径被再次选择的可能性过大，也会影响到算法的随机性能和全局搜索能力；
- 反之，通过减小信息素挥发度 ρ 虽然可以提高算法的随机性能和全局搜索能力，但又会使算法的收敛速度降低.

内容组织

1. 粒子群算法

2. 量子粒子群算法

3. 蚁群算法

4. 自适应蚁群算法

自适应蚁群算法

自适应蚁群算法

- 自适应蚁群算法能根据判断搜索结果是否陷入局部收敛从而采用一种新的信息素更新策略
- 自适应动态调整陷入局部收敛的蚂蚁所经过路径上的信息素 ρ 和信息素强度 Q，使得算法能更快的跳出局部收敛，防止“早熟”
- 同时对所有路径上的信息素取值限定范围
- 有利于算法的全局搜索

自适应蚁群算法

自适应蚁群算法

1. 状态转移规则

- 第 k 只蚂蚁由节点 r 转移到节点 s 的概率按下式计算，所得的概率记为 P_{rs}^{k}：

$$P_{rs}^{k}=\begin{cases}\dfrac{\tau_{ij}^{\alpha}\cdot\eta_{ij}^{\beta}}{\sum_{i\in p}\tau_{ij}^{\alpha}\cdot\eta_{ij}^{\beta}} & if\ p_{ij}<p_{i\max}\\ 0, & others\end{cases}$$

- 其中：
  - τ_{ij} 表示节点 i 到节点 j 的信息素
  - η_{ij} 表示节点 i 到节点 j 的可见度
  - α,β 分别表示信息素和可见度的偏重系数

自适应蚁群算法

自适应蚁群算法

2. 判断是否发生局部收敛

- 各代所有蚂蚁爬行完毕后对所搜索到的最优解进行判断，看是否陷入局部收敛，判断方法如下：
  - 当连续几代最优蚂蚁搜索得到的路径相同时，算法即陷入了局部收敛
  - 当连续几代的最优蚂蚁爬行路径总长度相同时算法陷入了局部最优

自适应蚁群算法

自适应蚁群算法

3. 自适应信息素挥发系数 ρ

- 当算法陷入局部收敛时，ρ 不再为常数，而是随着连续最优解相同的代数的增大而增大，表达式如下：

$$\rho = \begin{cases} \rho_0, & n \leq n_0 + 1 \\ 1 - \frac{1-\rho_0}{n-n_0} & n > n_0 + 1 \end{cases}$$

- 其中：
  - ρ_0 为初始挥发度，n 为各代最优解连续相等的次数
  - n_0 为大于 1 的整数，当 $n > n_0 + 1$ 时 ρ 开始减小 n 越大 ρ 越小
  - 算法具体实现时，ρ_0, n_0 可以根据需要进行调节

自适应蚁群算法

自适应蚁群算法

4. 自适应信息素强度 $Q(n)$

- 当算法陷入局部收敛时采用时变函数 $Q(n)$ 来代替基本蚁群算法中调整信息素 $\Delta\tau_{ij}^k = \frac{Q}{L_k}$ 中为常数项的信息素强度 Q，即选择 $\Delta\tau_{ij}^k = \frac{Q}{L_k}$，随着人工蚂蚁搜索过程动态的调整，$Q(n)$ 如下所示：

$$Q(n) = \begin{cases} Q_0, & n \le n_0 \\ -Q_0 \times (n - n_0) & n > n_0 \end{cases}$$

- 其中：
  - Q_0 为初始信息素强度，可以根据需要调整.

