群体机器人研究前沿在线会议

2020年11月7日-8日 北京

北京郵電大學

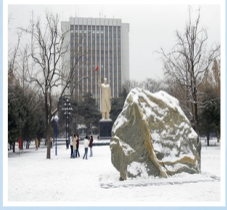
BEIJING UNIVERSITY OF POSTS AND TELECOMMUNICATIONS

群体智能理论与技术

李丽香

北京邮电大学网络空间安全学院

网络与交换技术国家重点实验室



我的课题组

人员组成：5个教授，3副教授，1个讲师

硕（博）在读学生：每个老师3-5硕士生/年，1博士生/年，每个老师平均>20

研究领域：网络安全，如区块链、量子密码、格密码、信息隐藏、数字水印、秘密共享、免疫系统安全、GAN生成对抗性网络；人工智能，如群体智能、忆阻神经网络、深度学习等；复杂网络、压缩感知等

智能优化算法：又称为现代启发式算法，是一种具有全局优化性能、通用性强、且适合于并行处理的算法

智能优化算法的共同特点

- 这些算法都是从任一解出发，按照某种机制，以一定的概率在整个求解空间中探索最优解。由于它们可以把搜索空间扩展到整个问题空间，因而具有全局优化性能

常用智能算法

- 遗传算法
- 模拟退火算法
- 禁忌搜索算法
- 差分进化算法
- 群体智能优化算法

群体智能算法

- 粒子群算法
- 蚂蚁种群优化算法
- 细菌觅食算法
- 人工免疫系统优化算法
- 蜂群优化算法

内容提要

- 粒子群优化算法
- 蚂蚁种群优化算法
- 细菌觅食算法
- 人工免疫系统算法

内容提要

- 粒子群优化算法
- 蚂蚁种群优化算法
- 细菌觅食算法
- 人工免疫系统算法

师法自然

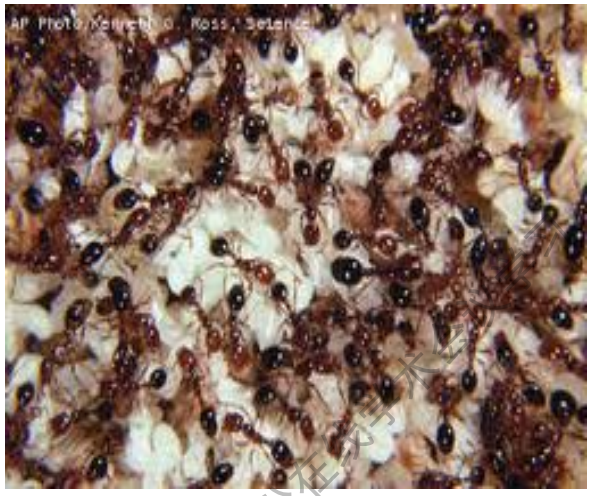
- 大自然始终是开启人类智慧的老师
- 启发：社会系统、物理系统、生物系统
- 建立和发展起一个个研究工具和手段
- 解决和攻克优化、智能领域研究过程中遇到的困难
- 经典算法：遗传算法、人工神经网络、禁忌搜索、模拟退火和进化规划等
- 这些算法已经被广泛应用于工程

人工智能研究的停滞

- 20世纪80年代：人工智能繁荣了整整10年
- 90年代初：方法论上始终没有突破经典计算思想
- 各种反思和批驳：以Penrose等为代表的论著纷纷涌现
- 人工智能的研究进入了寒冬季节

社会性群居动物的群体智能行为

albatrosses



ants

grey seals

生命科学的发展

- 随着人们对生命的本质的不断了解，人工智能的研究开始摆脱经典计算思想的束缚，大胆探索起新的非经典计算途径；
- 自然界中的群体行为：成群的鸟、鱼或者浮游生物。这些生物的聚集行为有利于它们觅食和逃避捕食者；
- 数以百万计的蚂蚁如何组成一个群落？在蚁群中，单个蚂蚁的能力和智慧如此简单，不论单个工蚁还是蚁后都不可能有足够的的能力来指挥完成筑巢、觅食、迁徙、清除蚁穴等复杂行为

群集智能的产生

- 人工智能先驱Minsky认为“我们应该从生物学而不是物理学受到启示”
- 对生物启发式计算（Bio-inspired Computing）的研究，成为人工智能开创新纪元的又一个里程碑
- 在这种背景下，社会性动物（如蚁群、蜂群、鸟群等）的自组织行为引起了人们的广泛关注
- 许多学者对这种行为进行数学建模并用计算机对其进行仿真，这就产生了所谓的“群集智能”（Swarm Intelligence，SI）

群体智能

群体智能，是指无智能的或具有简单智能的个体通过群体协作和组织来表现出群体智能行为的特性

群智能理论

群智能理论，就是利用群体的协作和组织等行为来进行智能优化的理论

群体智能利用群体的优势，在没有集中控制、不提供全局模型的前提下完成智能操作，它为寻找复杂问题的解决方案提供了新的思路

群集智能的特点

- **鲁棒性**：没有中心的控制与数据，不会因某一个或几个个体的故障而影响整个问题的求解
- **简单性**：群体中个体的能力比较简单，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
- **灵活性**：由于群体可以更好的适应随时间变化的环境，所以群体系统具有更强的灵活性
- **并行性**：群体智能是基于群体运行的，具有本质并行性，易于硬件并行实现
- **可扩充性**：个体之间通过间接通讯进行合作，系统个体的增加而引起的通信开销的增加很小，这样的系统具有可扩充性
- **自组织**：活动既不受中央控制，也不受局部监管
- **分布性**：群体中相互合作的个体是分布的，更能适应当前网络环境下的工作状态

群集智能发展简史

年份	学者	提出的模型
1987	Craig Reynolds	人工鸟系统模型（BOID）
1992	Marco Dorigo	蚂蚁系统模型（Ant System, AS ）
1995	Kennedy	粒子群优化模型（Particle Swarm Optimization）
1996	Gambardella	蚁群系统 (Ant Colony System, ACS)
1999	Dorigo和 Gambardella	蚂蚁种群优化 (Ant Colony Optimization, ACO)
1999	Bonabeau 和 Dorigo	《Swarm Intelligence: From Natural to Artificial System》出版，Swarm Intelligence 的概念被正式提出

群集智能发展简史

1999-至今，群体智能成为国际智能计算领域关注的热点和前沿课题。之后，又有学者提出了细菌觅食优化算法(Bacterial foraging optimization)、人工免疫系统优化算法、人工蜂群算法(Artificial bee colony)、人工鱼群算法(Artificial fish swarm algorithm)、杜鹃搜索算法(Cuckoo search)、萤火虫算法(firefly algorithm)、灰狼算法(Grey wolf optimizer)、鲸鱼算法(Whale optimization algorithm)、群搜索算法(Group search optimizer)、混合蛙跳算法(Shuffled frog leaping algorithm)、烟花算法(Fireworks algorithm)等

人工鸟系统

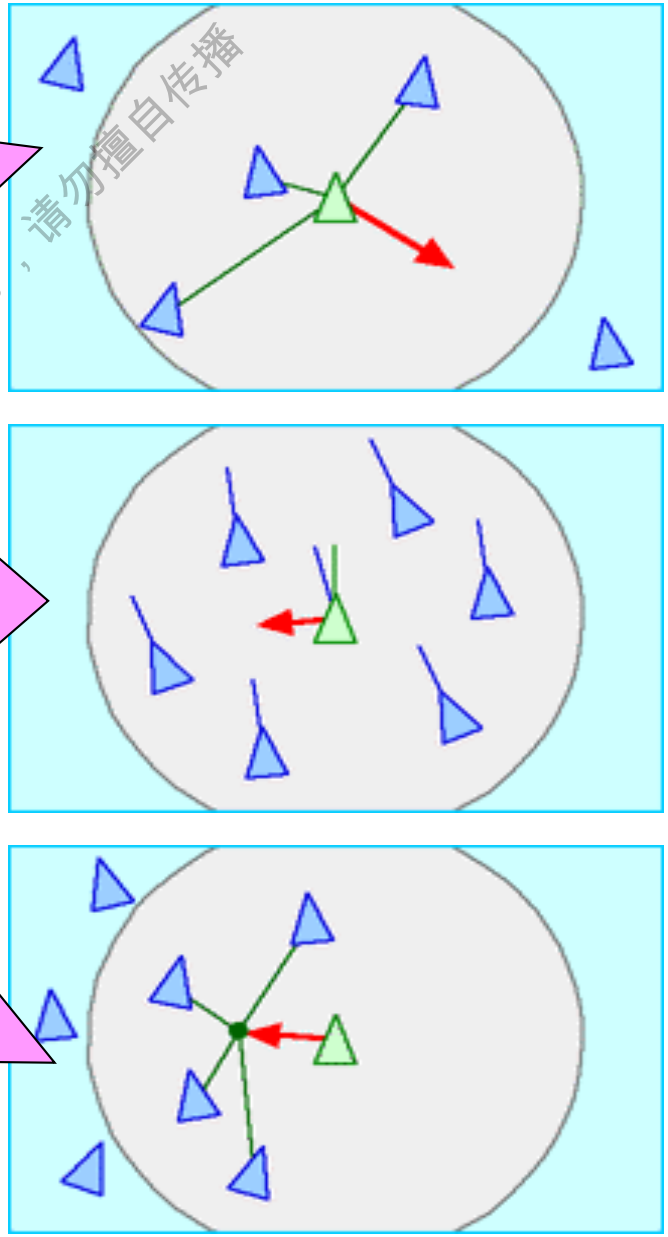
人工鸟系统中每只人工鸟有三种飞行规则：分离、列队及聚集，并且能够感知周围一定范围内其它鸟的飞行信息。人工鸟根据该信息，结合其自身当前的飞行状态，在三条简单行为规则的指导下做出下一步的飞行决策。

Reynolds 用计算机模拟了该系统的行为，每个人工鸟能够在快相撞时自动分开，遇到障碍物分开后又重新合拢

Computer Graphics, 21(4), July 1987, 25-34

人工鸟系统

- 局部控制规则
  - ① **分离**: 快速相撞时自动分开
  - ② **列队**: 根据邻居的平均飞行趋势选择飞行方向
  - ③ **聚集**: 分开后又重新合拢
- 全局的群体行为



建立在模拟鸟群社会的基础上，通过图形来模拟鸟群优美和不可预测的舞蹈动作，发现鸟群支配同步飞行和以最佳队形突然改变飞行方向并重新编队的能力

以粒子对解空间中最优粒子的追随进行解空间的搜索

James Kennedy & Russell Eberhart 于1995年提出

粒子群搜索的基本原理

同蚂蚁算法类似，粒子群优化(Particle swarm optimization, PSO)算法是基于群体行为的演化算法，其思想来源于人工生命和演化计算理论。

通过对鸟群飞行的研究发现，鸟仅仅是追踪有限数量的邻居，但最终的整体跟踪结果是整个鸟群好像在一个中心的控制之下，即复杂的群体行为是由简单规则的相互作用引起的。

粒子群优化源于对鸟群捕食行为的研究，一群鸟在随机搜寻食物，如果这个区域里只有一块食物，那么找到食物的最简单有效的策略就是搜寻目前离食物最近的鸟的周围区域。粒子群优化算法就是从这种模型中得到启示而产生的，并用于解决优化问题。

粒子群搜索的基本原理

PSO求解优化问题时，问题的解对应于搜索空间中一只鸟的位置，称这些鸟为“粒子(particle)”或“主体(agent)”。每个粒子都有自己的位置和速度（决定飞行的方向和距离），还有一个由被优化函数决定的适应值。各个粒子记忆、追随当前的最优粒子，在解空间中搜索。每次迭代的过程不是完全随机的，如果找到较好解，将会以此为依据来寻找下一个解。

另外，人们通常是以他们自己及他人的经验来作为决策的依据，这就构成了粒子群优化的一个基本概念

粒子群算法的数学模型——全局版

在每个时间步骤 t

对于每个粒子i

对每个 d 维分量

更新速度

$$v_{id}(t+1) = v_{id}(t) + c_1 rand_1 \times (pbest_{id}(t) - x_{id}(t)) + c_2 rand_2 \times (gbest_d(t) - x_{id}(t))$$

随机的

然后移动 $x_{id}(t+1) = x_{id}(t) + v_{id}(t+1)$

粒子群算法的数学模型——全局版

- 令PSO初始化为一群随机粒子（随机解）
- 在每一次迭代中，粒子通过跟踪两个“最好位置”来更新自己
  - 个体最好位置：就是粒子本身所找到的最好位置，用*pbest*表示
  - 全局极值点：全局PSO中的另一个极值点是整个群体目前找到的最好解，用*gbest*表示
  - 局部极值点：局部PSO中不用整个群体而是用其中一部分作为粒子的邻居，所有邻居中的最好解就是局部极值点，用*ibest*表示其位置
- 第i个粒子的位置可以用D维向量表示，即$\boldsymbol{X}_i=(x_{i1},x_{i2},\ldots,x_{iD})^T$
- 第i个粒子的速度表示为$\boldsymbol{V}_i=(v_{i1},v_{i2},\ldots,v_{iD})^T$
- 其它向量类似进行定义

在找到最好解后，粒子根据如下的式（1）和式（2）来更新自己的速度和位置：

$$v_{id}(t+1)=v_{id}(t)+c_1 rand_1\times(pbest_{id}(t)-x_{id}(t))+c_2 rand_2\times(gbest_d(t)-x_{id}(t)) \quad (1)$$

$$x_{id}(t+1)=x_{id}(t)+v_{id}(t+1) \quad (2)$$

其中$v_{id}(t)$是第i个粒子在第t次迭代中第d维的速度；

$x_{id}(t)$是第i个粒子在第t次迭代中第d维的当前位置；

$rand_1$,$rand_2$是[0, 1]之间的均匀分布的随机数；

$pbest_{id}(t)$是第i个粒子第t次迭代中在第d维的个体最好点的位置(或者坐标)；

$gbest_d(t)$是整个群第t次迭代中在第d维的全局最好点的位置

c_1,c_2是加速系数(或学习因子)，分别调节向全局最好粒子和个体最好粒子位置飞行的最大步长。若太小，则粒子可能远离目标区域，若太大则会导致突然向目标区域飞去或飞过目标区域。选择合适的c_1,c_2可以加快算法收敛速度且不致陷入局部最优，通常令$c_1=c_2=2$

为防止粒子远离搜索空间，粒子的每一维速度v_d都会被限制在[-v_{dmax}, v_{dmax}]之间，v_{dmax}太大，粒子将飞离最好解，v_{dmax}太小，粒子将陷入局部最优。假设将搜索空间的第d维定义为区间[-v_{dmax}, v_{dmax}]，则通常$v_{dmax}=kx_{dmax}$，$0.1\le k\le 1.0$，每一维都用相同的设置方法

粒子群算法的数学模型——局部版

在每个时间步骤 t

对于每个粒子

对每个 d 维分量

更新速度

$$\begin{cases} v_{id}(t+1)= \\ v_{id}(t) \\ +c_1 rand_1 \times (pbest_{id}(t)-x_{id}(t)) \\ +c_2 rand_2 \times (ibest_{id}(t)-x_{id}(t)) \end{cases}$$

