

第7章 群集智能算法

- 7.1 群集智能算法的研究背景
- 7.2 群集智能的基本算法介绍
- 7.3 集智系统介绍
- 7.4 群集智能的优缺点

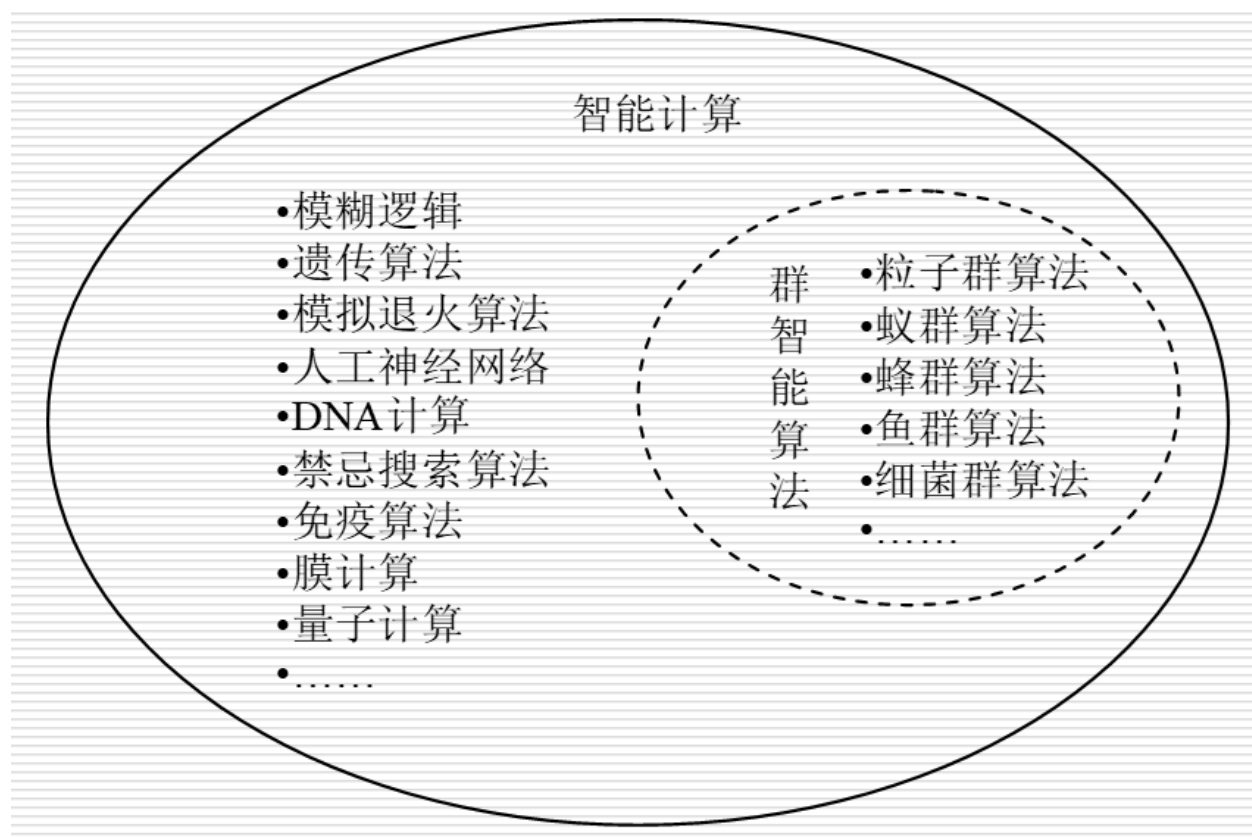
7.1 群集智能算法的研究背景

起源于对人工生命的研究

“人工生命”是用来研究具有某些生命基本特征的人工系统。包括两方面的内容：

1. 研究如何利用计算技术研究生物现象
2. 研究如何利用生物技术研究计算问题

7.1 群集智能算法的研究背景



7.1 群集智能算法的研究背景

群集智能算法 (swarm algorithm, SI) :受动物群体智能启发的算法

群体智能 : 有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、.....

7.2 群集智能的基本算法介绍

7.2.1 蚁群算法

蚁群算法是对蚂蚁群落食物采集过程的模拟，已经成功运用在很多离散优化问题上。

7.2.2 flock算法

flock算法后者也是起源对简单社会系统的模拟，最初设想是模拟鸟群觅食的过程。

7.2.1 蚁群算法

蚁群优化算法起源

20世纪50年代中期创立了仿生学，人们从生物进化的机理中受到启发。提出了许多用以解决复杂优化问题的新方法，如进化规划、进化策略、遗传算法等，这些算法成功地解决了一些实际问题。

- 1991年 意大利米兰理学院 M. Dorigo 提出Ant System, 用于求解TSP等组合优化问题。
- 1995年 Gramdardella和Dorigo提出Ant-Q算法，建立了AS和Q-learning的联系。
- 1996年 二人又提出Ant Colony System
- 1997年 有人提出Max-Min Ant System
- 1999年 Dorigo等人把先前各种算法归结为Ant Colony Optimization meta-heuristic的统一框架下，给出抽象而规范的算法描述。
- 目前，被较广泛的应用



Dorigo

7.2.1 蚁群算法

蚁群优化算法的应用领域

- 蚁群优化算法自1991年由Dorigo提出并应用于TSP问题以来，已经发展了20年。
- 具有鲁棒性强、全局搜索、并行分布式计算、易与其他问题结合等优点
- 应用领域不断扩张，如车间调度问题、车辆路径问题、分配问题、子集问题、网络路由问题、蛋白质折叠问题、数据挖掘、图像识别、系统辨识等。
- 这些问题大都是NP难的组合优化问题，用传统算法难以求解或者无法求解，各种蚁群算法及其改进版本的出现为这些难题提出了有效而高效的手段。