自适应蚁群算法

自适应蚁群算法

5. 改进的信息素更新策略

- 信息素更新策略存在多种方式：
  - 走过的全部路径上的信息素进行更新，导致算法获得的结果振荡，不易收敛
  - 更新搜索到最优边上的信息素，则进一步加强了蚁群算法的正反馈作用，导致搜索过程迅速陷入局部最优解

自适应蚁群算法

5. 改进的信息素更新策略

记 l 为每代最优解对应的蚂蚁，蚂蚁总数为 m

(1) 当算法未陷入局部最优时，采用全局更新和局部更新结合的策略，其中 ρ 和 Q 均为初始值：

Step1: 全局更新，计算所有蚂蚁经过路径上的信息素增量：

$$\Delta\tau_{ij}=\sum_{k=1}^{m}\Delta\tau_{ij}^{k}\quad \Delta\tau_{ij}^{k}=\frac{Q(n)}{L_k}\quad Q(n)=Q_0\quad k=1,2,\cdots,m$$

Step2: 局部更新，如果该代最优解为历代最优解，则调整蚂蚁 l 经过路径上的信息素增量：

$$\Delta\tau_{ij(new)}^{(l)}=\Delta\tau_{ij(old)}^{(l)}+\Delta\tau_{ij}^{l}\quad \Delta\tau_{ij}^{l}=\frac{Q(n)}{L_l}\quad Q(n)=Q_0$$

Step3: 更新所有蚂蚁经过路径上的信息素：

$$\tau_{ij(new)}=(1-\rho)\tau_{ij(old)}+\Delta\tau_{ij}\quad \rho=\rho_0$$

自适应蚁群算法

5. 改进的信息素更新策略

记 l 为每代最优解对应的蚂蚁，蚂蚁总数为 m

(2) 当算法陷入局部最优时，仅采用全局更新策略：

Step1: 计算除最优蚂蚁 l 外所有其他蚂蚁经过路径上的信息素增量：

$$\Delta\tau_{ij}=\sum_{k=1}^{m}\Delta\tau_{ij}^{k}\quad \Delta\tau_{ij}^{k}=\frac{Q(n)}{L_k}\quad Q(n)=Q_0\quad k=1,2,\cdots,m$$

Step2: 计算蚂蚁 l 经过的路径上的信息素增量：

$$\Delta\tau_{ij}^{(l)}=\Delta\tau_{ij}^{l}\quad \Delta\tau_{ij}^{l}=\frac{Q(n)}{L_l}\quad Q(n)=-Q_0\times(n-n_0)$$

Step3: 更新除蚂蚁 l 外所有其他蚂蚁经过路径上的信息素：

$$\tau_{ij}=(1-\rho)\tau_{ij}+\Delta\tau_{ij}\quad \rho=\rho_0$$

Step4: 更新蚂蚁 l 经过路径上的信息素：

$$\tau_{ij}^{(l)}=(1-\rho)\tau_{ij}^{(l)}+\Delta\tau_{ij}^{(l)}\quad \rho=1-\frac{1-\rho_0}{n-n_0}$$