随机的

然后移动 $x_{id}(t+1)=x_{id}(t)+v_{id}(t+1)$

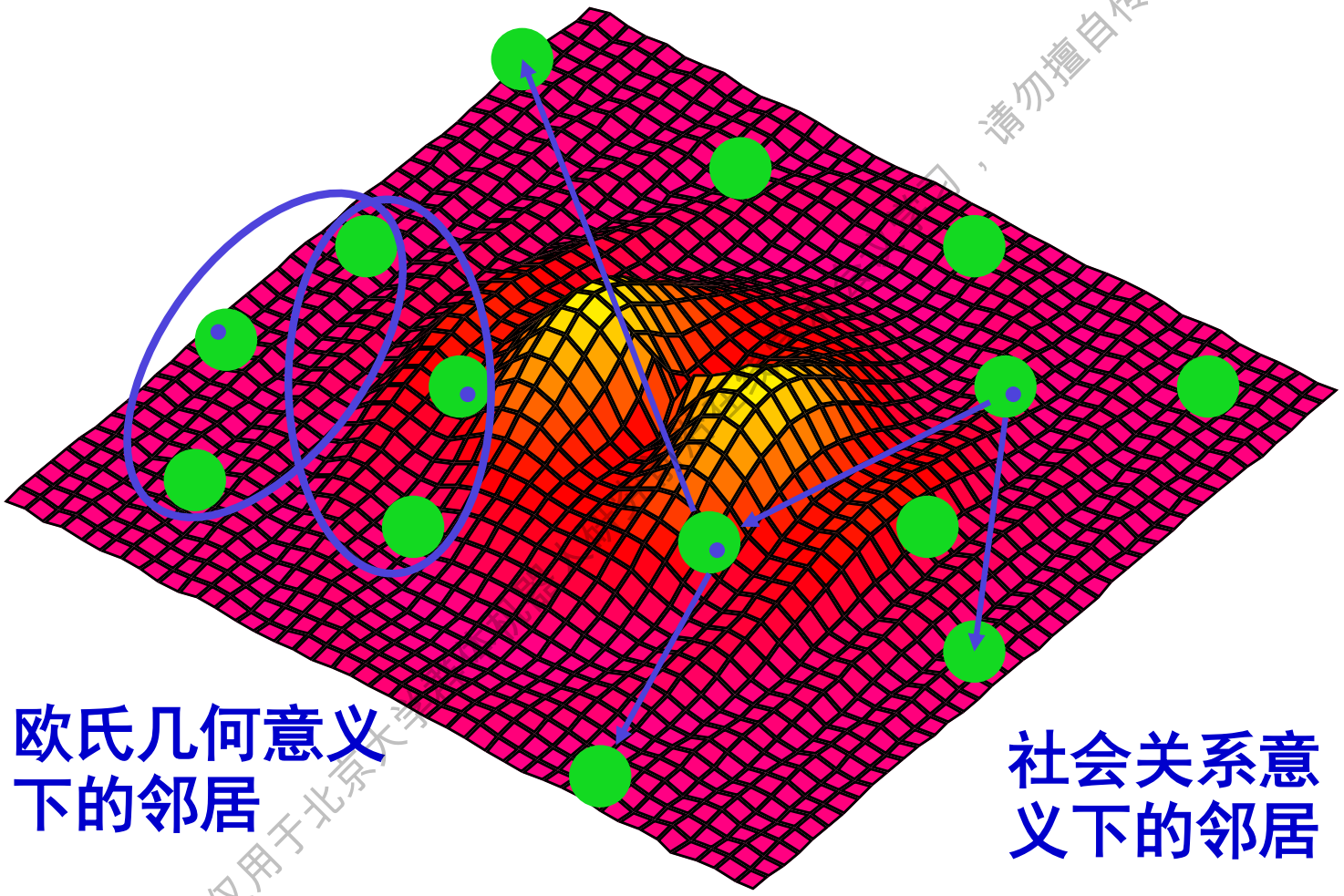
局部版PSO算法中邻居的定义

基本的PSO算法有全局版和局部版两种，全局版收敛快，但有时会陷入局部最优。基本的局部版PSO算法通过保持多个吸引子来避免早熟，每个粒子的邻域是基于粒子序号划分的，即

$$N_i = \{pbest_{i-l}, pbest_{i-l+1}, \ldots, pbest_{i-1}, pbest_i, pbest_{i+1}, \ldots, pbest_{i+l-1}, pbest_{i+l}\}$$

从N_i中选出最好的, 将其位置作为$ibest$代替公式中的$gbest$, 其他与全局版PSO相同。实验表明，局部版比全局版收敛慢，但不容易陷入局部最优。在实际应用中, 可以先用全局版找到大致的结果, 再用局部版进行搜索

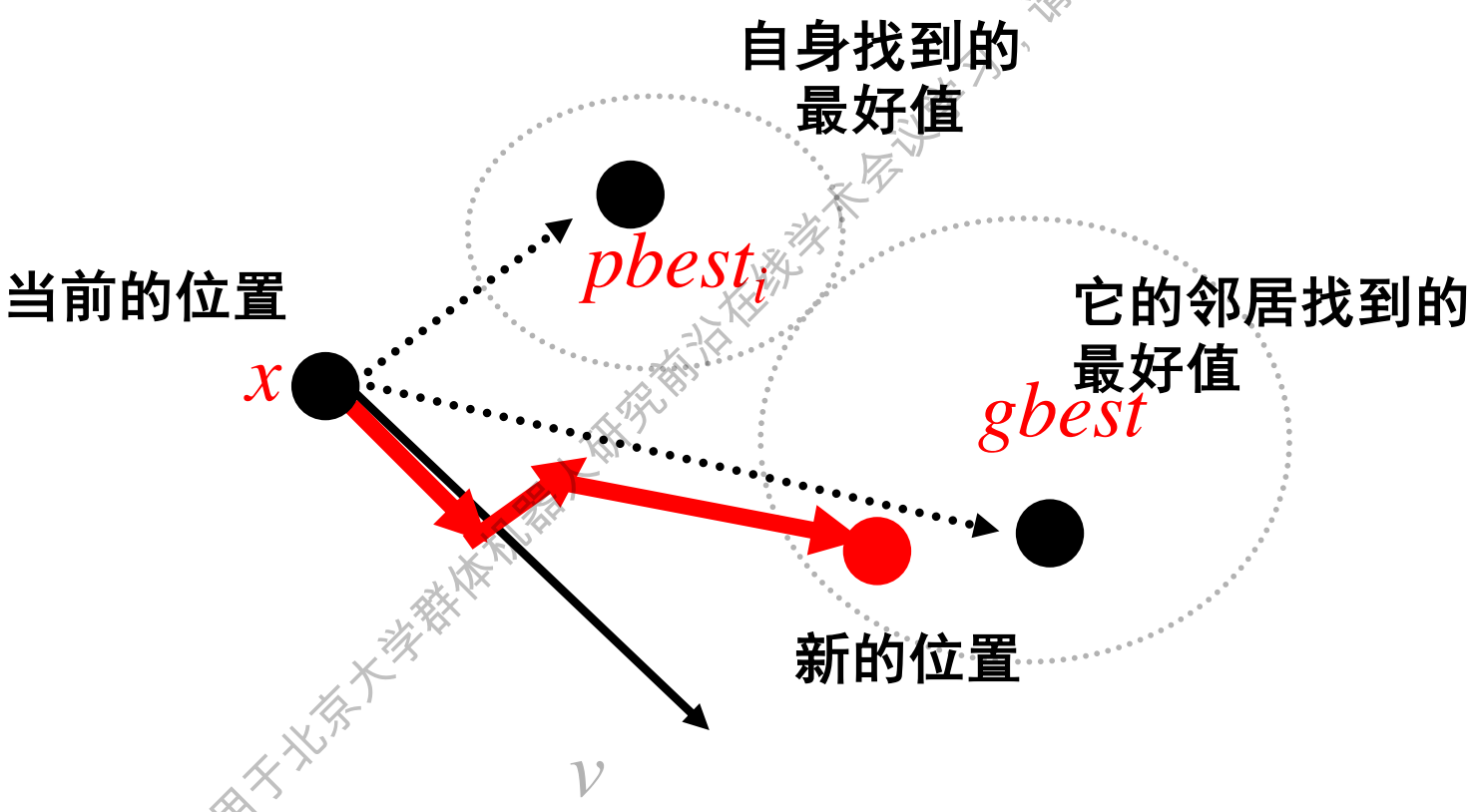
两种邻居的定义方式



欧氏几何意义下的邻居

社会关系意义下的邻居

粒子群个体行为分析



基本PSO算法的步骤

步骤1（初始化）：在允许的范围内随机产生初始搜索点的位置$x_{id}(0)$及其速度$v_{id}(0)$，每个粒子的$pbest_{id}(0)$坐标设置为其当前位置，且计算出其相应的个体最好位置的适值，而全局最好位置就是个体最好位置中适值最好的，记录该最好位置的粒子序号，并将$gbest_d(0)$设置为该最好粒子的当前位置；

步骤2（评价每一个粒子）：计算每个粒子的适值，如果好于该粒子当前的最好个体适值，则将$pbest_{id}(1)$设置为该粒子的位置，并且更新个体最好位置。如果所有粒子的个体位置中最好的位置的适值好于当前的全局最好位置的适值，则将$gbest_d(1)$设置为该粒子的位置，记录该粒子的序号，且更新全局最好位置；

步骤3（粒子的更新）：用式（1）和式（2）对每一个粒子的速度和位置进行更新；

步骤4（检验是否符合结束条件）：如果当前的迭代次数达到了预先设定的最大次数（或达到最小错误要求），则停止迭代，输出最优解，否则转到步骤2。

粒子群算法的特点

- 基本算法最初是处理连续优化问题的
- 类似于遗传算法，也是多点搜索
- 式(1) 的第一项对应多样化(diversification) 的特点，第二项、第三项对应于搜索过程的集中化(intensification) 特点，因此，这个方法在多样化和集中化之间建立均衡

二进制版本的粒子群的数学模型

Eberhart等又提出了PSO的离散二进制版，用来解决工程实际中的组合优化问题。他们将每一维x_{id}和$pbest_{id}$限制为1或者为0，而速度v_{id}不作这种限制。用速度来更新位置时，如果v_{id}高一些，粒子的位置x_{id}更有可能选1，如果v_{id}低一点则x_{id}选0，阈值在[0,1]之间，而有这种特点的函数就是Sigmoid函数：

$$sig(v_{id}(t)) = \frac{1}{1+\exp(-v_{id}(t))}$$

二进制版本的粒子群优化的公式为

如果 $\rho_{id}(t+1) < sig(v_{id}(t+1))$ 那么 $x_{id}(t+1)=1$ 否则 $x_{id}(t+1)=0$

向量$\rho_{id}(t+1)$的各个分量是[0,1]之间的随机数。为了防止*Sigmoid*函数饱和, 可以将$v_{id}(t)$限制在[-4.0,4.0]之间。二进制PSO其他部分与连续PSO基本类似。实验结果标明, 在大多数测试函数中，二进制PSO都比遗传算法速度快，尤其在问题的维数增加时

粒子群算法的改进

- **收敛速度的改进**
  - （1）惯性权重的改进
  - （2）模糊惯性权重法
  - （3）约束因子法
- **增加粒子多样性的改进**
  - （1）空间邻域法
  - （2）邻域拓扑法
  - （3）社会趋同法
- **全局方法**
  - （1）序列生境技术
  - （2）函数延伸法
  - （3）动态目标函数
- **算法融合**

为了进一步提高PSO的基本性能，许多研究者还尝试了将其与其它计算智能方法相融合，以突破其自身局限

惯性权重（inertia weight）法

Shi等提出了惯性权重的方法。惯性权重w是与上一次迭代速度有关的一个比例因子，速度更新方程为

$$v_{id}(t+1)=wv_{id}(k)+c_1rand_1(pbest_{id}(t)-x_{id}(t))+c_2rand_2(gbest_d(t)-x_{id}(t))$$

用惯性权重来控制前面的速度对当前速度的影响, 较大的w可以加强PSO的全局搜索能力，而较小的w能加强局部搜索能力。基本PSO可以看作w=1，因此，在迭代后期缺少局部搜索能力。实验结果表明，w在[0.8, 1.2]之间PSO有更快的收敛速度。后来又试验了将w设置为从0.9到0.4的线性下降，使得PSO在开始时探索较大的区域，较快地定位最优解的大致位置，随着w逐渐减小，粒子速度减慢，开始精细的局部搜索（这里w类似于模拟退火中的温度参数）。该方法加快了收敛速度，提高了PSO算法的性能。当待求解问题很复杂时，该方法使得PSO在迭代后期全局搜索能力不足，导致不能找到要求的最优解，这可以用自适应改变惯性权重来克服

模糊惯性权重法

Shi等提出**用模糊控制器来动态自适应地改变惯性权重**w的技术，即构造一个2输入1输出的模糊控制器来动态地修改惯性因子。控制器的输入是当前惯性权重w和当前最好性能评价值(current best performance evaluation，$CBPE$)，输出是w的改变量。$CBPE$衡量PSO算法目前找到的最好候选解的性能。由于不同的问题有不同范围的性能评价值，因此，需要对$CBPE$进行如下的规范化

$$NCBPE = \frac{CBPE - CBPE_{\min}}{CBPE_{\max} - CBPE_{\min}}$$

NCBPE是规范化后的评价值，$CBPE_{\min}$和$CBPE_{\max}$分别是$CBPE$可能取值的上下限，依问题而定，且需事先得知或者可以估计。

模糊惯性权重法与线性下降惯性权重方法的比较结果表明，后者不知道应该降低w的合适时机，而**自适应模糊控制器能预测使用什么样w的更合适，可以动态地平衡全局和局部搜索能力**。但是由于需知道$CBPE_{\min}$和$CBPE_{\max}$等, 使得模糊权重法的实现较为困难，因而无法广泛使用

约束因子（constriction factor）法

Clerc通过研究得到约束因子有助于确保PSO算法收敛的结论。这种方法的速度更新方程为

$$v_{id}(t+1)=\chi\left(v_{id}(t)+c_1 rand_1\left(pbest_{id}(t)-x_{id}(t)\right)+c_2 rand_2\left(gbest_d(t)-x_{id}(t)\right)\right)$$

其中 $\chi=\dfrac{2}{\left|2-\phi-\left(\phi^2-4\phi\right)^{1/2}\right|}$ 为约束因子，$\phi=c_1+c_2$，且$\phi>4$。

约束因子法控制系统行为最终收敛，且可以有效搜索不同的区域，该方法能得到高质量的解，如果与此同时将每维$v_{d\max}$设置为一维搜索空间的大小$x_{d\max}$，则可以得到更好的效果。

空间邻域(spatial neighborhoods)法

Suganthan提出了基于粒子的空间位置划分的方案。在迭代中，计算每一个粒子与群中其他粒子的距离，记录任何2个粒子间的最大距离为$d_{\max}$。对每一粒子按照

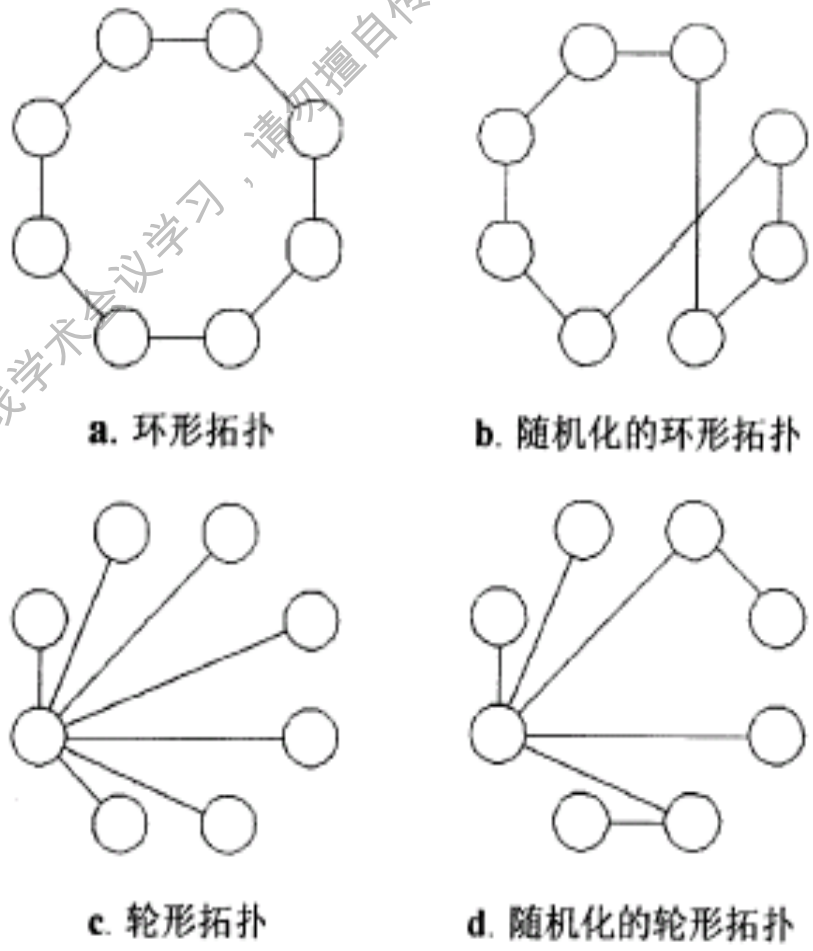
$$\|X_a - X_b\| / d_{\max}$$

计算一个比值，其中$\|X_a - X_b\|$是当前粒子a到粒子b的距离

而选择阈值$frac$根据迭代次数而变化。当另一粒子b满足$\|X_a - X_b\| / d_{\max} < frac$时，则$b$成为当前粒子的邻域。所有满足该条件的粒子组成$N_i'$，其他同基本的局部版PSO算法。应用改进的邻域规则，且采用时变w的PSO在绝大多数测试中都取得了比全局版PSO更优良的性能