7.2.1 蚁群算法

- 蚂蚁属于群居昆虫

单个蚂蚁的行为极其简单，但由这样的单个简答的个体所组成的蚁群群体却表现出极其复杂的行为，能够完成复杂的任务。

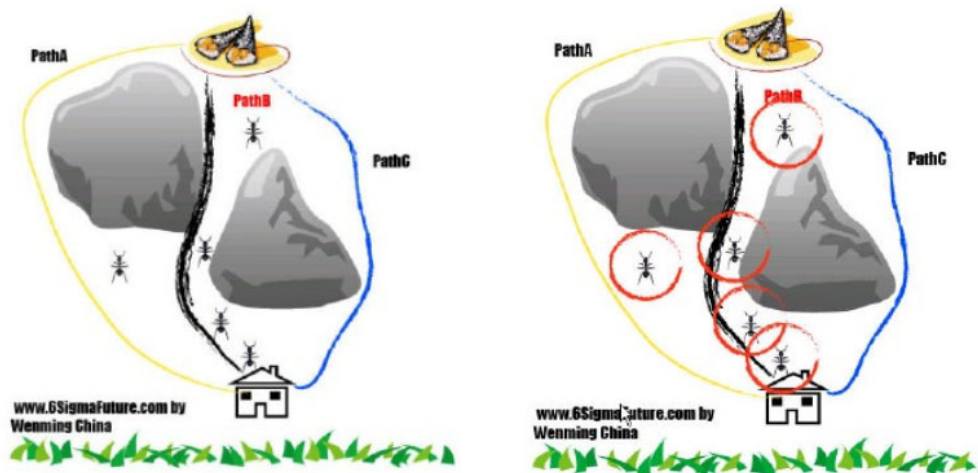


7.2.1 蚁群算法

● 蚂蚁觅食

蚂蚁没有发育完全的视觉感知系统，甚至很多种类完全没有视觉，他们在寻找食物的过程中是如何选择路径的呢？

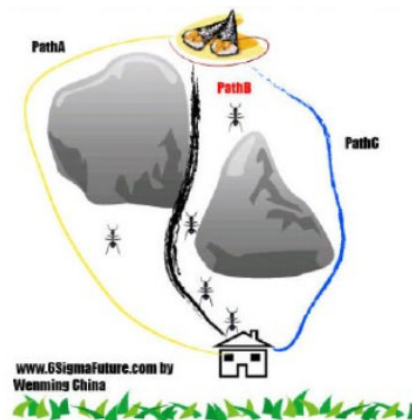
蚂蚁往往像军队般有纪律、有秩序的搬运食物，他们是通过什么方式进行群体间的交流协作呢？大部分的蚂蚁都是按照图中的最近的路线走，小小的蚂蚁是如何有这么高的智能的呢？



7.2.1 蚁群算法

蚁群算法原理

- 蚂蚁是如何完成这些复杂的任务的呢？人们经过大量研究发现，蚂蚁个体之前通过一种称之为信息素（pheromone）的物质进行信息传递。从而能相互协作，完成复杂的任务。
- 蚁群之所以表现出复杂有序的行为，个体之间的信息交流与相互协作起着重要的作用。