自适应蚁群算法

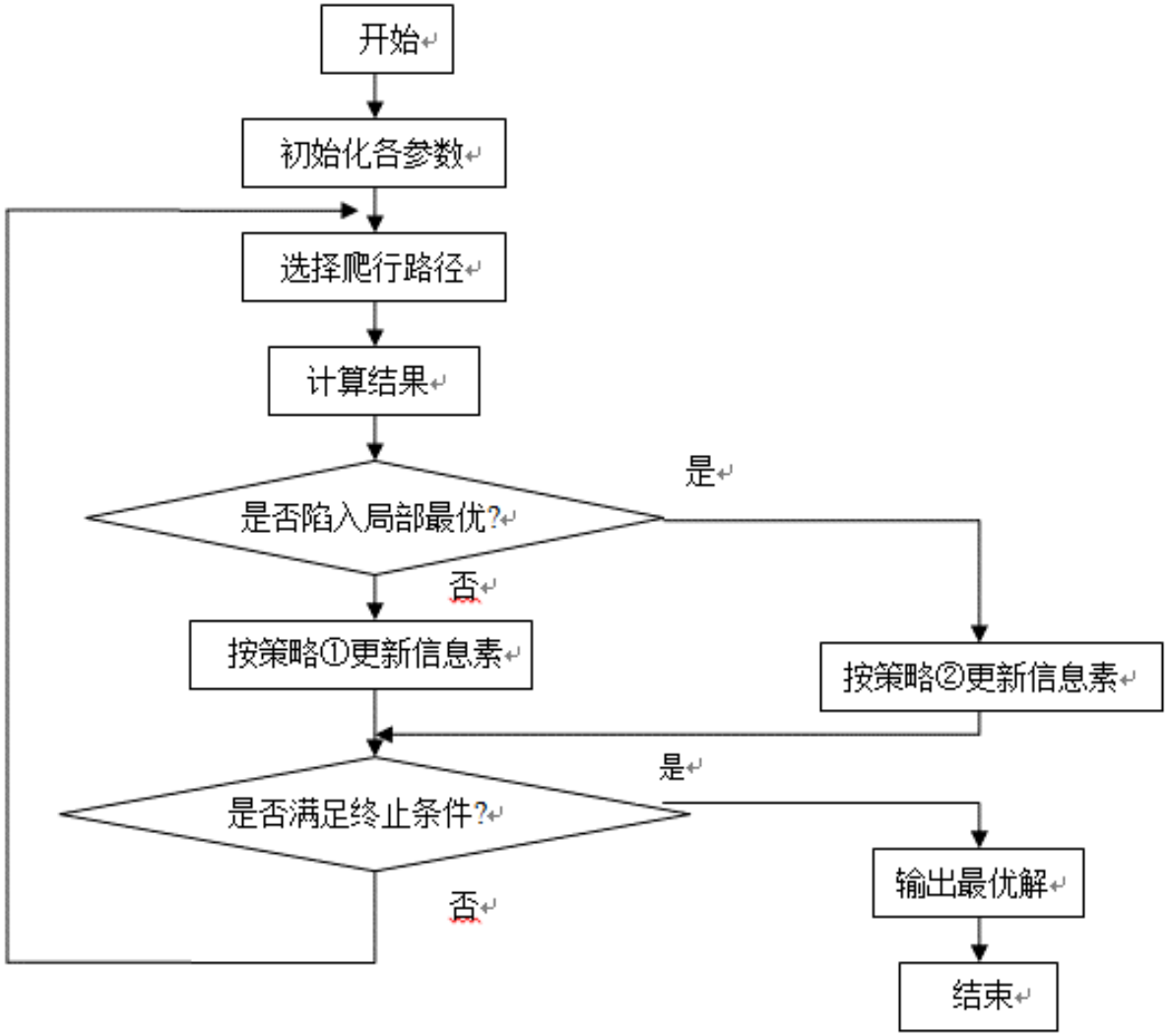


图: 自适应蚁群算法流程图

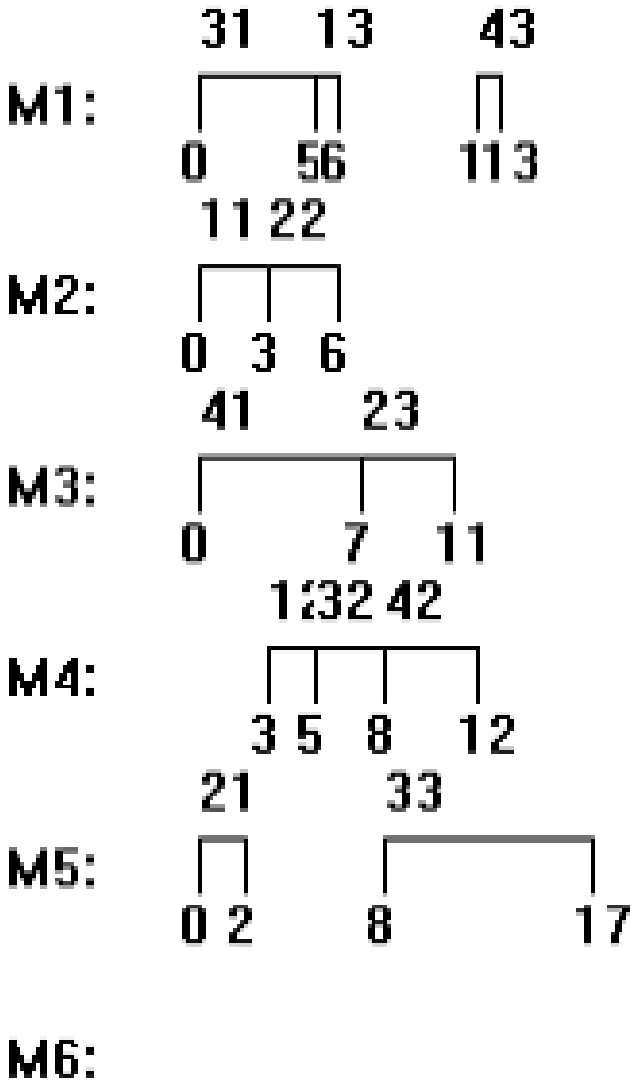
蚁群算法的应用

柔性作业车间调度问题：某加工系统有 6 台机床，要加工 4 个工件，每个工件有 3 道工序，如表所示.

工序选择		加工机床及加工时间					
		1	2	3	4	5	6
J_1	p_{11}	2	3	4			
	p_{12}		3		2	4	
	p_{13}	1	4	5			
J_2	p_{21}	3		5		2	
	p_{22}	4	3		6		
	p_{23}			4		7	11
J_3	p_{31}	5	6				
	p_{32}		4		3	5	