邻域拓扑(neighborhood topologies)法

每个粒子左右两边的粒子都是它的邻居, 实际上这是应用了环形拓扑。Kennedy测试了如右图所示的几种邻域拓扑结构，结果表明，**拓扑非常影响算法的性能，且最佳的拓扑形式因问题而定**。如对有很多局部最优的函数，轮形拓扑邻域算法能得到最好的结果，Kennedy推测这是由于较慢的信息流动；而对于单峰函数，星形拓扑邻域算法可以产生较好的结果，因为它有较快的信息流动（这里的拓扑都是在粒子序号空间下的拓扑）



几种邻域拓扑结构

社会趋同(social stereotyping)法

Kennedy提出了混和空间邻域和环形拓扑方法的另一个局部PSO版本，称为社会趋同法。因为生活中人们往往是试图追随一个群体的共同观点，而不是群体中某个人的立场。将该思想应用到PSO中，即不用每个粒子的经验而是用它所属空间聚类的共同经验来更新自己

序列生境技术(sequential niche technique, SNT) 法

Beasley等提出的最初使用在遗传算法中的序列生境技术可以系统地访问每一个全局极值。其思想是在找到每一个极值后, 都用下降函数来自适应地改变适应值函数, 如此算法就不会再回到该极值。虽然将SNT引入PSO会带来诸如参数选择、引入更多局部极值等问题, 但是该法能够枚举所有全局极值, 在多目标优化问题上还是很有意义的

很多现实优化问题的目标函数是随时间变化的, 因此已经有很多学者开始致力于这方面的研究。

Parsopoulos等的试验表明, 基本PSO就可以有效而稳定地在噪声环境中工作, 且在很多情况下, 噪声的存在还可以帮助PSO避免陷入局部最优。

Carlisle等的研究得出PSO不能跟随一个非静态的目标函数

Eberhart等提出一种动态惯性权重法试图解决这一问题, 该方法令w=0.15+$r(t)$/2.10, $r(t)$ 是(0 ,1) 间的随机数(w的平均值为0.175) , 常数c_1=c_2= 1.1494 , 这是受到约束因子方法的启发而得出的。结果表明能跟随非静态目标函数, 比进化规划和进化策略得到的结果精度更高

函数延伸(function stretching) 法

为了减小定位所有全局极值的花费, Parsopoulos等将自适应改变目标函数方法应用到了PSO中。他们提出一个两步的转化过程来改变目标函数, 以防止PSO返回已经找到的局部极值。该方法能跳出局部最优, 有效定位全局最优点, 有助于PSO稳定地收敛, 虽然增加了计算量, PSO的成功率却有明显的提高。但该方法不能枚举全部最优

选择（selection）法

Angeline提出的混和PSO，主要应用PSO的基本机制以及演化计算所采用的自然选择机制。由于PSO搜索过程依赖pbest和gbest，所以搜索区域有可能被他们限制住了。自然选择机制的引入将会逐渐减弱其影响。测试结果表明，虽然在大多数测试函数中选择法取得了比基本PSO更好的效果，却在Griewank函数上得到了较差的结果。因此，该方法提高了的局部PSO搜索能力，但同时削弱了全局搜索能力

繁殖(Breeding)法

Lovbjerg等人将遗传算法中的复制和重组这些称为繁殖的操作加入到全局版PSO中, 该方法是对按概率p_i选出的粒子进行如下式

$$child_1(X_i) = p_i\, parent_1(X_i) + (1.0 - p_i)\, parent_2(X_i)$$

$$child_2(X_i) = p_i\, parent_2(X_i) + (1.0 - p_i)\, parent_1(X_i)$$

$$child_1(V_i) = \frac{parent_1(V_i) + parent_2(V_i)}{\left|parent_1(V_i) + parent_2(V_i)\right|} * \left|parent_1(V_i)\right|$$

$$child_2(V_i) = \frac{parent_1(V_i) + parent_2(V_i)}{\left|parent_1(V_i) + parent_2(V_i)\right|} * \left|parent_2(V_i)\right|$$

的代数杂交操作，产生子代的粒子取代父代。选择父代没有基于适应值，防止了基于适应值的选择对那些多局部极值的函数带来潜在问题。p_i是(0,1)间的随机数（经验值约为0.2）。理论上讲繁殖法可以更好地搜索粒子间的空间，2个在不同次优峰处的粒子经繁殖后，可以从局部最优逃离。结果显示，对单峰函数，繁殖法虽略加快了收敛速度，却不如基本PSO和GA找到的解好，而对于多局部极值的函数，繁殖PSO不仅加快了收敛速度，而且找到了同样好或更好的解

保收敛(guaranteed convergence)PSO

若粒子的当前位置恰是全局最好位置, 那么速度更新方程式就只剩下$v_{id}(t)$, 这将会导致早熟。后来又提出一种确保PSO收敛到局部最优的改进方法(GCPSO) , 其策略是对全局最好粒子用一个新的更新方程, 该方程将该粒子的位置复位到全局极值点, 并且使其在全局最好位置附近产生一个随机搜索, 而其他粒子仍用原方程更新。该方法较基本的PSO在收敛速度上有很大提高, 尤其是当粒子数较少时, 在单峰函数中性能提高明显。但粒子数较少时, 在单峰函数中性能提高明显。但是该改进同基本PSO一样不能保证算法收敛到全局最优

协同(cooperative)PSO

协同PSO是将n维向量分到n个粒子群中, 每个粒子群优化一维向量, 评价适应值时将分量合成一个完整向量。例如，对第j个粒子群, 除第j个分量外, 其他n-1个分量都设为最好值, 不断用第j个粒子群中的粒子替换第j个分量, 得到第j维的最好值, 其他维相同。为将有联系的分量划分在一个群, 可将n维向量分到k个粒子群来优化。称这种算法为CPSO-S_k , 它在某些问题上有更快的收敛速度, 但该算法容易被欺骗, 而GCPSO可以保证收敛到局部最优解, 因此，作者又将这两种方法结合, 结果表明, 结合版拥有两者的优点, 取得了较好的效果

粒子群算法的优点

- 同遗传算法类似，是一种基于迭代的优化工具
- 与遗传算法比较，优势在于概念简单，在算法实现过程中没有交叉变异操作，以粒子对解空间中最优粒子的追随进行解空间的搜索
- 可直接采用实数编码
- 算法参数简洁，所需调整的参数少，无需复杂的调整
- 算法流程、结构简单，容易实现，同时又有深刻的智能背景
- 收敛速度快，既适合科学研究，又特别适合工程应用

粒子群优化算法与其它进化算法一样,可以解决几乎所有的优化问题,或者是可以转换为优化问题进行求解的问题。其中最具应用前景的领域包括多目标问题的优化、系统设计、分类、模式识别、生物系统建模、规划、信号处理、决策和模拟等。目前，在模糊控制器的设计、车间任务调度、机器人实时路径规划、图像分割、EEG信号模拟、语音识别、时频分析、烧伤诊断以及自动探测移动目标等方面已经有成功应用的先例

内容提要

- 粒子群优化算法
- **蚂蚁种群优化算法**
- 细菌觅食算法
- 人工免疫系统算法

蚂蚁种群优化算法

- 20 世纪90 年代以来，一种新型的分布式智能模拟算法，即蚂蚁种群优化算法(ant colony optimization algorithm，ACO)，开始引起人们的注意并逐渐得到应用。这种算法同其它模拟进化算法一样，都是从对自然界的观察中受到启发而产生的。 蚁群种群优化算法的基本思想是**模仿蚂蚁依赖信息素(pheromone) 进行通信而显示出的社会性行为，在智能体(agent) 定义的基础上，由一个贪心法指导下的自催化(autocatalytic) 过程引导每个智能体(agent) 的行动**。
- 它具有通用性和鲁棒性，是基于群体智能优化的方法。它是一种随机的通用试探法，可用于解各种不同的组合优化问题。 蚂蚁种群优化算法表现出了强大的生命力

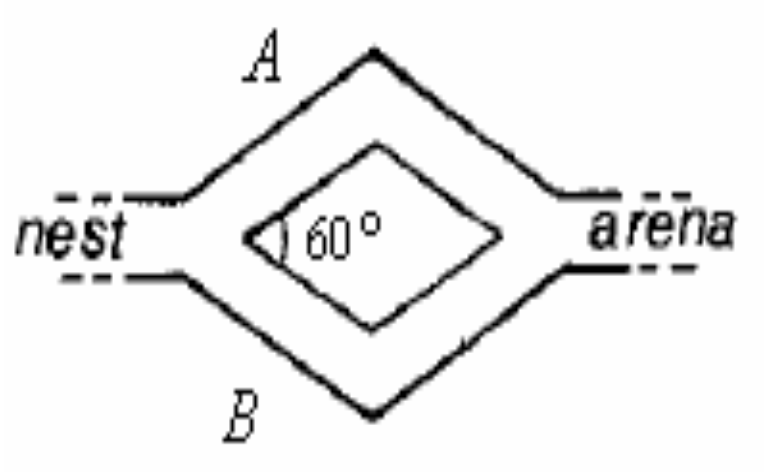
蚂蚁的群体行为

- 蚂蚁是最古老的社会昆虫之一，它的个体结构和行为虽简单，但由这些简单个体构成的**蚂蚁群体，却表现出高度结构化的社会组织**。蚂蚁王国俨然是一个小小“社会”。这里，有专司产卵的蚁后；有为数众多的从事觅食打猎、兴建屋穴、抚育后代的工蚁；有负责守卫门户、对敌作战的兵蚁、还有专备后蚁招婿纳赘的雄蚁等等。
- 蚂蚁是社会性昆虫，组成社会的三要素之一就是社会成员除有组织、有分工之外，还有相互的通讯和信息传递。研究表明，**蚁群有着奇妙的信息系统**。其中包括视觉信号、声音通讯和更为独特的无声语言，即包括**化学物质不同的组合、触角信号和身体动作**在内的多个征集系统，来策动其它个体。蚂蚁特有的控制自身环境的能力，是在高级形式的社会性行为及不断进化过程中获得的。

蚂蚁的觅食行为

- 觅食行为是蚁群一个重要而有趣的行为。据昆虫学家的观察和研究，发现**蚂蚁有能力在没有任何可见提示下找出从蚁穴到食物源的最短路径**，并且能随环境变化适应性地搜索新的路径，产生新的选择。虽然单个蚂蚁的行为极其简单，但由大量的蚂蚁个体组**成蚂蚁群体却表现出极其复杂的行为，能够完成复杂的任务**。
- 生物学家和仿生学家经过大量的细致观察研究发现，**蚂蚁在觅食走过的路径上释放一种蚂蚁特有的分泌物——信息素**(Pheromone)。蚂蚁个体之间正是**通过这种信息素进行信息传递，从而能相互协作**，完成复杂任务。**在一定范围内蚂蚁能够察觉到这种信息激素并指导它的行为，当一些路径通过的蚂蚁越多，则留下的信息激素轨迹(trail)也就越多，招致后来更多的蚂蚁选择该路径的概率也越高，于是越发增加了该路径的信息素强度**。这种选择过程称之为蚂蚁的自催化过程，形成一种正反馈机制，可理解为**增强型学习系统**，蚂蚁最终可以发现最短路径。

蚂蚁的觅食行为——双宽桥实验



Wide binary bridge

基于双宽桥实验的蚂蚁觅食行为模型：

$$\text{prob}_A = \frac{(k + A_i)^n}{(k + A_i)^n + (k + B_i)^n};$$

$$\text{prob}_A + \text{prob}_B = 1;$$

$$A_{i+1} = A_i + \delta;$$

$$B_{i+1} = B_i + (1 - \delta);$$

$$A_i + B_i = i.$$

Deneubourg J.L. et al., J Insect Behavior 3, 159-168 (1990).

蚂蚁种群优化算法的原理

- 蚂蚁种群优化算法是模拟真实蚁群觅食过程寻求最短路径的原理，由意大利学者M. Dorigo等人首先提出的。最初的蚂蚁算法称为蚂蚁系统(Ant System)，对于旅行商问题(TSP)及二次分配问题(QAP) 等取得了较好效果，经过一系列改进后称为蚂蚁种群优化算法。
- 蚁群算法吸收了蚂蚁群体行为的典型特征：
  - 能察觉小范围区域内状况，并判断出是否有食物或其他同类的信息素轨迹
  - 能释放自己的信息素
  - 所遗留的信息素数量会随时间而逐步减少
- 蚂蚁算法通过侯选解组织群体的进化过程来寻求最优解，这一过程包括适应阶段和协作阶段：
  - 在适应阶段，各侯选解根据积累的信息不断调整自身的结构
  - 在协作阶段各侯选解间通过信息交流， 以便产生性能更好的解

蚂蚁种群优化算法的原理

- 在蚁群算法中，一个有限规模的人工蚁群体，通过相互协作搜索用于解决优化问题的较优解。
- 每只蚂蚁根据问题依赖的准则，从被选的初始状态出发建立一个可行解或是解的一个组成部分。
- 在建立蚂蚁自己的解决方案中，每只蚂蚁都搜集关于问题拓征和其自身行为的信息，并且使用这些信息来修改问题的表现形式，正如其它蚂蚁所看到的那样。
- 蚂蚁既能共同的行动又能独立的工作，显示出了一种相互协作的行为，它们之间不使用直接通讯，而是使用信息激素指导着蚂蚁间的信息交换。
- 蚂蚁使用一种结构上的贪婪启发式搜索可行解。根据问题的约束条件列出了一个解，作为此时该问题状态的最小代价(最短路径)。每只蚂蚁都能够找出一个解，但可能是较差解。蚁群中的个体同时建立了很多不同的解决方案，找出高质量的解是群体中的所有个体之间全局性的相互协作的结果。

蚂蚁种群优化算法的优点

- ACO原理是一种正反馈机制或称增强型学习系统，它通过信息素的不断更新达到最终收敛于最优路径上；由于其原理是一种正反馈机制，因此，也可将蚂蚁王国(ant colony)理解成所谓的增强型学习系统
- 它是一种通用型随机优化方法，例如它不仅可以用于求解TSP问题还可以用于ATSP问题。而且人工蚂蚁决不是对实际蚂蚁的一种简单模拟，它融进了人类的智能
- 它是一种分布式的优化方法，不仅适合目前的串行计算机，而且适合未来的并行计算机
- 它是一种全局优化的方法，不仅可用于求解单目标优化问题，而且可用于求解多目标优化问题
- 它是一种启发式算法，计算复杂性为$o(NC\cdot m\cdot n)$，其中NC是迭代次数，m是蚂蚁数目，n是目的节点数目
- 它具有鲁棒性，仅做一点小小的改动就可以用于其他的组合优化问题

ACO算法的优化策略

优化策略

1. 选择策略。信息素浓度越大的路径,被选择的概率越大
2. 更新策略。路径上面的信息素浓度会随蚂蚁的经过而增长，而且同时也随时间的推移逐渐减小
3. 协同策略。蚂蚁之间实际上是通过信息素浓度来达到间接的互相通讯、协同工作