7.2.1 蚁群算法

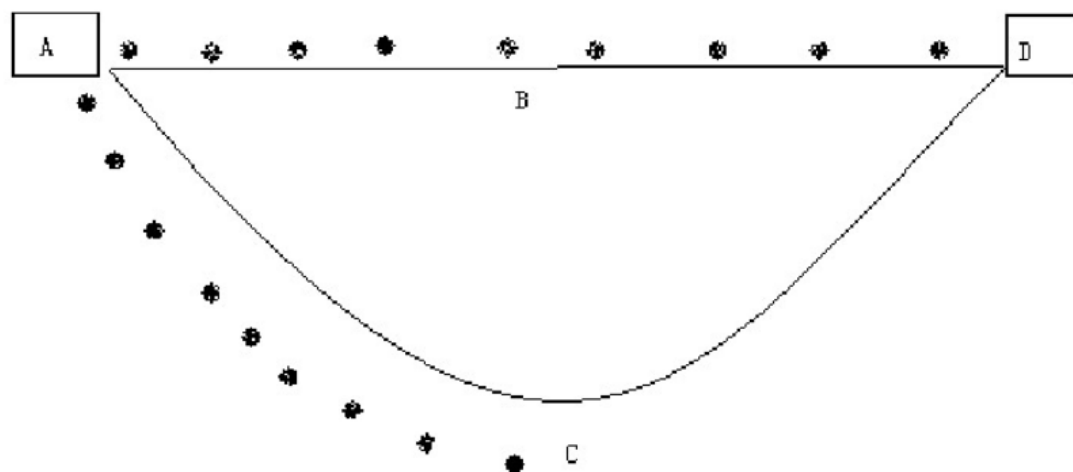
基本思想

- 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。



7.2.1 蚁群算法

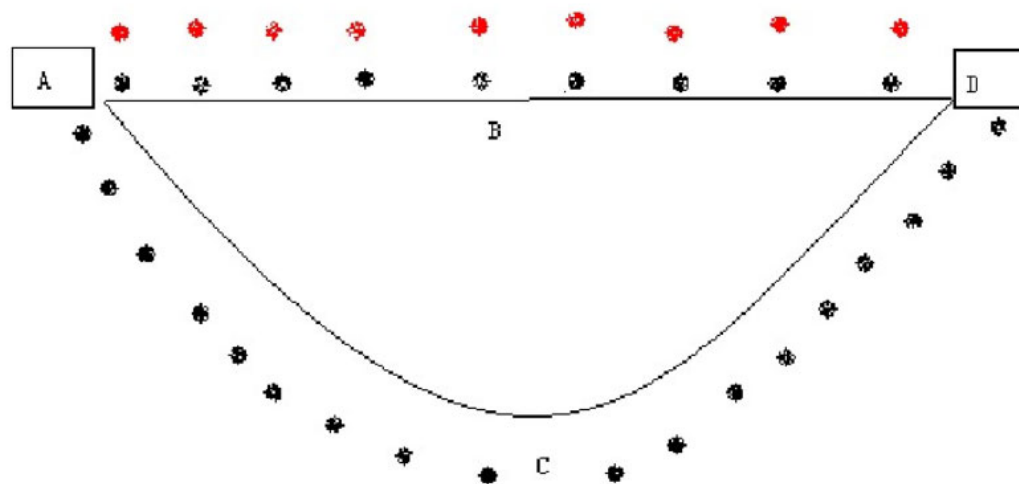
简化的蚂蚁寻食过程



蚂蚁从A点出发，速度相同，食物在D点，可能随机选择路线ABD或ACD。假设初始时每条分配路线一只蚂蚁，每个时间单位行走一步，本图为经过9个时间单位时的情形：走ABD的蚂蚁到达终点，而走ACD的蚂蚁刚好走到C点，为一半路程。

7.2.1 蚁群算法

简化的蚂蚁寻食过程



本图为从开始算起，经过18个时间单位时的情形：走ABD的蚂蚁到达终点后得到食物又返回了起点A，而走ACD的蚂蚁刚好走到D点。

7.2.1 蚁群算法

简化的蚂蚁寻食过程

假设蚂蚁每经过一处所留下的信息素为一个单位，则经过36个时间单位后，所有开始一起出发的蚂蚁都经过不同路径从D点取得了食物，此时ABD的路线往返了2趟，每一处的信息素为4个单位，而ACD的路线往返了一趟，每一处的信息素为2个单位，其比值为2: 1。

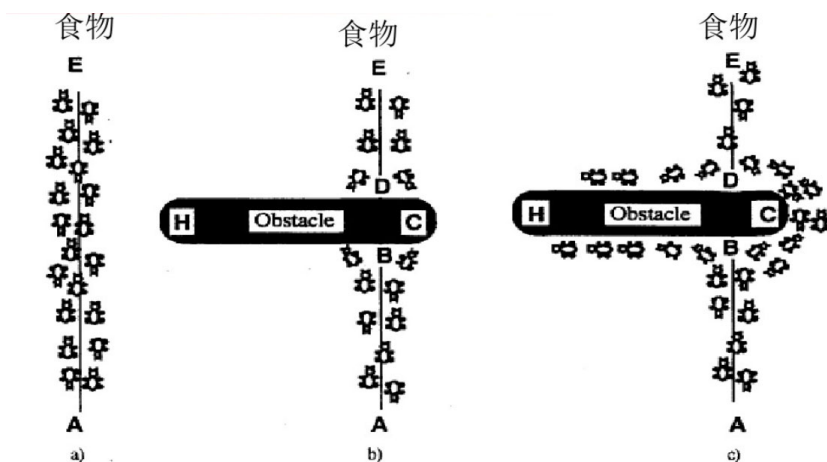
寻找食物的过程继续进行，则按信息素的指导，蚁群在ABD路线上增派一只蚂蚁（共2只），而ACD路线上仍然为一只蚂蚁。再经过36个时间单位后，两条线路上的信息素单位积累为12和4，比值为3: 1。

若按以上规则继续，蚁群在ABD路线上再增派一只蚂蚁（共3只），而ACD路线上仍然为一只蚂蚁。再经过36个时间单位后，两条线路上的信息素单位积累为24和6，比值为4: 1。

若继续进行，则按信息素的指导，最终所有的蚂蚁会放弃ACD路线，而都选择ABD路线。这也就是前面所提到的正反馈效应。

7.2.1 蚁群算法

- (1) **环境**：有障碍物、有其他蚂蚁、有信息素。
- (2) **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。
- (3) **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。
- (4) **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。
- (5) **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。



7.2.1 蚁群算法

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。

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。

旅行商问题

阐明

蚁群系统模型

7.2.1 蚁群算法

自然蚁群与人工蚁群算法

- 基于以上蚁群寻找食物时的最优路径选择问题，可以构造人工蚁群，来解决TSP问题。人工蚁群中把具有简单功能的工作单元看作蚂蚁。

人工蚁群和自然蚁群的相似之处

- 都是优先选择信息素浓度大的路径。
较短路径的信息素浓度高，所以能够最终被所有蚂蚁选择，也就是最终的优化结果。

人工蚁群和自然蚁群的区别

- 人工蚁群有一定的记忆能力，能够记忆已经访问过的节点；
- 人工蚁群选择下一条路径的时候是按一定算法规律有意识地寻找最短路径，而不是盲目的。例如在TSP问题中，可以预先知道当前城市到下一个目的地的距离。