	p_{33}			13		9	12
J_4	p_{41}	9		7	9		
	p_{42}		6		4		5
	p_{43}	1		3			3

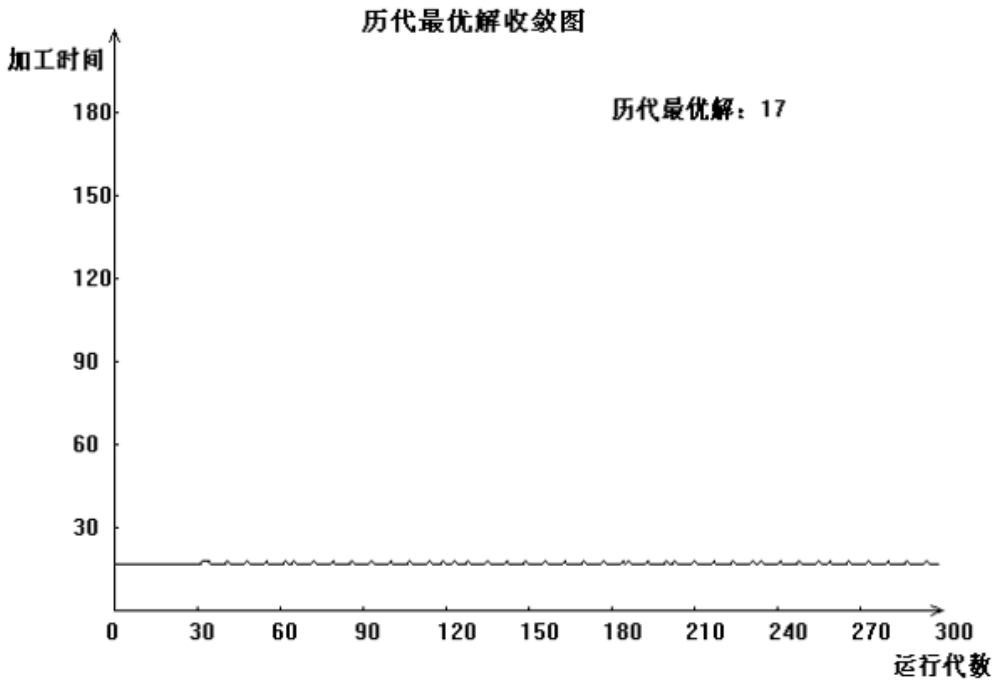
蚁群算法的应用

可以看出机器 6 并没有加工任何工件．分析其原因为它虽然可以加工工序 $p_{23}, p_{33}, p_{42}, p_{43}$ 但从表可知机器 6 的加工时间大于其他可加工机器，特别是 p_{23}, p_{33} 的加工时间，因此机器 6 并未分到任何加工任务．



蚁群算法的应用

由图知，算法在大约 30 代以前就收敛到最优解，且各代最优解相差不大，可见算法较为稳定.



Question & Answer