这样的策略使得蚁群算法在初始解的基础上经过一段时间后能够发现其中的最优解

ACO算法的选路关键

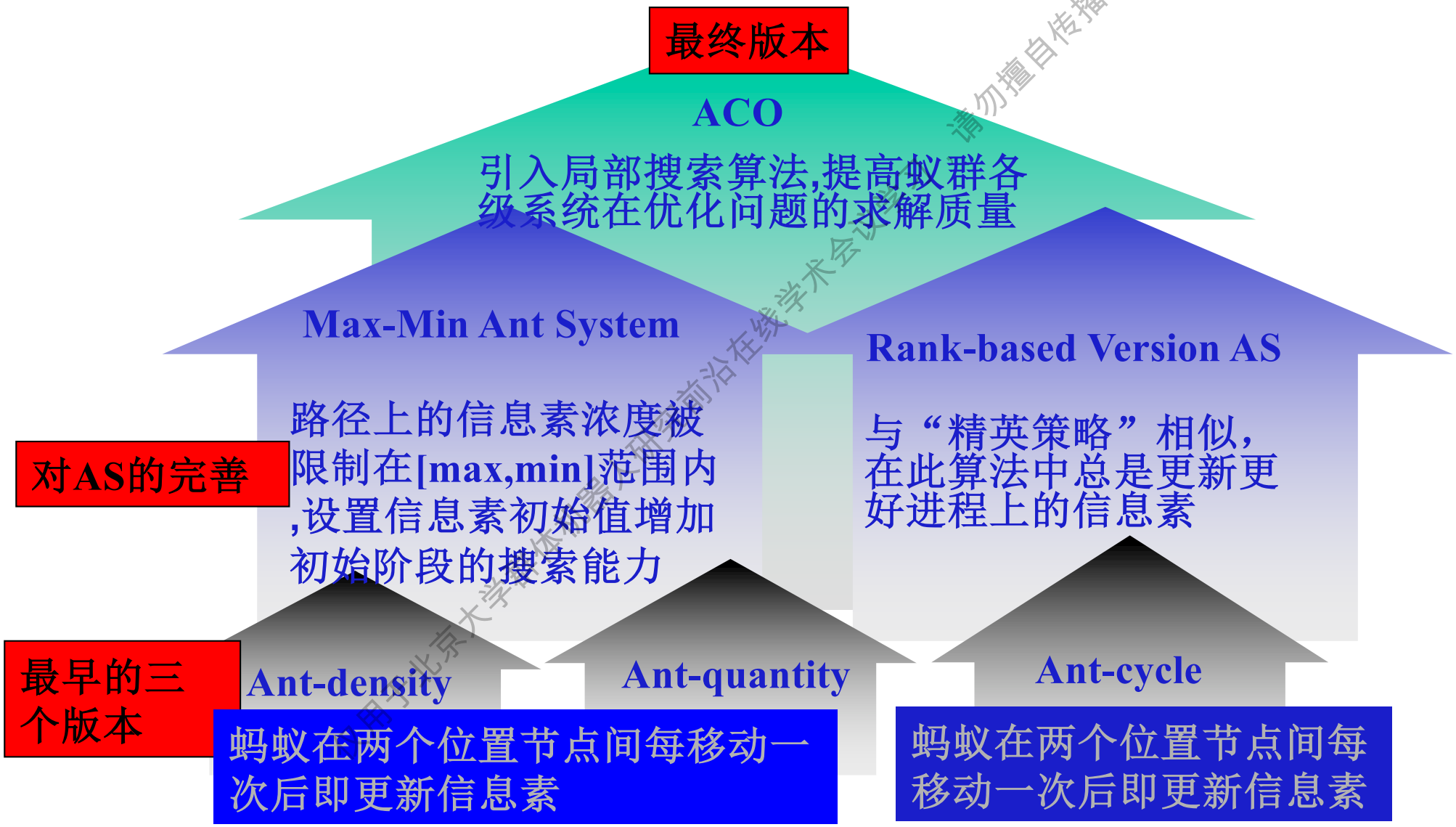
蚂蚁觅食时，在它走过的路上，留下随时间变化的外激素，这些外激素就象留下路标一样，留给后来蚂蚁一个寻路的标志。外激素越多，引诱蚂蚁走这条路径的能力就越强

蚂蚁选路关键在于两种规则

1. 状态转移规则，该规则反映蚂蚁选择当前路径的原则
2. 外激素更新规则，该规则表现在，蚂蚁在行走的路途中其分泌的外激素随着时间的推移而变化

因此，对于一个蚂蚁算法，其主要是进行两方面工作：一是蚂蚁依照其前方各路径上所存在的分泌物强度选择路径，即状态转移规则；二是对其所经过的路径上的分泌物强度进行调整，即外激素更新规则。总之，**路径选择准则和分泌物调整准则的合适选取是非常重要的**

蚂蚁算法的发展过程

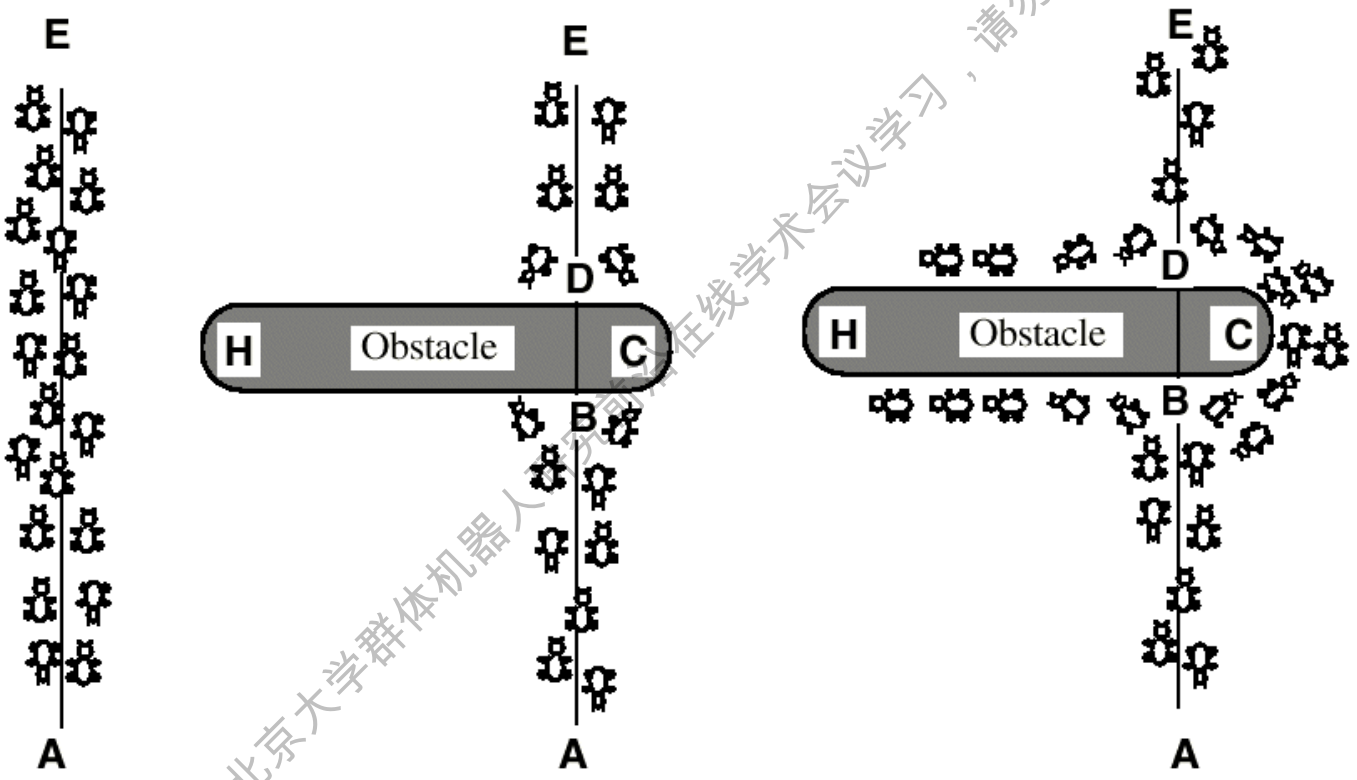


蚂蚁算法的发展过程

- 受蚂蚁行为的启发，Colorni和Dorigo等人于1991年提出Ant System(AS)的概念。AS算法最初包括ant-cycle、ant-density 和ant-quantity三种算法，由于实验结果表明另两种算法效果不如ant- cycle，因此，主要在算法ant-cycle的基础上产生了Ant System。研究成果经应用于传统的货郎担问题(Travelling Salesman Problem，TSP)上，取得了很好的优化效果
- 然而，蚁群算法存在搜索时间过长、易于停滞的问题。为了克服这些缺点，不少学者提出了改进算法。例如，1996 年，Gambardella 和 Dorigo 又提出一种修正的蚁群算法，他们称之为蚁群系统(ant colony system，ACS)。1997年德国学者Thomes stitzle等提出了改进的最大最小蚁群 (max-min ant system简称MMAS)算法，1999 年，吴庆洪等提出了具有变异特征的蚁群算法，意大利学者Fabio.Abbattista等提出了与遗传算法相结合的蚁群算法（genetic algorithm ant algorithm）
- M. Dorigo 等人先后提出了模拟蚁群这一觅食行为的AS(ant system)算法和ACS(ant colony system)算法,并将综合AS 算法和ACS 算法等一系列算法得到的最终改进的版本定义为ACO(ant colony optimization)算法

蚂蚁系统算法的原理

生物学家关于真实蚂蚁行为的研究启发了这个算法的形成



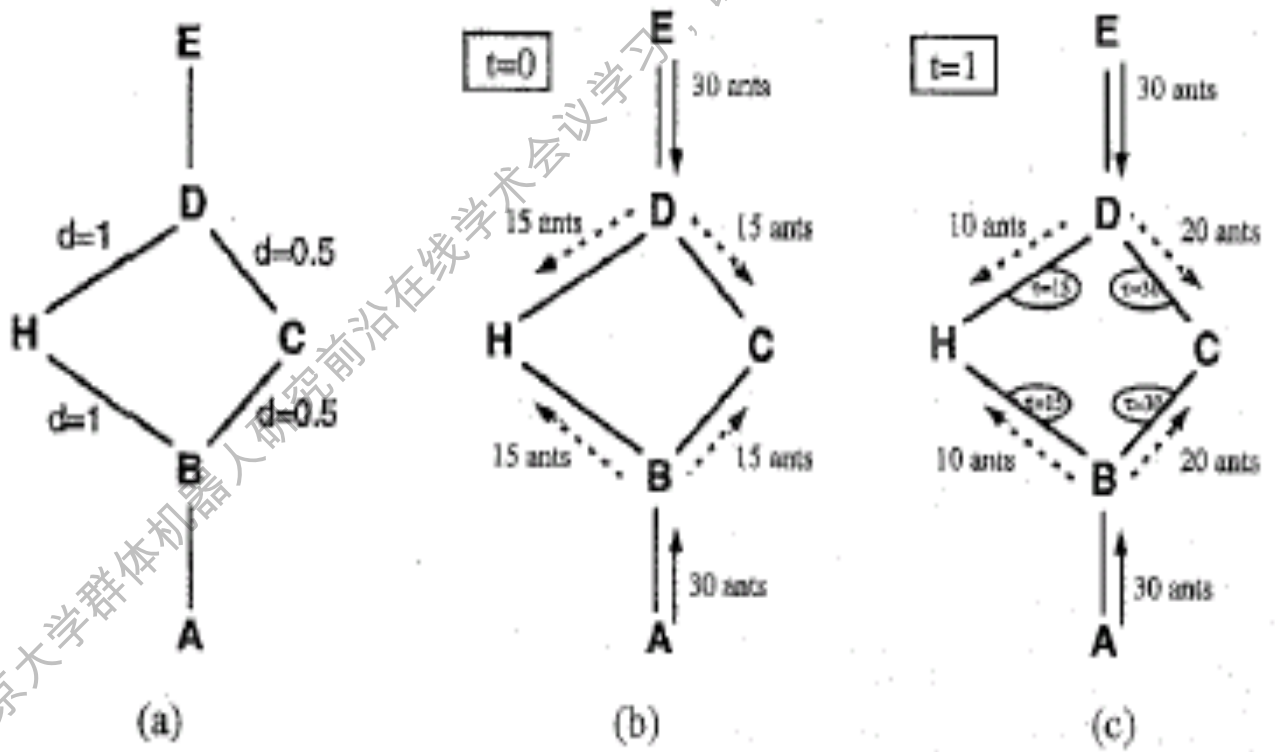
真实蚂蚁行为实验。(a) 蚂蚁沿着从食物源A到蚁巢E之间的路径行走；(b) 出现一个障碍物，蚂蚁以相同的概率决定选择路径BCD还是路径BHD；(c) 结果导致BCD这条相对来说较短的路径上的信息激素浓度比较高，从而大多数蚂蚁选择路径BCD

蚂蚁系统算法的原理

我们对真实蚂蚁行为的仿真并不感兴趣，我们的兴趣在于如何受真实蚂蚁种群的行为启发设计有效的优化算法

人工蚂蚁同真实的蚂蚁的不同点

- 人工蚂蚁存在记忆；
- 人工蚂蚁并不是完全看不见的；
- 人工蚂蚁生活在一个时间是离散的环境中



人工蚂蚁行为实例。(a) 初始的距离图；(b) 在时间t路径上不存在信息素，所以每个蚂蚁等概率的选择向左拐弯还是向右拐弯；(c) 在时间t=1，较短路径上的信息素浓度较大，因此，蚂蚁偏爱选择这条较短的路径

旅行商(TSP)问题

我们以求解著名的n个城市的旅行商(TSP)问题为例说明蚂蚁系统的数学模型。对于其他的组合优化问题，对此模型稍作修改便可应用。

给定n个城市及各城市间的距离，**旅行商问题可以描述为求一条经过各城市一次且仅一次的最短路线的问题**。在欧拉TSP问题的情况下，设d_{ij}为城市i与j间的欧拉距离，即弧(i,j)的长度为$\sqrt{(x_i-x_j)^2+(y_i-y_j)^2}$。其图论描述为：给定图$G=(N,E)$，其中$N$为城市集合，$E$为城市之间相互连接组成的边的集合，已知各城市间连接距离，要求确定一条长度最短的Hamilton回路，即遍历所有城市一次且仅一次的最短回路，即适当选择图上所有顶点的一个排列以组成最短路径。引入决策变量

$$x_{ij}=\begin{cases}1, & \text{若旅行商访问城市}i\text{后访问}j\\ 0, & \text{其它的情况下}\end{cases}$$

则TSP的目标函数为

$$\min Z=\sum_{i,j=1}^{n}x_{ij}d_{ij}.$$

蚂蚁系统的数学模型

每个蚂蚁都是具有如下的特征的简单智能体：

- 它根据某种概率选择走哪个城市，这个概率是城市距离和同它连接的路径的信息素的数量的函数；
- 为了使得蚂蚁能够完成合理的旅行，必须禁止蚂蚁旅行访问过的城市，这个通过一个禁忌表格来实现；
- 当蚂蚁完成了一次旅行，它就在走过的每条边(i,j)上释放适量的信息素

令$b_i(t)$ $(i=1,2,...,n)$是在时间t在城市i的蚂蚁的数目，令$m=\sum_{i=1}^{n} b_i(t)$是蚂蚁的总数。令$\tau_{ij}(t)$是时间t路径(i,j)上的信息素强度。每个蚂蚁在时间t选择下一个时间$t+1$要到达的城市，在时间间隔$(t,t+1)$内，对m个蚂蚁，我们调用蚂蚁系统迭代算法一次，算法的n次迭代叫做一圈（cycle），即每只蚂蚁完成了遍历所有城市一次的一次旅行。

蚂蚁系统的数学模型

在每只蚂蚁k构造出一个完整闭合路径并计算了相应长度之后（用L_k表示），路径上的信息素强度根据以下公式得到更新

$$\tau_{ij}(t+n)=\rho\times\tau_{ij}(t)+\Delta\tau_{ij},$$

其中，$\rho\,(0\le\rho\le1)$是一个常数，它表示信息素痕迹挥发后的剩余度，即轨迹的持久性，1-ρ表示在时间t和时间$t+n$内信息素的蒸发，并且：

$$\Delta\tau_{ij}=\sum_{k=1}^{m}\Delta\tau_{ij}^{k}$$

$\tau_{ij}(t)$表示路径(i,j)在时刻t的信息素轨迹强度，$\Delta\tau_{ij}^{k}$表示蚂蚁k在时间间隔$(t,t+n)$内在路径(i,j)上留下的单位长度的路径的信息素数量，其具体公式为

$$\Delta\tau_{ij}^{k}=\begin{cases}\dfrac{Q}{L_k}, & \text{如果第}k\text{个蚂蚁在间隔}(t,t+n)\text{内利用路径}(i,j)\\ 0, & \text{其它的情况}\end{cases}$$

其中，Q是一个常数，并且L_k表示每一个蚂蚁k旅行过的路径的总长度。

蚂蚁系统的数学模型

为了确保每只蚂蚁只访问每个节点一次以避免重复，每只蚂蚁都有一个禁忌表$tabu_k$，用来存储蚂蚁当前已经访问过的节点。利用禁忌表使蚂蚁到这些点的转移概率为0，也就是禁止蚂蚁到这些节点。当一次旅行完成后，利用禁忌表来计算问题的现在的解。然后清空禁忌表，蚂蚁就可以进行自由的重新选择路径了。$tabu_k(s)$是禁忌表中的第s个元素，表示在现在的一次旅行中k个蚂蚁访问的第s个城市。