自然蚁群与人工蚁群算法

蚁群觅食

蚁群优化算法

蚁群

搜索空间的一组有效解（表现为种群规模 N ）

觅食空间

问题的搜索空间（表现为维数 D ）

信息素

信息素浓度变量

蚁巢到食物的一条路径

一个有效解

找到的最短路径

问题的最优解

7.2.1 蚁群算法

TSP问题表示为一个N个城市的有向图 $G = (N, A)$,

其中 $N = \{1, 2, \dots, n\}$ $A = \{(i, j) \mid i, j \in N\}$

城市之间距离 $(d_{ij})_{n \times n}$

目标函数为 $f(w) = \sum_{l=1}^n d_{i_{l-1} i_l}$

其中 $w = (i_1, i_2, \dots, i_n)$ 为城市1, 2, ..., n的

一个排列, $i_{n+1} = i_1$ 。

7.2.1 蚁群算法

- TSP问题的人工蚁群算法中，假设 m 只蚂蚁在图的相邻结点间移动，从而协作异步地得到问题的解。每只蚂蚁的一步转移概率由图中的每条边上的两类参数决定：1.信息素值也称信息素痕迹。2.可见度，即先验值。
- 信息素的值更新方式有两种，一是挥发，也就是所有路径上的信息素以一定的比率进行减少，模拟自然蚁群的信息素随时间挥发的过程；二是增强，给评价“好”（有蚂蚁走过）的边增加信息素。
- 蚂蚁向下一个目标的运动是通过一个随机原则来实现的，也就是运用当前所在节点存储的信息，计算出下一步可达节点的概率，并按此概率实现一步移动，逐此往复，越来越接近最优解。

7.2.1 蚁群算法

蚁群优化算法的基本流程

AS算法对TSP的求解有两大步骤：路径构建和信息素更新

1. 路径构建

每个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达到的城市。

7.2.1 蚁群算法

m 是蚁群中蚂蚁的数量

$d_{xy}(x, y = 1, \dots, n)$ 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, $m = \sum_{x=1}^n b_x(t)$

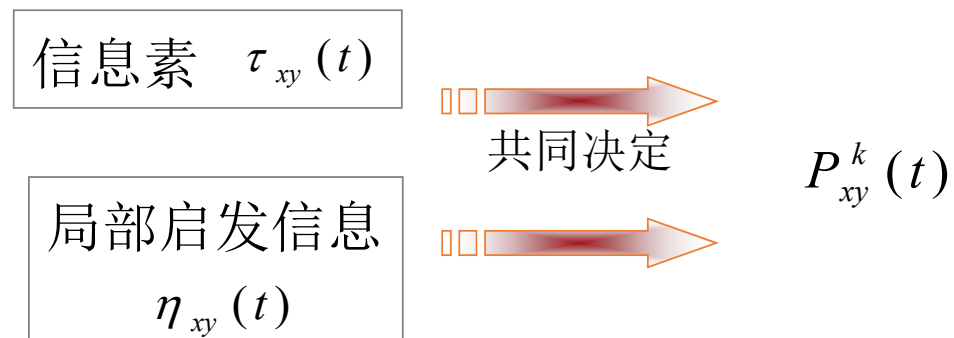
$\tau_{xy}(t)$ 表示 t 时刻在 xy 连线上残留的信息素, 初始时刻, 各条路径上的信息素相等即

$$\tau_{xy}(0) = C(const)$$

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。

7.2.1 蚁群算法

$P_{xy}^k(t)$ 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。



7.2.1 蚁群算法

基本蚁群算法模型

$P_{xy}^k(t)$ 表示如下:

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

其中:

$allowed_k(x)$ 为尚未访问过的节点集合

α, β 为两常数, 分别是信息素和能见度的加权值。

7.2.1 蚁群算法

基本蚁群算法模型

$P_{xy}^k(t)$ 表示如下:

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

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。

7.2.1 蚁群算法

蚁群优化算法的基本流程

AS算法对TSP的求解有两大步骤：路径构建和信息素更新

2. 信息素更新

初始化信息浓度 $\tau_{ij} = C, \quad \forall i, j$

如果C太小，算法容易早熟，蚂蚁会很快全部集中到一条局部最优的路径上。

反之，如果C太大，信息素对搜索方向的指导作用太低，也会影响算法性能。

AS中： $C = m / C^{nn}$

7.2.1 蚁群算法

信息素更新的作用

- 1.信息素挥发（evaporation）信息素痕迹的挥发过程是每个连接上的信息素痕迹的浓度自动逐渐减弱的过程，这个挥发过程主要用于避免算法过快地向局部最优区域集中，有助于搜索区域的扩展。
- 2.信息素增强（reinforcement）增强过程是蚁群优化算法中可选的部分，称为离线更新方式（还有在线更新方式）。这种方式可以实现由单个蚂蚁无法实现的集中行动。基本蚁群算法的离线更新方式是在蚁群中的 m 只蚂蚁全部完成 n 城市的访问后，统一对残留信息进行更新处理。

7.2.1 蚁群算法

为了模拟蚂蚁在较短路径上留下更多的信息素，当所有蚂蚁到达终点时，必须把各路径的信息素浓度重新更新一次，信息素的更新也分为两个步骤：

首先，每一轮过后，问题空间中的所有路径上的信息素都会发生蒸发
然后，所有的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

$$\tau_{ij}(t) = (1 - \rho)\tau_{ij} + \sum_{k=1}^m \Delta\tau_{ij}^k$$

M为蚂蚁个数， $0 < \rho \leq 1$ 为信息素的蒸发率，在AS中通常设置为0.5， $\Delta\tau_{ij}^k$ 为第K只蚂蚁在路径i到j所留下来的信息素

$\Delta\tau_{xy}(t)$ 的三种不同模型

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：

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

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

$\Delta\tau_{xy}^k(t)$ 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

$\Delta\tau_{xy}(t)$ 的三种不同模型

2. 蚂蚁数量系统 (Ant-quantity System)

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


3. 蚂蚁密度系统 (Ant-density System)

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

7.2.1 蚁群算法

三种模型比较

效果最好，通常作为蚁群优化算法的基本模型。



蚂蚁圈系统	利用的是全局信息 Q/L_k ，即蚂蚁完成一个循环后，更新所有路径上的信息。
蚂蚁数量系统	利用的是局部信息 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。

7.2.1 蚁群算法

全局信息更新方法的优点：

保证了残留信息素不至于无限累积；

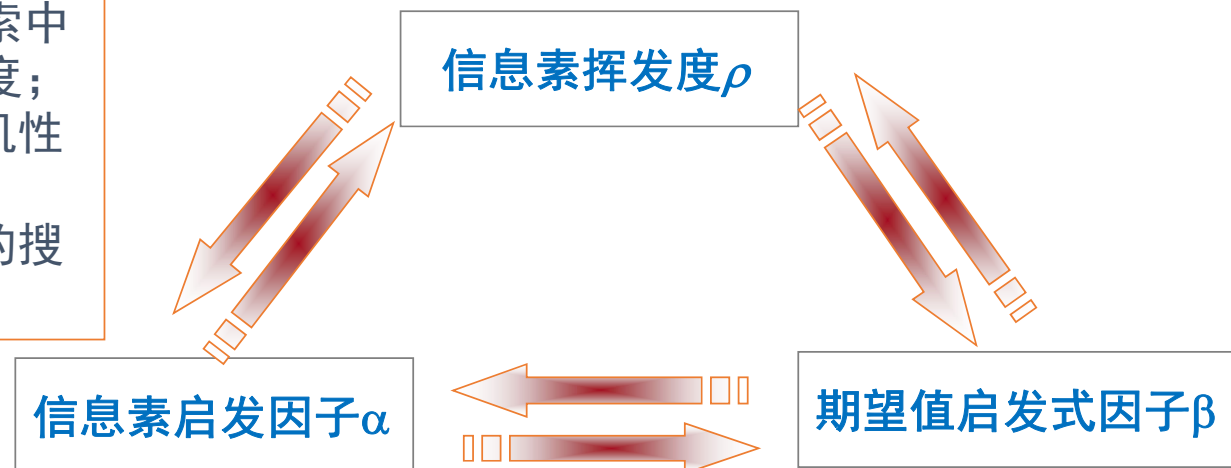
- 如果路径没有被选中，那么上面的残留信息素会随时间的推移而逐渐减弱，这使算法能“忘记”不好的路径；
- 即使路径经常被访问也不至于因为 $\Delta\tau_{xy}^k(t)$ 的累积，而产生 $\Delta\tau_{xy}^k(t) \gg \eta_{xy}(t)$ 使期望值的作用无法体现；
- 充分体现了算法中全局范围内较短路径(较好解)的生存能力；
- 加强了信息正反馈性能；
- 提高了系统搜索收敛的速度。

7.2.1 蚁群算法

蚁群算法的参数选择

- 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，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。

- 当要处理的问题规模比较大时，降低了算法的全局搜索能力；
- 而且当 ρ 过小时，以前搜索过的路径被再次选择的可能性过大；
- 反之，又会使算法的收敛速度降低。



- 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。

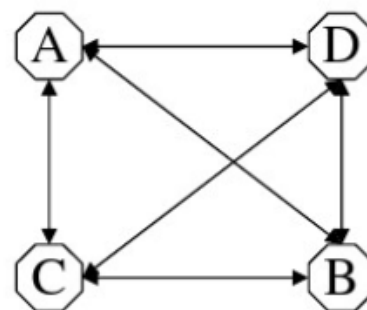
TSP问题的蚁群算法伪代码

```
Procedure AS
  for each edge
    set initial pheromone value  $t_0$ 
  end for
  while not stop
    for each ant  $k$ 
      randomly choose an initial city
      for  $i=1$  to  $n$ 
        choose next city  $j$  with probability
      end for
    end for
    compute the length  $C_k$  of the tour constructed by the  $k$ th ant
    for each edge
      update the pheromone value
    end for
  end while
  print result
end procedure
```

应用举例

四个城市的**TSP**问题，距离矩阵和城市图示如下：

$$D = (d_{ij}) = \begin{pmatrix} 0 & 3 & 1 & 2 \\ 3 & 0 & 5 & 4 \\ 1 & 5 & 0 & 2 \\ 2 & 4 & 2 & 0 \end{pmatrix}$$



应用举例

假设共 $m=3$ 只蚂蚁，参数 $\alpha=1$ ， $\beta=2$ ， $\rho=0.5$

步骤1 初始化 首先使用贪婪算法得到路径的（ACDBA），则 $C_{mn}=1+2+4+3$ 求得 $\tau_0=m/C_{mn}=0.3$

$$\tau(0) = (\tau_{ij}(0)) = \begin{pmatrix} 0 & 0.3 & 0.3 & 0.3 \\ 0.3 & 0 & 0.3 & 0.3 \\ 0.3 & 0.3 & 0 & 0.3 \\ 0.3 & 0.3 & 0.3 & 0 \end{pmatrix}$$

步骤2 为每个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。

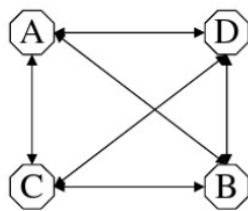
应用举例

步骤3.1 为每个蚂蚁选择下一访问城市，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访问各个城市的概率

四个城市的TSP问题，距离矩阵和城市图示如下：

$$D=(d_{ij})=\begin{pmatrix} 0 & 3 & 1 & 2 \\ 3 & 0 & 5 & 4 \\ 1 & 5 & 0 & 2 \\ 2 & 4 & 2 & 0 \end{pmatrix}$$



$$A \Rightarrow \begin{cases} B: \tau_{AB}^a \times \eta_{AB}^\beta = 0.3^1 + (1/3)^2 = 0.033 \\ C: \tau_{AC}^a \times \eta_{AC}^\beta = 0.3^1 + (1/1)^2 = 0.300 \\ D: \tau_{AD}^a \times \eta_{AD}^\beta = 0.3^1 + (1/2)^2 = 0.075 \end{cases}$$
$$p(B) = 0.033 / (0.033 + 0.3 + 0.075) = 0.081$$
$$p(C) = 0.3 / (0.033 + 0.3 + 0.075) = 0.74$$
$$p(D) = 0.075 / (0.033 + 0.3 + 0.075) = 0.18$$