称$\eta_{ij}=1/d_{ij}$为路径(i,j)的能见度，定义对于蚂蚁k来说，从城市i到城市j的转移概率为

$$p_{ij}^{k}(t)=\begin{cases}\dfrac{(\tau_{ij}(t))^{\alpha}(\eta_{ij})^{\beta}}{\sum_{k\in allowed_k}(\tau_{ik}(t))^{\alpha}(\eta_{ik})^{\beta}} & \text{如果} j\in allowed_k \\ 0 & \text{其它}\end{cases}$$

其中$allowed_k=\{N\text{-}tabu_k\}$，其中α和β是控制路径和能见度相对重要性的参数。若(i,j)之间的距离短，则η_{ij}较大，p_{ij}也较大。也就是说，距离较近的城市以较大的概率被选择。

蚂蚁系统的分类

蚂蚁系统算法最核心的部分是对痕迹强度的处理

若某些支路上痕迹较浓，就更有可能被蚂蚁选中。蚂蚁系统算法根据痕迹更新方式的不同可以分为蚁密算法 (ant-density algorithm)、蚁量算法 (ant-quantity algorithm)和蚁周算法(ant-cycle algorithm)

（1）蚁密算法

$$\Delta\tau_{ij}^{k}=\begin{cases}Q & \text{若在时刻}t\text{和}t+1\text{之间第}k\text{蚂蚁使用边}(i,j)\\ 0 & \text{其它}\end{cases}$$

其中，Q表示一只蚂蚁每次经过边(i,j)时放在边(i,j)上的信息素

（2）蚁量算法

$$\Delta\tau_{ij}^{k}=\begin{cases}\dfrac{Q}{d_{ij}} & \text{若第}k\text{只蚂蚁在时间}t\text{和}t+1\text{之间路过边}(i,j)\\ 0 & \text{其它}\end{cases}$$

其中，Q/d_{ij}表示一只蚂蚁每次经过边(i,j)时放在边(i,j)上的信息素

蚂蚁系统的分类

蚂蚁系统算法最核心的部分是对痕迹强度的处理

若某些支路上痕迹较浓，就更有可能被蚂蚁选中。蚂蚁系统算法根据痕迹更新方式的不同可以分为蚁密算法 (ant-density algorithm)、蚁量算法(ant-quantity algorithm)和蚁周算法(ant-cycle algorithm)

（3）蚁周算法

$$\Delta\tau_{ij}^{k}=\begin{cases}\dfrac{Q}{L_k}, & \text{如果第}k\text{个蚂蚁在间隔}(t,t+n)\text{内利用路径}(i,j)\\ 0, & \text{其它的情况}\end{cases}$$

上述三种模型，由于蚁周算法在搜索过程中使用的反馈信息是全局的，而蚁量和蚁密使用的反馈信息都是局部的，故**蚁周算法要优于另两种算法**，仿真的结果也证明了这个推断

蚂蚁系统算法的简单表述

在0时刻进行初始化过程，蚂蚁分别被放置在不同的城市，每一条边都有一个初始的外激素强度值$\tau_{ij}(0)$。每一只蚂蚁的禁忌表的第一个元素置为它的开始城市。然后，每一个蚂蚁从城市i移动到城市j，蚂蚁依据两城市之间的转移概率函数选择移动城市。在n次循环后，所有蚂蚁都完成了一次周游，它们的禁忌表将满；在这时，计算每一个蚂蚁k旅行过的路径的总长度L_k,$\Delta\tau_{ij}^k$ 依据轨迹强度的更新方程更新。而且同时，由蚂蚁找到的最短路径(即 $\min L_k$, k=1,2,...,m) 将被保存，所有禁忌表将被清空。这一过程重复直到周游计数器达到最大(用户定义) 周游数$NC_{\max}$，或者所有蚂蚁都走同一路线

蚂蚁系统算法的步骤过程

我们以蚁周算法为例详细介绍蚂蚁系统算法

1. 初始化

设置 t:=0　　{t是时间计数器}

设置 NC:=0　　{NC是周游计数器}

对每个边(i,j)，设置信息素强度初始值$\tau_{ij}(t)=c$和$\Delta\tau_{ij}=0$

在n个节点上放置m个蚂蚁

2. 禁忌表

设置 s:=1　　{s是禁忌表索引}

For k:=1 to m do

把第k个蚂蚁的开始城市放在禁忌表$tabu_k(s)$中

3. Repeat until 禁忌表满了　{这个步骤将重复n-1次}

设置 s:=s+1

For k:=1 to m do

以概率 $p_{ij}^k(t)$ 选择一个城市 j　　{在时间t蚂蚁k在城市$i=tabu_k(s-1)$}

将第k个蚂蚁移动到城市j

在禁忌表$tabu_k(s)$中写入城市j

蚂蚁系统算法的步骤过程

我们以蚁周算法为例详细介绍蚂蚁系统算法

4. For k:=1 to m do

将第k个蚂蚁从$tabu_k(n)$移动到$tabu_k(1)$

计算第k个蚂蚁这次周游的总长度L_k

更新找到的最短路径

对于每一个边(i,j)

For k:=1 to m do

$$\Delta\tau_{ij}^{k}=\begin{cases}\dfrac{Q}{L_k}, & \text{如果第}k\text{个蚂蚁在间隔}(t,t+n)\text{内利用路径}(i,j)\\ 0, & \text{其它的情况}\end{cases}$$

$$\Delta\tau_{ij}:=\Delta\tau_{ij}+\Delta\tau_{ij}^{k}$$

蚂蚁系统算法的步骤过程

我们以蚁周算法为例详细介绍蚂蚁系统算法

5. For 每条边(i,j) 计算$\tau_{ij}(t+n)$ 根据方程$\tau_{ij}(t+n)=\rho\times\tau_{ij}(t)+\Delta\tau_{ij}$

设置 $t:=t+n$

设置 $NC:=NC+1$

对于每条边(i,j)设置$\Delta\tau_{ij}:=0$

6. 如果 $NC<NC_{max}$ 并且 没有停滞行为

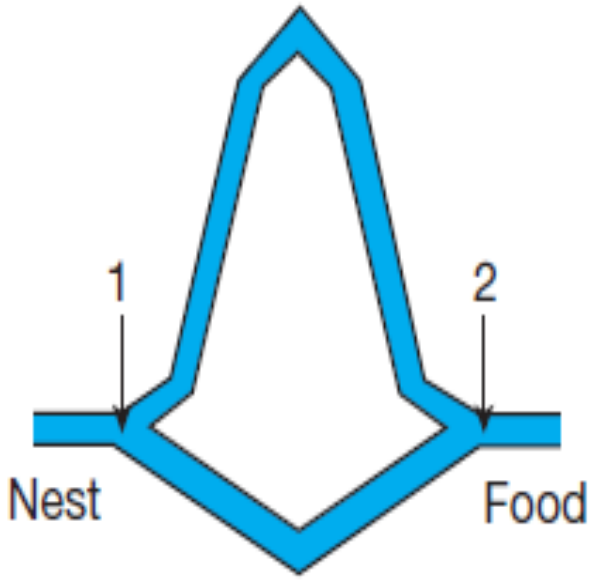
那么 清空所有禁忌表

返回步骤2

否则 就打印出最短路径

程序终止

蚂蚁种群优化算法的数学模型



目标函数：

$$\min Z = \sum_{i,j=1}^{n} X_{ij} d_{ij};$$

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2};$$

$$X_{ij} = \begin{cases} 1 & \text{if an ant visits } j \text{ after visiting } i \\ 0 & \text{else} \end{cases}$$

ACO算法的模型：

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

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{Q}{L_k} & \text{if ant } k \text{ used route } (i,j), \\ 0 & \text{else.} \end{cases}$$

$$p_{ij}^{k}(t) = \begin{cases} \frac{(\tau_{ij}(t))^{\alpha}(\eta_{ij})^{\beta}}{\sum_{k \in allowed_k} (\tau_{ik}(t))^{\alpha}(\eta_{ik})^{\beta}} & \text{if } j \in allowed_k, \\ 0 & \text{else.} \end{cases}$$

$$\eta_{ij} = \frac{1}{d_{ij}}.$$

Dorigo M. et al., Nature 406, 39-42 (2000).

精英策略(elitist strategy)

精英策略(elitist strategy)是应用在算法的信息素更新方面的，首先由Dorigo 等人提出，以强化精英蚂蚁(发现迄今最好路径的蚂蚁) 的影响。算法记录历史上最好的解，每次更新信息素时，不是针对本次循环得到的最好的解做信息素的增加，而是更新全局最优解所包含的路径。精英策略方法应用于TSP 算法的更新规则是

$$\tau_{ij}(t+1)=(1-\rho)\times\tau_{ij}(t)+\Delta\tau_{ij}(t)+\Delta\tau_{ij}^{*}(t),$$

$$\Delta\tau_{ij}(t)=\sum_{k=1}^{\text{蚂蚁数}}\frac{Q}{L_k} \qquad \text{如果} arc(i,j)\in \text{Tour}-\text{of}-\text{Ant}_m$$

$$\Delta\tau_{ij}^{*}(t)=\begin{cases}\dfrac{k}{L(t)} & \text{如果} arc(i,j)\in \text{Tour}^{\text{global-best}}\\ 0 & \text{否则}\end{cases}$$

其中，Q是一个固定的正数，并且L_k表示每一个蚂蚁k旅行过的路径的总长度

如果连接点i、j的边是全局最优解的一部分的话, 在每次更新信息素时就对这条边增加信息素。增加的量与全局最优解的大小有关。k是一个正数，$L(t)$是所找出的最优解的路径长度。ρ的取值越大，存储在信息素中的有用知识会很快地丢失掉，ρ偏小，则算法不容易忘记知识。

1996年，Gambardella和Dorigo提出一种修正的蚁群算法，称为蚁群系统(ant colony system，ACS)。该算法对AS算法的选路和信息更新策略作了相应的改进。

(1) 采用伪随机比率选择规则(pseudo - random - proportional action choice rule) 的选路方式，即对于在城市i的蚂蚁，按如下公式选择下一个城市j:

$$j=\begin{cases}\arg\max\{\tau(i,u)\cdot\eta^{\beta}(i,u)\} & \text{如果} q\le q_0\\ S & \text{否则}\end{cases}$$

其中$q_0\in(0,1)$为常数，$q\in(0,1)$为随机生成的数。$\tau(i,u)$表示城市i与城市u之间的信息量，$\eta(i,u)$表示城市i与城市u之间的启发式因子，β表示启发式因子的相对强弱。在选择下一个城市之前随机生成q，如果$q\le q_0$，则从城市i到所有可行的城市中找出最大的城市，即为下一个要选择的城市，这称为利用已知信息(exploitation)，这是非随机的方法(其余参数含义同前)；如果$q>q_0$，则按P117的公式来选择下一个城市，称为搜索(exploration)。

(2) 局部信息更新。蚂蚁从城市i转移到城市j后，路径(i,j)上的信息量按如下公式进行更新：

$$\tau_{ij}(t+1)=(1-\xi)\times\tau_{ij}(t)+\xi\cdot\tau_0$$

其中τ_0为常数，$\xi\in(0,1)$为可调参数。

(3) 全局信息更新。对全局最优解所属的边，按如下公式进行信息更新：

$$\tau_{ij}(t+1)=(1-\rho)\times\tau_{ij}(t)+\rho\cdot\Delta\tau_{ij}^{gb}(t),$$

$$\Delta\tau_{ij}^{gb}=\frac{1}{L^{gb}}$$

其中L^{gb}为当前全局最优解的长度，$\rho\in(0,1)$为外激素的蒸发系数。

综上所述，ACS算法和以前AS算法的主要不同在于，蚂蚁选择下一城市之前，多进行了一次随机试验，将选择情况分成“利用已知信息”和“探索”两类。

最大最小蚂蚁系统(MMAS)算法

德国学者Thomas Stützle与Holger Hoos提出了一种改进的蚁群算法，称为最大最小蚁群系统(MAX-MIN ant system，MMAS)，也是一种较好的通用优化算法。MMAS算法在防止算法过早进入停止状态和解决大规模TSP问题上都有较大改进。其基本思想是对路径上的信息素进行限制，以期克服停滞问题，并且仅让每一代中最好的个体所走的路径上的信息作调整，从而加快收敛速度。

MMAS算法对基本蚂蚁算法(AS)进行了三点改进：

（1）为了更加充分地进行寻优，各路径信息素初值设为最大值τ_{max}；

（2）一圈中只有最短路径的蚂蚁才进行信息素修改增加，这与AS 蚂蚁圈模型调整方法相似；

（3）为了避免算法过早收敛非全局最优解，将各路经的信息素浓度限制在于$[\tau_{min},\tau_{max}]$之间，超出这个范围的值被强制设为τ_{min}或者τ_{max}

从实验结果看，MMAS算法在防止算法过早停滞及有效性方面对AS算法有较大的改进。

蚂蚁种群优化算法的应用

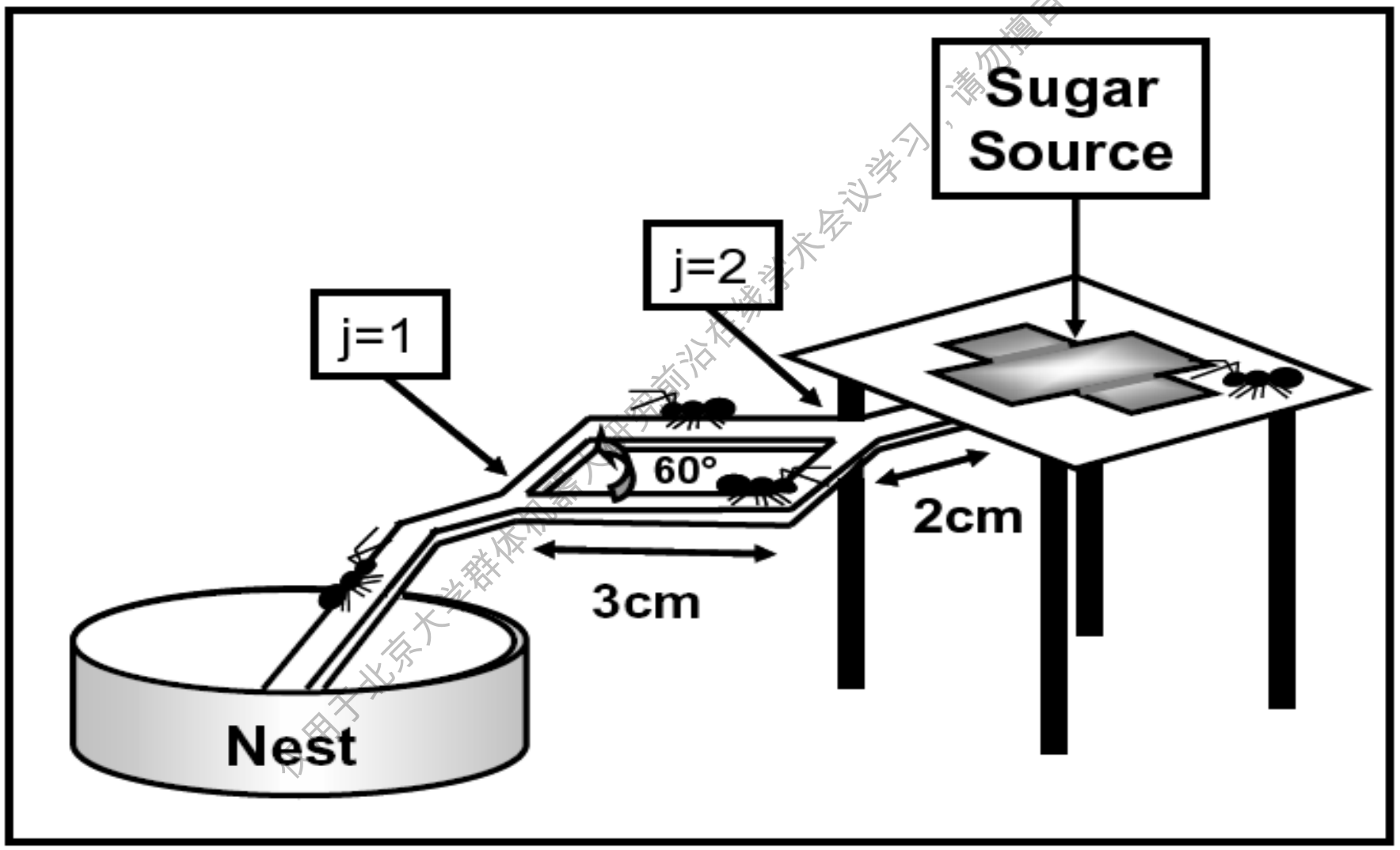
- ACO算法现已被用于广泛求解多种工程问题：
  - 旅行商问题(TSP)
  - 二次分配问题(QAP)
  - 有序排列问题(SOP)
  - 车间作业调度问题(JSSP)
  - 资源受限的工程调度问题(RCPSP)
  - 热电分配问题(CHP)
  - 通信网络的动态路由选择问题
  - 移动通信中频率分配
  - 电力系统中故障点的估计