用轮盘赌法选择下一个访问城市，假设产生的随机数 $q=0.05$ ，则蚂蚁1会选择城市B

应用举例

步骤3.2 为每个蚂蚁选择下一访问城市，仅以蚂蚁1为例

当前城市 $i=B$ ，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访问C,D城市的概率

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

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

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

用轮盘赌法选择下一个访问城市，假设产生的随机数 $q=0.67$ ，则蚂蚁1会选择城市D。

同样，假设蚂蚁2选择城市C，蚂蚁3选择城市D。

应用举例

此时，所以蚂蚁的路径都已经构造完毕

蚂蚁1: $A \rightarrow B \rightarrow D \rightarrow C \rightarrow A$

蚂蚁2: $B \rightarrow D \rightarrow C \rightarrow A \rightarrow B$

蚂蚁3: $D \rightarrow A \rightarrow C \rightarrow B \rightarrow D$

应用举例

蚂蚁1: A→B→D→C→A

蚂蚁2: B→D→C→A→B

蚂蚁3: D→A→C→B→D

步骤4 信息素更新

计算每只蚂蚁构建的路径长度: C1=3+4+2+1=10; C2=4+2+1+3=10; C3=2+1+5+4=12.

更新每条边上的信息素:

$$\tau_{AB} = (1 - \rho) \times \tau_{AB} + \sum_{k=1}^3 \Delta\tau_{AB}^k = 0.5 \times 0.3 + (1/10 + 1/10) = 0.35$$

$$\tau_{AC} = (1 - \rho) \times \tau_{AC} + \sum_{k=1}^3 \Delta\tau_{AC}^k = 0.5 \times 0.3 + (1/12) = 0.16$$

步骤5 如果满足结束条件, 则输出全局最优结果并结束程序, 否则, 则转向步骤2继续执行。

蚁群优化算法研究现状

一般蚁群算法的框架和基本蚁群算法大致相同，有三个组成部分：

- 蚁群的活动
- 信息素的挥发
- 信息素的增强

主要体现在转移概率公式和信息素更新公式的不同

蚁群优化算法研究现状

1. 蚁群优化算法的发展

- 90年代Dorigo最早提出了蚁群优化算法-蚂蚁系统（Ant System, AS），并将其应用于解决计算机算法学中经典的旅行商问题（TSP）。
- 从蚂蚁系统开始，基本的蚁群算法得到了不断的发展和完善，并在TSP以及许多实际优化问题求解中进一步得到了验证；
- 这些AS改进版本的一个共同点就是增强了蚂蚁搜索过程中对最优解的探索能力，它们之间的差异仅在于搜索控制策略方面；
- 取得最好结果的ACO是通过引入局部搜索算法实现的，实际上是一些结合了标准局域搜索算法的混合型概率搜索算法，有利于提高蚁群的各级系统在优化问题中的求解质量。

蚁群优化算法研究现状

2 蚂蚁系统的三种基本版本

Ant-density, Ant-quantity和Ant cycle

通过与其他各种通用的启发式搜索算法相比，在不大于75城市的TSP中，这三种基本算法的求解能力还是比较理想的，但是当问题规模扩展时，AS解题能力大幅度下降。

3. 精英策略的蚂蚁系统 (Elitist Ant System, EAS)
4. 基于排列的蚂蚁系统 (Rank-based AS, AS Rank)
5. 最大最小蚂蚁系统 (MAX-MIN Ant System, MMAS)
6. 蚁群系统 (Ant Colony System)

7.2.2 flock算法

Flock算法是由Craig Reynolds于1987年在一篇为SIGGRAPH所写的论文“Flocks, Herds, and Schools: A Distributed Behavioral Model”中首次提出的一种集智技术。这种技术有3个简单的规则，当它们组合在一起时，为自治主体（boid）群给出了一个类似于鸟群、鱼群或蜂群的群体行为的逼真表现形式。这些被Reynolds称之为定向行为（Steering Behaviors）。

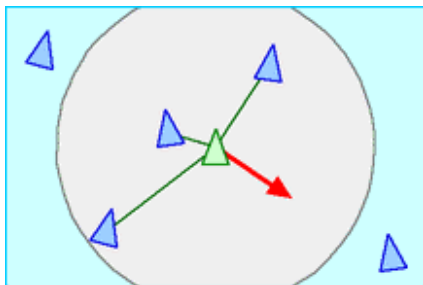
7.2.2 flock算法

定向行为的规则：

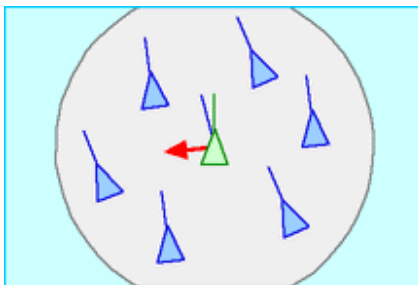
分离：定向时要避免与本地flock同伴拥挤

列队：驶向本地flock同伴的平均航向

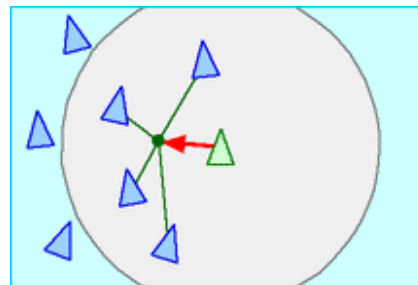
聚合：定向时朝着本地flock同伴的平均位置移动



分离



列队



聚合

7.2.2 flock算法

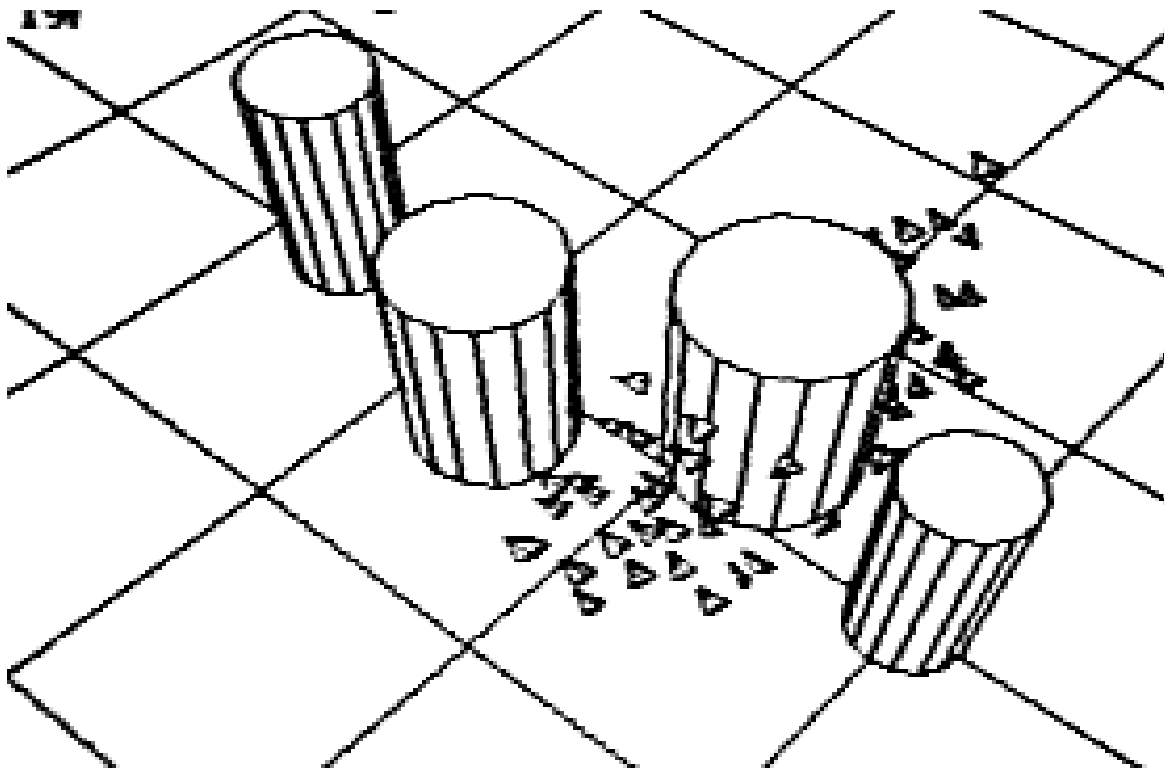
- **分离规则**给了一个主体试图与其它邻近的主体保持一定的距离的能力。确保主体之间以一个“看似自然”的接近度，模拟真实世界中的群体，以避免主体拥挤在一起。
- **队列规则**为一个主体提供了与其他邻近主体列队的能力（即与其他邻近主体航向或速度相同）。与分离类似，本文将队列说明为：通过每一个flock成员观察邻近同伴，然后调整它的航向和速度以与其邻近同伴的平均航向和速度相匹配。

7.2.2 flock算法

- 聚合规则给了一个主体与其他邻近主体“聚合（group）”的能力，从而模拟自然界的类似行为。
- Reynolds在稍后的实现和论文中又增加了有时被称作flocking“第四规则”的规则。

7.2.2 flock算法

躲避：使避免撞上局部区域的障碍和敌人



7.2.2 flock算法

- 躲避规则的作用是为主体提供了使它绕过障碍和避免碰撞的能力。这种控制行为是这样完成的：通过赋予每个主体“向前看”一段距离的能力，决定与一些对象的碰撞是否可能，然后调整航向以避免碰撞。
- Flock技术通过这四个简单的规则最终模拟出逼真的群体行为，更有意思的是这种移动算法本身是无状态的：在移动更新中，不记录任何信息。

7.2.2 flock算法

在每次更新循环中，每只boid都将重新评估其环境。这样不但降低了内存需求，同时让物群能够对不断变化的环境状况做出实时的反应。因此，物群将具备突发行为的（Emergent Behavior）特性，即物群中的所有成员都对要前往何方一无所知，但作为一个整体行动，避开障碍物和天敌，并保持若即若离。

7.3 集智系统介绍

7.3.1 人工鱼

7.3.2 Terrarium世界

7.3.1 人工鱼

人工鱼群体是一种典型的多智能主体（Multiple Intelligent Agent）的分布式人工智能系统（distributive artificial intelligent system）。

中国青年学者涂晓媛研究开发的新一代计算机动画“人工鱼”被学术界称之为“晓媛鱼”（Xiaoyuan's fish），她发表的论文“人工动物的计算机动画”（artificial animals for computer animation: biomechanics , locomotion, perception and behavior），在1996年获国际计算学会ACM最佳博士论文奖。

7.2.2 flock算法

- 涂晓媛研究开发的“人工鱼”构成了栖息在虚拟海底世界中人工鱼群的社会，其中，每条“人工鱼”都是一个自主的智能体（autonomous intelligent agent），都可以独立地活动，也可以相互交往。每条鱼都表现出某些人工智能，如：自激发（self-animating）、自学习（self-learning）、自适应（self-adapting）等智能特性.