蚂蚁最优交通模型

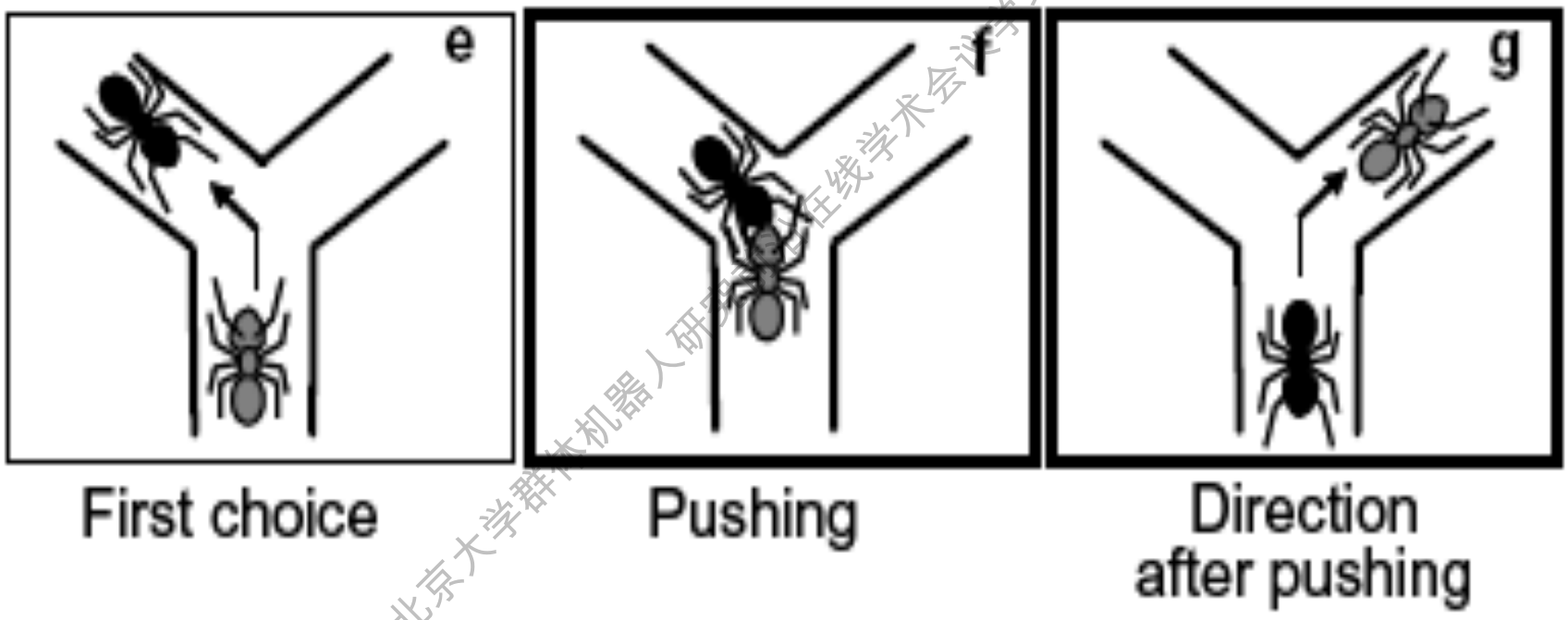
- 等路径窄桥蚂蚁觅食实验
- 不同桥宽的流量实验测试结果
- 蚂蚁退搡转向过程

Audrey Dussutour
于2004年提出
（Nature）

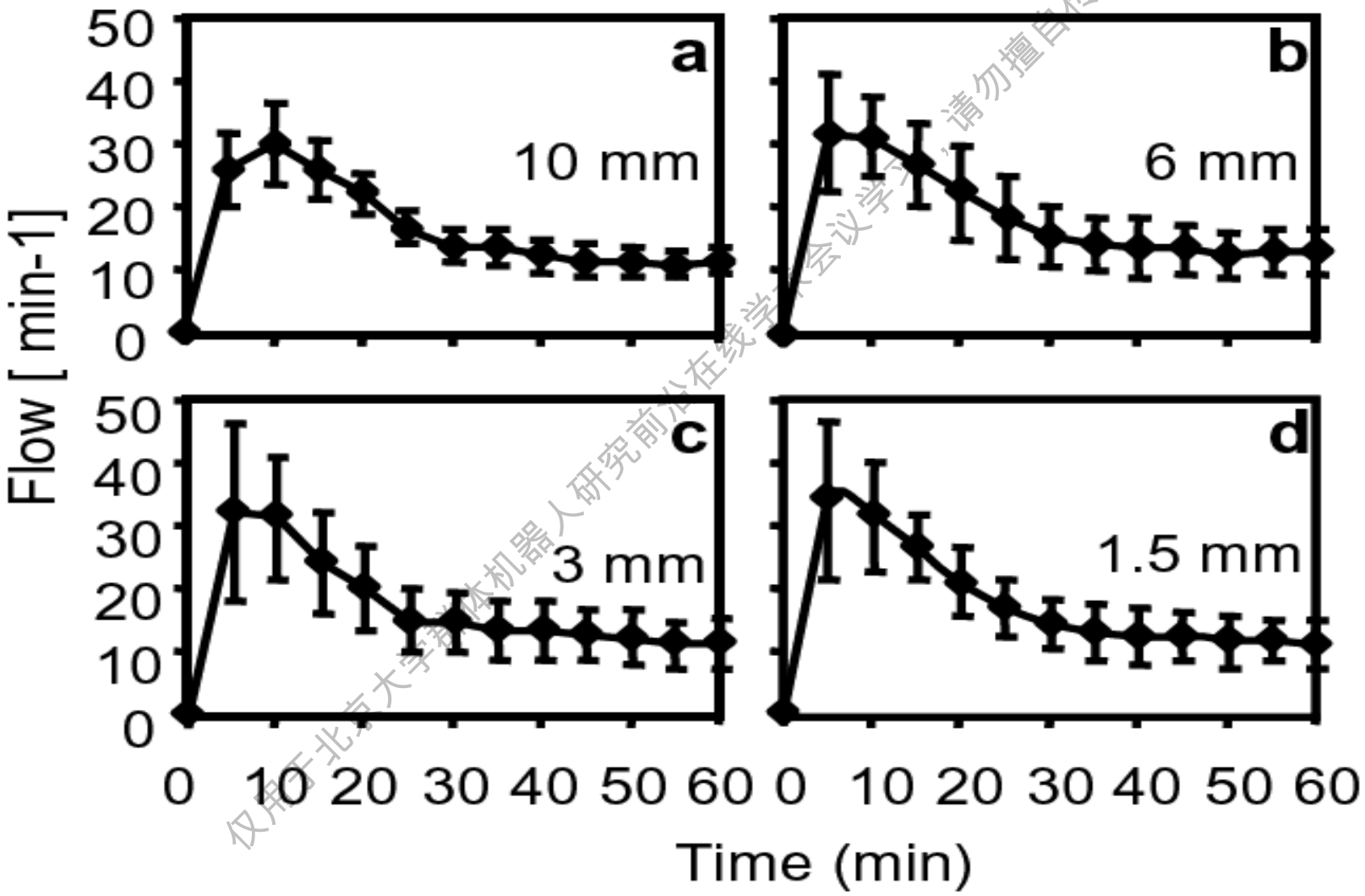
等路径蚂蚁觅食实验



蚂蚁的推搡转向过程



不同桥宽的流量测试



实验结果分析

- 当每个通道只有10mm宽时，双向的交通却仍偏重在某通道上，显示蚂蚁偏好信息素大的道路，在10mm宽的桥上不会产生推搡现象；
- 当通道宽≤6mm时，两条通道的交通量却变得一致。这是因为刚从巢中外出的蚂蚁在撞上返巢蚂蚁时，会被推搡从而被迫选择另一条通道，这种行为实际上最后会保证一种窄道对称的交通秩序（两个单行道路的出现）；
- 在窄道上这种推搡的比例大很多，而且是越窄的地方(1.5-6mm)推搡的比例越大，最后形成最优化交通秩序。

Dussutour等建立路径信息素集合C_{ij}动力学方程：

$$\frac{dC_{ij}}{dt} = q\Phi_{ij}(t) + q\Phi_{ij'}(t-\tau) - vC_{ij}(t), j' = 3 - j$$

其中：$i = 1,2$ 表示分支，$j = 1,2$ 表示选择节点。

$\Phi_{i1}(t)$ **代表从巢穴到食物源选择1节点第i 个分支**

$\Phi_{i2}(t)$ **代表从食物源到巢穴选择2节点第i 个分支，当另一个选择点为j'=3-j =2**

τ **表示选择从一个节点到另一个节点所要求的平均时间**

q **表示残留在路径上的信息素量**

v **表示信息素的衰退率**

当推搡发生时，选择节点 *j*, 分支 *i* 的蚂蚁流量可表示为：

$$\Phi_{ij}(t)=\underbrace{\phi_j(t)F_{ij}(t)\left[1-\gamma\alpha\Phi_{ij}(t-\tau)/w\right]}_{\text{从}i\text{分支推到}i'\text{分支后剩余的蚂蚁流量}}+\underbrace{\phi_j(t)F_{i'j}(t)\gamma\alpha\Phi_{i'j'}(t-\tau)/w}_{\text{从}i'\text{分支推到}i\text{分支的蚂蚁流量}}$$

$\phi_j(t)F_{ij}(t)$ **表示在第*i*个分支上，从第*i*个分支推到*i'*分支减少后的蚂蚁流量。**

$\alpha\Phi_{ij'}(t-\tau)/w$ **表示相互作用减速的百分比，*a*为相互作用时间及蚂蚁宽度(常数)。**

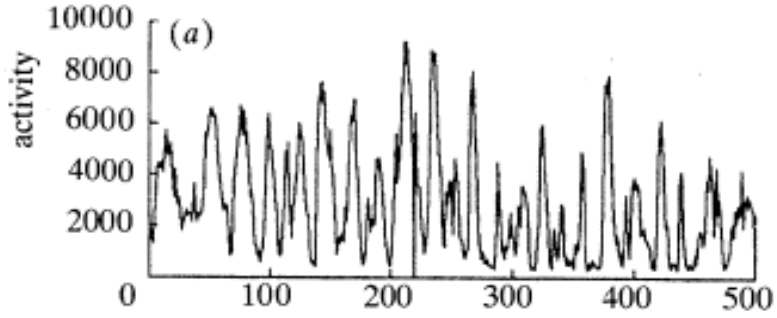
$\gamma=0.57$

蚁群最优交通模型潜在的应用

- 探索生态系统演化机理
- 处理网络大流量的信息拥塞问题
- 建设分布式智能交通
- 构建现代物流产业平台
- 实现多渠道营销

蚂蚁搜索呈现混沌行为

Cole B.J., Proc R Soc Lond 244, 253-259 (1991).



蚂蚁种群：周期为15-37分钟的周期行为为；

单个蚂蚁：维数为2.4的低维决定性混沌行为

Solé R.V., J Theoret Biol 161, 343-357 (1993).

单个蚂蚁： $x_{n+1} = x_n e^{\mu(1-x_n)}.$

Nemes L. et al., IEEE Trans Circ Syst-I 42, 741-745 (1995).

周期性震荡：

$$x_{i,t+1} = (1-h)x_{i,t} + h\sum_{(r)} a(y_{i,t}, y_{(r),t})y_{(r),t};$$

$$y_{i,t} = \phi(x_{i,t}).$$

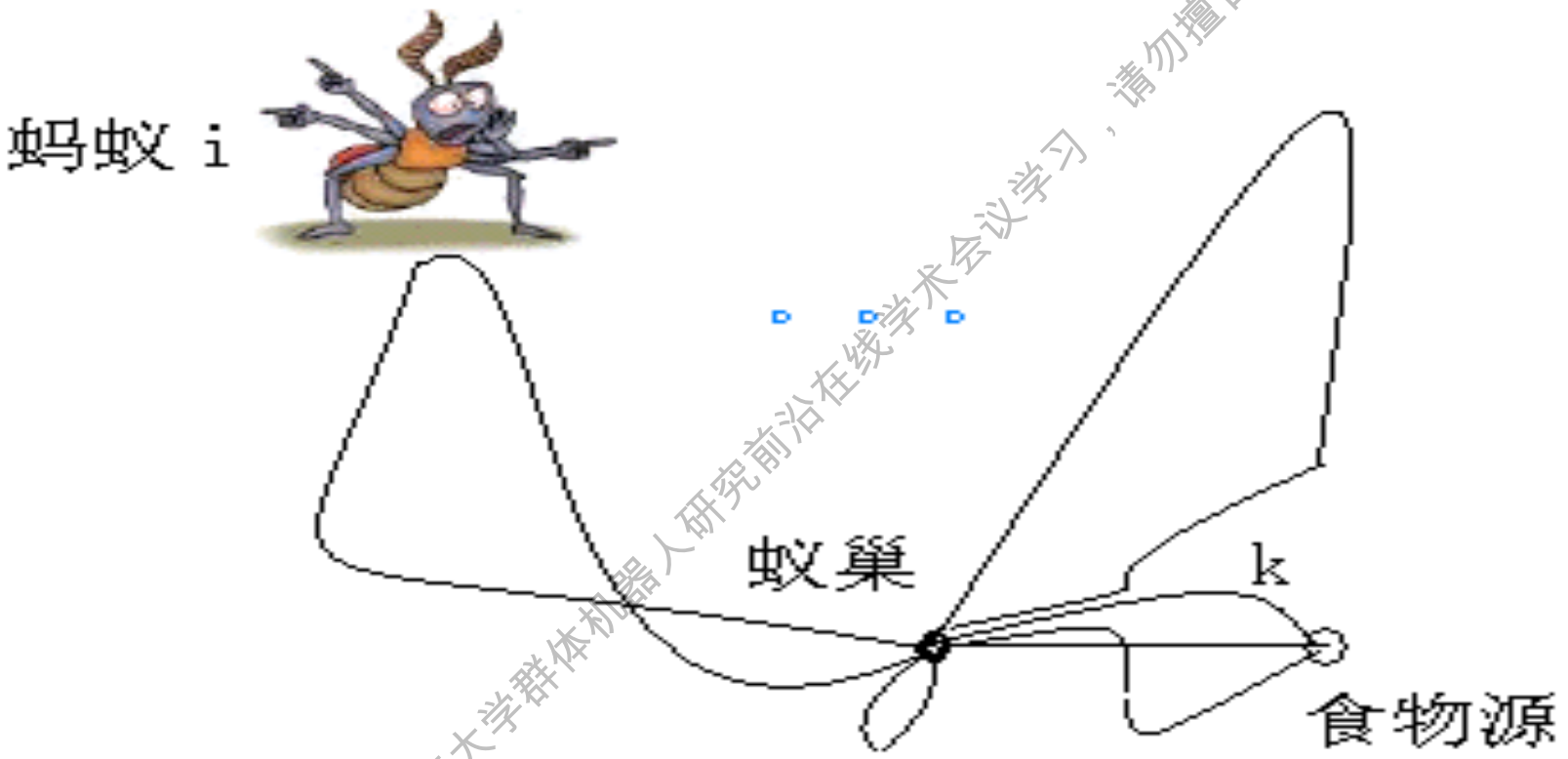
混沌优化方法

- Hopfield neural network (1986)
- Chaotic neural network (Aihara, 1990)
- Chaotic simulated annealing (Chen and Aihara,1995)
- Chaotic tabu search (Hasegawa, 2002)
- Chaos artificial immune algorithm (Xing & Shing, 2003)
- …… …

混沌蚂蚁优化思想的提出

- Dorigo基于蚁群的最短路径选择试验建立起了著名的蚁群优化模型ACO, ACO算法是基于概率搜索的算法；
- 但Cole等生物专家观测到单个蚂蚁的低维混沌行为以及整个种群的周期性动力学行为；
- 从动力学的角度来说,单个蚂蚁的混沌行为和种群强大的自组织能力以及蚁群建立起的最短路径之间必然存在着某种内在的关系；
- 现在的问题是单个蚂蚁的混沌行为与蚁群能找到最佳食物路径之间的内在关系是什么？它们是如何组织和觅食的？