7.2.2 flock算法

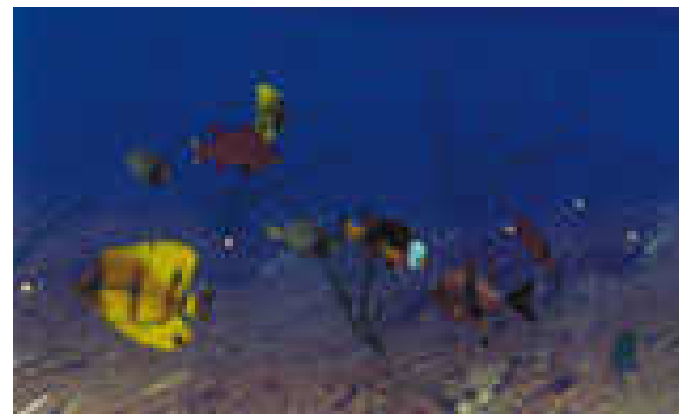
会产生相应的智能行为，如：因饥饿而激发寻食、进食行为；能吸取其他鱼被鱼钩钓住的教训，而不去吞食有钩的鱼饵；能适应有鲨鱼的社会环境，逃避被捕食的危险等。人工鱼群的社会具有某些自组织（self-organizing）能力和智能集群行为，如：人工鱼群体在漫游中遇到障碍物等，会识别障碍改变队形，绕过障碍后，又重组队列，继续前进。

7.3.1 人工鱼

•1. 人工鱼与动画鱼

区别：

- “人工鱼”具有“人工生命”和自然鱼的某些生命特征。
- 在计算机动画中，创作者需要在动画设计和程序编制中确定动画鱼的所有动作的细节和全部动作过程。然而，人工鱼的创作者并不去设计和规定每条鱼的动作和行为的细节，也不能预知人工鱼群中可能发生各种具体动作和实际行为。



7.3.1 人工鱼

2. “人工鱼”与“人工生命”

涂晓媛所说的“人工生命”（artificial life）是指具有“自然生命”特性和功能的人造系统，或者说是“人造活体”。

人工生命的研究方法和技术途径，可以分：

- 生命科学途径
- 工程技术途径

7.3.1 人工鱼

3. “人工鱼”的创作

“人工鱼”的动画创作方法和技术，已经突破了传统的计算机动画的框架。

首先，“人工鱼”不仅有逼真于“自然鱼”的外形和彩色，且具有类似于“自然鱼”的运动和姿态。

其次，“人工鱼”不仅具有“自然鱼”的形态，而且具有“自然鱼”类似的生命特性——“活性”。

再次，“人工鱼”是具有人工智能的“灵巧鱼”，而传统的“动画鱼”是程序化的“木偶鱼”。

最后，“人工鱼”是具有各种不同的人工鱼的鱼群社会。

7.3.1 人工鱼

4. “人工鱼”有何意义和价值？

- (1) “人工鱼”开拓了计算机动画创作的新途径。
- (2) “人工鱼”提供了“人工生命”的新范例。
- (3) “人工鱼”实现了分布式“人工智能”。

7.3.2 Terrarium世界

- ◆ 饲养场模式（Terrarium Mode）：这种模式给用户提供了两种选择。
 - 用户可以独立运行，无须与其他节点连接。在这种情况下，屏幕上显示的生态系统就代表了整个生态环境。这种模式最适合于对我们开发出来的生物进行测试。
 - 用户也可以选择加入到一组特定的节点环境中，所有连入此环境的计算机共同组成一个小型生态系统。

7.3.2 Terrarium世界

加入特定节点组的方法非常简单，每个用户选择一个特定的专网，并在Terrarium控制台的“Channel”一项中输入一个事先约定好的字串，就可以加入该网络了。输入字串后，用户的计算机只与那些输入了相同字串的计算机构成一个独立的对等网络，并一同组建生态系统。

7.3.2 Terrarium世界

◆ 生物圈模式（Ecosystem Mode）：这是游戏的标准模式。全世界所有连入游戏的计算机共同构成一个完整的生态系统。每个参与者的计算机只是该生态系统的一个局部场景。

在上述两种模式下，开发者都可以使用Terrarium类库、.NET框架开发包和Visual Studio .NET工具随意创建它们自己的生物。或者，可以简单地把Terrarium当作一个独立运行的应用程序或是屏幕保护程序运行，并通过Terrarium观看其他开发者创建的生物在生态系统中为生存而战。

7.3.2 Terrarium世界



7.4 群集智能的优缺点

特点和优点：

1. 群体中相互合作的个体是分布的 (Distributed)，这样更能够适应当前网络环境下的工作状态；
2. 没有中心控制的机制与数据，这样的系统更具有鲁棒性 (Robust)，不会由于某一个或者某几个个体的故障而影响整个问题的求解；

7.4 群集智能的优缺点

3. 可以不通过个体之间直接通信而是通过非直接通信进行合作，这样的系统具有更好的可扩充性(Scalability)，由于系统中个体的增加而造成的系统的通信开销的增加在这里非常小；
4. 系统中每个个体的能力十分简单，这样每个个体的执行时间比较短，并且实现也比较简单，具有简单性(Simplicity)。

7.4 群集智能的优缺点

缺陷：

1. 并不是总能从涌现的群体行为中推导出个体的行为；
2. 真实生物个体如此复杂，以至于几乎不可能在一个仿真系统中完成复制；
3. 简单的规则产生类似生命的群体行为并不能保证真实的生态系统一定能够遵循这些简单的规则。