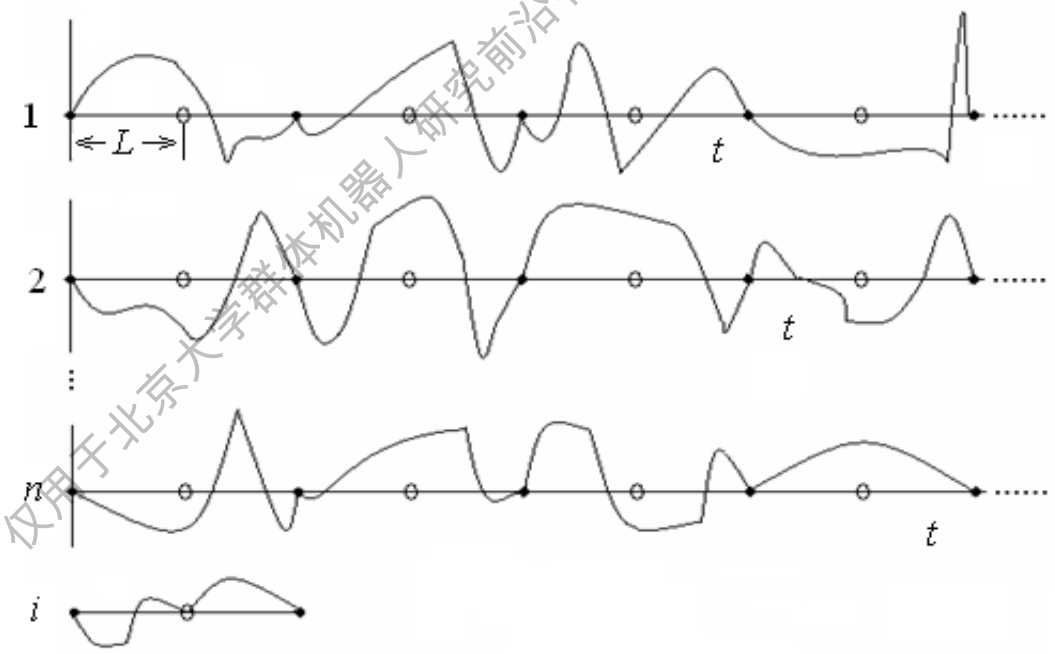
蚂蚁外出游荡捕食



蚂蚁外出捕食，每次捕食后回巢的示意图

Hunting strategy

- **Scout ants: the first phase of foraging**
- **Search for an unknown point/region**
- **Initially, no pheromone**
- **Freely chaotic crawling**
- **many ants conduct a parallel search**

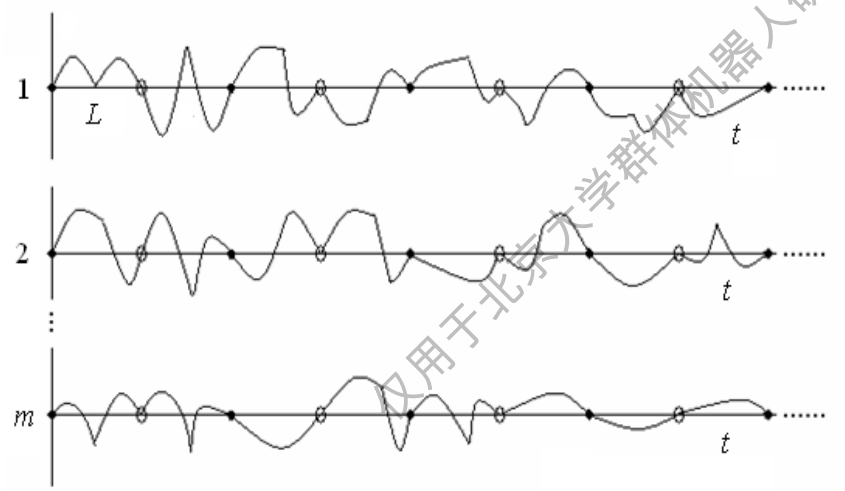


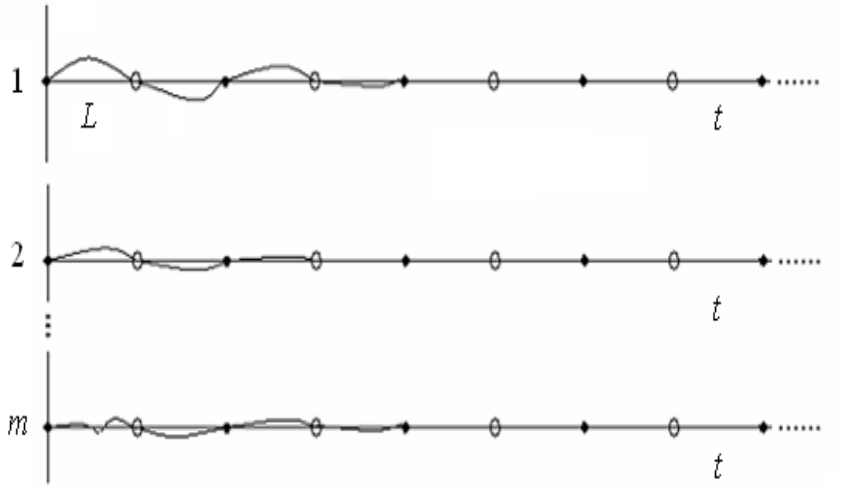
Homing strategy

- to return to the nest for the energy supply
- a special region that ants are familiar with
- different with the searching of food
- use chaos in combination with their knowledge
- the starting time of homing
- the tiring time

Path building strategy

- recruited ants
- initially, chaotic walking dominates their behaviors: global search
- pheromone: to direct the crawling of recruited ants
- pheromone field (pf): be the cause of self-organization of ants; to cause the ants to conduct a local search
- gradually, the impact of pf: stronger and stronger
- finally, the status of ant: from chaotic walking to periodicity





混沌蚂蚁捕食(chaotic ant foraging)模型

Generally: $\vec{Z}_i(t) = g(\vec{Z}_i(t-1), \vec{P}_{food}, \vec{P}_{nest}, y_i(t), r_i)$ （1）

Detailed: $y_i(t) = y_i(t-1)^{(1+r_i)}$ （2）

$$\begin{aligned} Z_{ik}(t) = (Z_{ik}(t-1) + V_k)e^{(1-e^{-ay_i(t)})(3-\Psi_k(Z_{ik}(t-1)+V_k))} \\ -V_k + e^{-2ay_i(t)+b}(|\sin(\omega t)|(P_{foodk} - P_{nestk}) \\ -(Z_{ik}(t-1) - P_{nestk})), \end{aligned} \quad （3）$$

Explanations:

Ants can center around the nest to search $V_k = 7.5/(2\Psi_k) - P_{nestk}$

To achieve periodic oscillation behavior

$$|sin(\omega t)|(P_{foodk} - P_{nestk}) - (Z_{ik}(t-1) - P_{nestk})$$

Initially, r_i=0, Eq. (3) becomes $Z_{ik}(t) = (Z_{ik}(t-1) + V_{ik})e^{(3-\Psi_k(Z_{ik}(t-1)+V_k))} - V_k$

Finally, y_i near zero, Eq. (3) becomes

$$Z_{ik}(t) = Z_{ik}(t-1) + e^b(|sin(\omega t)|(P_{foodk} - P_{nestk}) - (Z_{ik}(t-1) - P_{nestk}))$$

混沌蚂蚁(chaotic ant)优化模型

Model of chaotic ant:

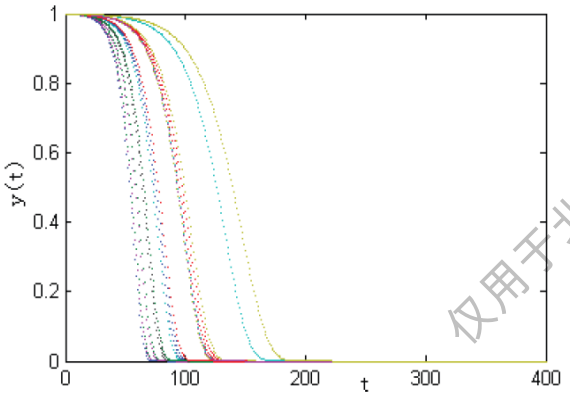
$$y_i(t) = y_i(t-1)^{(1+r_i)}$$

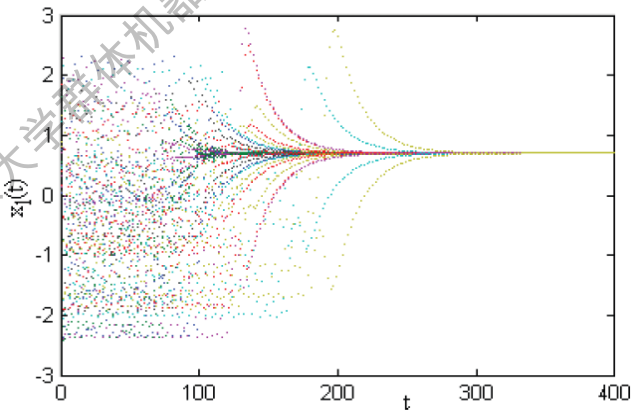
$$z_{id}(t) = \left(z_{id}(t-1) + \frac{7.5}{\psi_d} \times V_i\right) \times \exp\left((1-\exp(-ay_i(t))) \times \left(3 - \psi_d\left(z_{id}(t-1) + \frac{7.5}{\psi_d} \times V_i\right)\right)\right) - \frac{7.5}{\psi_d} \times V_i + \exp(-2ay_i(t) + b) \times (p_{id}(t-1) - z_{id}(t-1)),$$

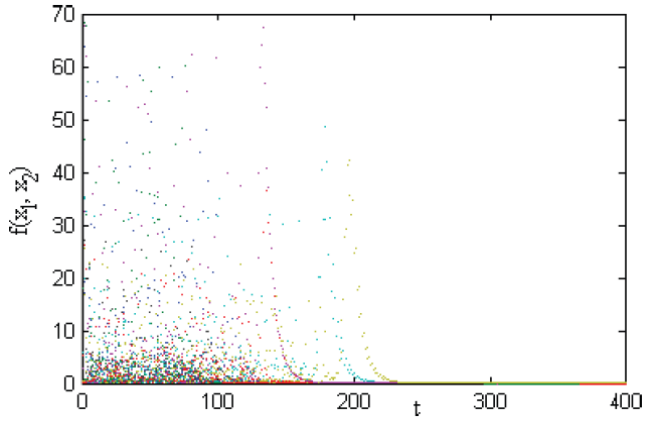
Li L. et al., J Bifur Chaos 16, 2351-2364 (2006).

Objective function:

$$f(x_1, x_2) = (x_1 - 0.7)^2((x_2 + 0.6)^2 + 0.1) + (x_2 - 0.5)^2((x_1 + 0.4)^2 + 0.15)$$







- 为了检验新算法的性能，进行了大量的测试工作，所有这些测试结果都表明混沌蚁群优化算法可以很好地进行优化搜索：
  1. 用多模态函数进行算法性能测试，并和免疫系统优化算法进行了比较；
  2. 采用标准的高维测试函数进行了测试研究，并且在相同条件下与粒子群和凯尔曼群的算法性能测试的结果相比较；
  3. 利用有约束的测试函数进行了测试。

混沌蚁群模型的应用

- **Parameter estimation of dynamical systems:** Peng H. et al., Phys Rev E 81, 016207 (2010); Li L. et al., Chaos Solitons Fractals 40, 1399-1407 (2009); Tang Y. et al., Chaos Solitons Fractals 41, 2097-2102 (2009).
- **Optimization of power system:** Cai J. et al., Electric Power Systems Research 77, 1373-1380 (2007); Cai J. et al., Electrical Power and Energy Systems 32, 337-344 (2010); Cai J. et al., Electrical Power and Energy Systems 34, 154-160 (2012); Cai J. et al., Energy 38, 346-353 (2012).
- **Web user clustering:** Wan M. et al., Nonlinear Dynamics 61, 347-361 (2010); Wan M. et al., Knowledge and Information Systems, DOI:10.1007/s10115-011-0453-x; Wan M. et al., Applied Soft Computing, published online.
- **PID control:** Zhu H. et al., Chaos Solitons Fractals 42, 792-800 (2009); Tang Y. et al., Expert Systems with Applications 39, 6887-6896 (2012).
- **Fuzzy system identification:** Li L. et al., NeuroQuantology 6, 379-386 (2008); Li L. et al., Chaos Solitons Fractals 41, 401-409 (2009).
- **Combinatorial optimization:** Wei Z., Nonlinear Dynamics 65, 271-278 (2011); Ge F. et al., IEEE Int Conf on Intel Comput and Intel syst 1, 512-516 (2010).
- **Improved chaotic ant:** Li Y. et al., Chaos Solitons Fractals 42, 880-889 (2009); Ge F. et al., Int J Bifur Chaos 21, 2597-2622 (2011).

内容提要

- 粒子群优化算法
- 蚂蚁种群优化算法
- **细菌觅食算法**
- 人工免疫系统算法

细菌觅食算法(Bacterial Foraging Algorithm)

- 2002年，K. M. Passino提出一种新型仿生类算法
- 模仿E. coli大肠杆菌在人体肠道内吞噬食物的行为
- 也叫做Bacterial Foraging Optimization，BFO
- 在BFO算法中，一个细菌代表一个解，它在寻找最优解时只依靠自己
- 由于其简单、高效的特点，同时具有群体智能算法并行搜索、易跳出局部极小值等优点，在许多工程和科学领域得到了广泛的应用
- 然而，在处理更复杂的优化问题，特别是高维多模态问题时，与其他群体智能优化算法相比，BFO算法的收敛性较差
- 由于它的生物学驱动方式和神奇优美的结构，研究人员正在尝试着去混合BFOA算法与其他不同的算法，尽量去探索该算法的局部和全局两个方面的特性

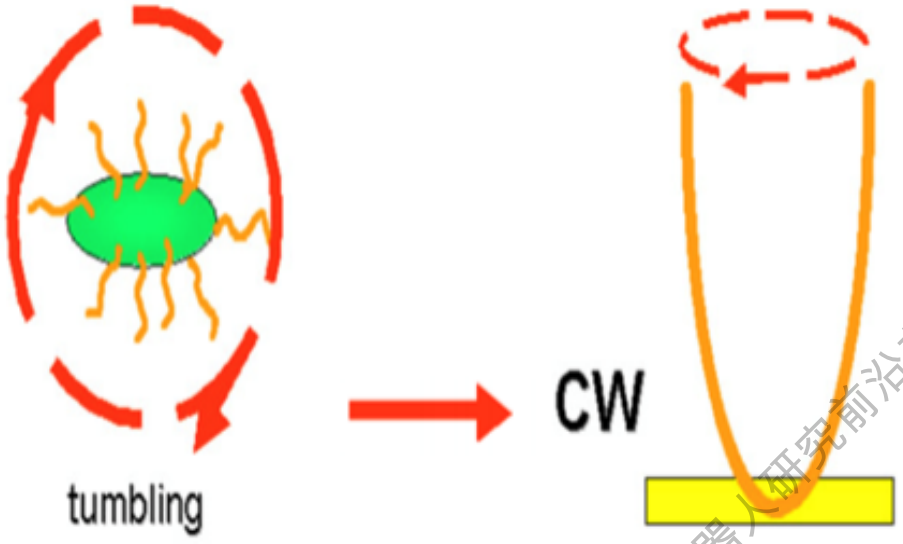
K. Passino, "Biomimicry of bacterial foraging for distributed optimization and control," IEEE Control Systems Magazine, 2002.

细菌趋化性的运动方式

- 细菌趋化性是指有运动能力的细菌对物质化学浓度梯度作出的反应，使细菌趋向有益刺激，逃避有害刺激
- Engelmann和Pfeffer发现细菌的运动不是任意的，而是定向移动
- 直到1960年，Alder深入研究了细菌趋化性的分子机制，提出大肠杆菌(Escherichia coli)对氨基酸以及糖的趋化性是由位于细胞表面的受体蛋白调节的，并由细胞内分子传递信号最终影响细菌的运动。
- 以大肠杆菌为例，一个细胞有4~10根鞭毛，鞭毛快速旋转使得细胞具有运动的能力。
- 鞭毛的运动方式分为 2 种：顺时针旋转、逆时针旋转。

顺时针旋转(clockwise，CW)

按照本轮计算的优化方向进行迭代调整



当鞭毛顺时针旋转时，细胞鞭毛分开，原地做翻转运动，来调整运动方向

CW过程我们可以理解为LMS(最小均方算法)过程中计算最速下降梯度方向，但是生物机制没法一步得到最速梯度方向，而且我们知道液体中转动物体也存在粘滞惯性，因此，细菌在CW过程中实际上是在计算一个大致正确的梯度下降方向。它并不是一次得到最优结果，而是依靠多次地不断迭代逐渐靠近最优

逆时针旋转(counter-clockwise， CCW)

（1）在存在局部浓度梯度变化的环境中调整方式 - GD(梯度下降)过程

- 当鞭毛逆时针旋转时，鞭毛拧成一束，产生向前的推动力
- CCW过程中前进的这段，我们可以理解为梯度下降中，每次调整的总距离，即 $\eta x(n)e(n)$
- 但是细菌并不是说一次CW调整方向后，就一直保持前进
- 细菌先直线运动一段距离 然后通过比较现有浓度和过去浓度（计算浓度梯度）来控制鞭毛接下来的运动方式（根据浓度梯度调整下一步大致方向）
  - ① 当诱导剂浓度降低或趋避剂浓度增加时，翻转频率增加（缩短学习率），远离不利环境；
  - ② 反之，翻转频率降低，细菌游动，趋向有利环境（类似梯度下降中加入momentum动量因素，如果方向对了就继续保持）。

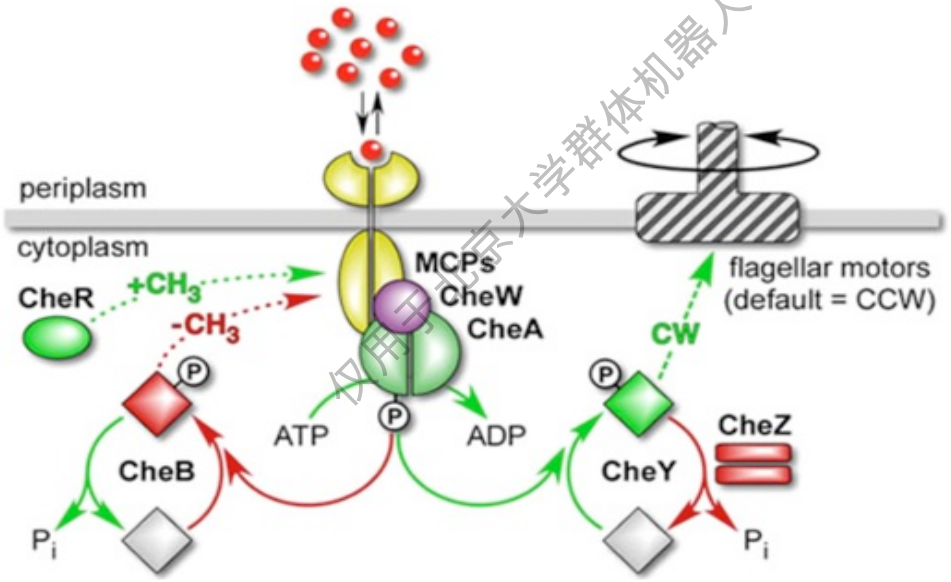
逆时针旋转(counter-clockwise， CCW)

（2）在局部环境中浓度梯度都相等时的调整方式 - Ramdom过程

- 我们知道，溶液中浓度的传播是一种渐变衰减的形态，即一个浓度源的浓度梯度只能传播一定的范围，超过一定的范围就会衰减到几乎无法感知。这就造成了溶液中的浓度梯度看起来就像一个个的小山包此起彼伏。有山包就有山谷、平原
- 在无化学刺激物质或浓度相同的化学环境中（可能刚好存在于一个浓度梯度的平原中），细菌先平稳地直线泳动一段距离，然后突然翻滚改变运动方向，再向前泳动， 再翻滚
- 游泳和翻转循序变化，其特点为随机选择运动方向
- 这就为细菌提供了一种能力，即脱离困境的能力。即使不小心处于一个没有营养的溶液区域，或者说当前离营养源距离比较远。但是依靠自己的ramdom walk机制，在概率上，通过一定的步数后，是有机会到达存在浓度梯度的区域的，到达了浓度梯度区域后就简单了，细菌自身的趋向性会开始发挥作用，帮助细菌达到营养源中心

细菌趋化性分子机制

- 趋化性的发现刺激了许多科学家的兴趣，Julius Adler用基因、生物化学和行为学方法分析大肠杆菌的趋化行为，为详细了解细菌趋化性分子机制做了铺垫
- 趋化细菌膜表面存在专一性的化学受体，以此来感知外界环境中化学物质的浓度变化，并将接收到的化学信号转化为细胞内信号，进而来控制鞭毛的运动方式，表现出相应的趋化性
- 通常信号转导途径分3个部分：1）膜上趋化受体接收信号；2）从膜 受体到鞭毛马达的信号转导；3）对最初信号输入的适应



基本趋化信号传导路径中包括的蛋白。刺激反应包括趋化接收体(MCP)接受信号，通过CheW、CheA、CheY传递到鞭毛马达。通过CheR和CheB调节MCPs的甲基化、去甲基化适应环境

细菌觅食算法的优化机制—趋化运动

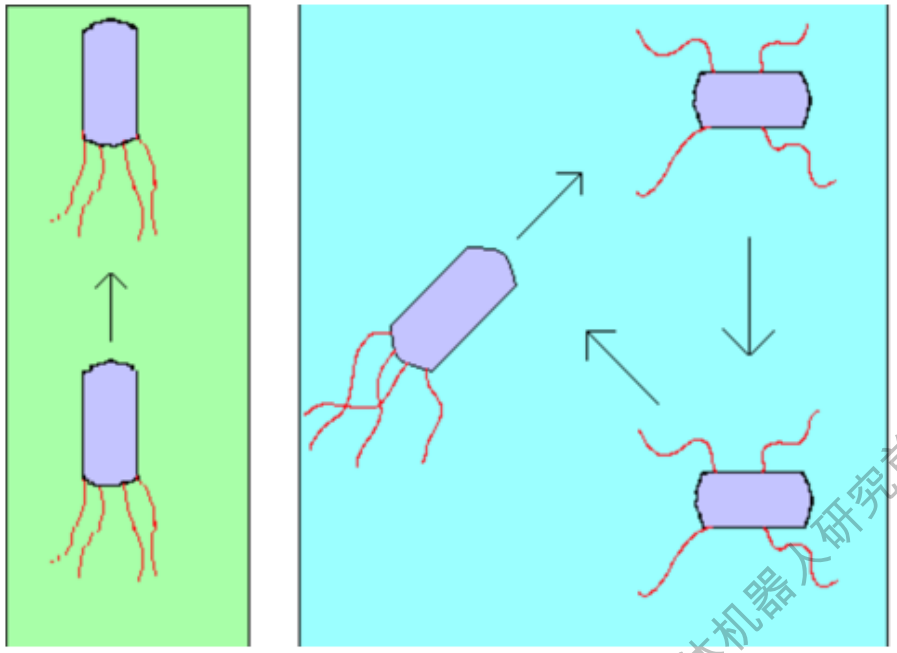
BFO算法通过趋化、复制和驱散三种行为来实现寻优

趋化(chemotaxis)运动

- 在实际的细菌觅食过程中，运动是靠一系列拉伸的鞭毛来实现的
- 鞭毛帮助的大肠杆菌细菌翻转或前进(游泳)
- 这是由细菌在觅食时执行两个基本操作细菌通过趋化运动
- 朝着它们喜欢的营养梯度地方移动并且避免进入有害的环境
- 通常情况下，细菌在友好的环境中会移动较长的一段距离

BFOA的关键思想是在问题搜索空间模仿细菌趋化运动

趋化运动—翻转和前进



(a) (b)

细菌趋化运动—前进和翻转。(a)前进，(b) 翻转

- **翻转**：当它们顺时针方向翻转时，每一根鞭毛都会拉动细胞。这导致了鞭毛的独立运动，并且最终以最少的代价去翻转。在糟糕的地方则频繁地翻转，去寻找一种营养梯度
- **前进(游泳)**：逆时针方向移动鞭毛有助于细菌以非常快的速度游泳

BFO算法通过趋化、复制和驱散三种行为来实现寻优

- **繁殖(reproduction, 复制)**：当它们(细菌)获得了足够的食物，它们的长度增加以及面对着合适的温度，它们将从自己本身的中间断裂开来，形成两个新的细菌。这个现象启发Passino在BFOA中引进繁殖事件
- **驱散(elimination-dispersal, 消除-分散)**：由于突然的环境变化或攻击发生后，趋化过程可能被破坏，一群细菌可能会转移到其他地方或者一些细菌可能被引进到细菌群中。这些构成了真实细菌环境中的消除-分散事件
- 群体：一个区域内的所有细菌被杀死或者一组细菌分散到环境的新部分

细菌觅食算法三个典型的行为模式

BFO算法通过趋化、复制和驱散三种行为来实现寻优

趋化(chemotaxis)行为

- 细菌向事物丰富的区域聚集的行为称为趋化。在趋化过程中，细菌运动模式包括翻转(tumble)和前进(run or swim)。细菌向任意方向移动一定的距离定义为翻转。当细菌完成一次翻转后，若适应度函数值得到改善，将沿同一方向继续移动若干步，直至适应值不再改善，或达到预定的移动步数临界值。此过程定义为前进。通过趋化行为，细菌可获得连续局部寻优的能力

细菌觅食算法三个典型的行为模式

BFO算法通过趋化、复制和驱散三种行为来实现寻优

复制(reproduction)行为

- 一旦生命周期结束，即达到临界趋化次数，细菌将进行繁殖。细菌的繁殖过程遵循自然界“优胜劣汰，适者生存”原则。以趋化过程中各细菌适应值累加和为标准，较差的半数细菌死亡，较好的半数细菌分裂成两个子细菌。子细菌将继承母细菌生物特性，具有与母细菌相同的位置及步长。为简化计算，可以规定复制过程中细菌总数保持不变。通过复制行为，可加快细菌寻优的速度

细菌觅食算法三个典型的行为模式

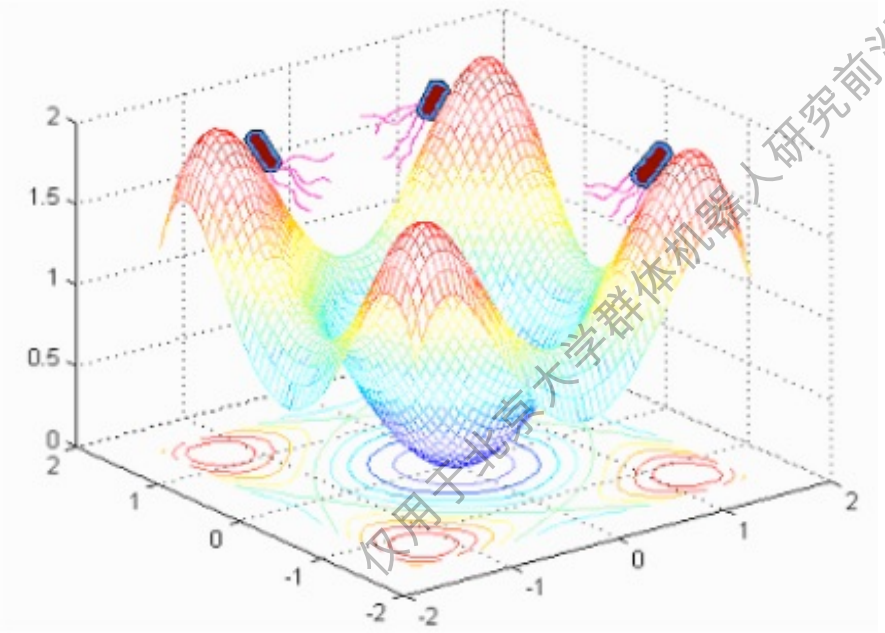
BFO算法通过趋化、复制和驱散三种行为来实现寻优

驱散(elimination-dispersal)行为

- 趋化过程可确保细菌的局部搜索能力，复制过程能加快细菌的搜索速度，但对于复杂的优化问题，趋化和复制无法避免细菌陷入局部极小现象发生。BFA引入驱散过程以加强算法全局寻优能力。细菌在完成一定次数的复制后，将以一定概率被驱散到搜索空间中任意位置。通过驱散行为，可以避免细菌陷入局部极值

细菌觅食系统机制的形式化定义

- 现在我们假设每个细菌的“人生终极目标”为最小化$J(\theta)$，每个细菌的各种趋化、繁衍等机制，目的都是想要找到最小的$J(\theta)$，其中$\theta \in R$ (θ是一个p维的取实数值的向量)。这样，就可以在最优化理论的框架内对细菌的各种生存机制进行分析。
- 注意，BFOA解决的是一个无梯度优化问题，即我们没有关于梯度$\nabla J(\theta)$的测量或者分析描述



每个细菌都在寻找全局最优解，同时又一起构成一个整体群体

- BFOA模拟了细菌系统中的四个观察到的主要机制：趋化，群体，复制以及消除-分散
- 一个虚拟细菌(可称为搜索代理)事实上是一个实验方案在其优化曲面移动，来寻找全局最优解

细菌觅食系统机制的符号化定义

- j：趋化步骤索引
- k：复制步骤索引
- l：消除-分散步骤索引
- p：搜索空间维度
- S：群体中细菌的总数
- N_c：趋化步骤的步数
- N_s：游泳长度
- N_{re}：复制步骤的次数
- N_{ed}：消除-分散事件的次数
- P_{ed}：消除-分散概率
- $P(j,k,l)=\{\theta(j,k,l)\mid i=1,2,\ldots,S\}$：代表 j 趋化步骤中，S细菌群体中的每一个成员的位置，其中，k 复制步骤、l 消除-分散事件
- $J(i,j,k,l)$：表示在 i 细菌搜索定位中的消耗（代价）。我们可以将 J当作“cost”代价来做参考（优化理论的术语）

细菌觅食算法的更新公式

- 细菌觅食算法主要包括三层循环，外层是驱散操作，中间层是繁殖操作，内层是趋化操作
- 细菌觅食算法的核心是内层的趋化性操作，它对应着细菌在寻找食物过程中所采取的方向选择策略，对算法的收敛性有着极其重要的影响
- 通常在趋化过程中，细菌运动模式包括翻转和前进

细菌觅食算法更新公式： $\theta(i,j+1,k,l)=\theta(i,j,k,l)+C(i)\phi(i)$

其中$\theta(i,j,k,l)$表示第i个细菌在经过第l次驱散、第k次复制、第j次趋化后的状态，$C(i)$为移动步长，$\phi(i)$表示取值在[-1,1]之间的随机向量

该算法主要是三个循环，其中的重点为三个循环中的内层循环

for l=1:N_{ed} %驱散操作

for k=1:N_{re} %繁殖操作

for j=1:N_c %趋化操作

细菌觅食算法的步骤

步骤1：**菌群参数的初始化**。包括种群规模(细菌个数) S、每次翻转或者游动的步长$C(i)$、驱散概率P_{ed}、搜索空间的维数p、最大迭代次数、游动次数N_s、趋化次数N_c、复制次数N_{re}、驱散次数N_{ed}等参数的设置

步骤2：**初始化细菌的位置并计算细菌的适应度函数值**。假设细菌个体代表解空间的一个解，$\theta(i,j,k,l)$表示细菌i在第j代趋化循环、第k代复制循环、第l代驱散循环的位置，并计算细菌的适应度函数值$J(i,j,k,l)$

步骤3：驱散循环$l=l+1$；

步骤4：复制循环$k=k+1$

细菌觅食算法的步骤

步骤5：趋化循环$j=j+1$，对每个细菌$i=1, 2, ..., S$，按照下面的方式进行一次循环

(a) 用$J_{last}=J(i, j,k,l)$来保存当前所找到的最优的适应度函数值。翻转，产生随机向量$\phi(i)$。移动，$\theta(i, j+1,k,l)=\theta(i, j,k,l)+C(i)\phi(i)$，其中$C(i)$是细菌$i$的移动步长；

(b) 用$\theta(i, j+1,k,l)$来计算适应度函数值$J(i, j+1,k,l)$，用m来计算游动次数，令$m=m+1$，如果$J(i, j+1,k,l)<J_{last}$，则在该方向上继续游动，否则令$m=N_s$

如果$j<N_c$，则继续按照上述方式进行趋化循环。

细菌觅食算法的步骤

步骤6：对每个细菌$i=1,2,...,S$，计算其N_c次趋化后的适应度函数值之和

$$J_{\text{health}}(i)=\sum_{j=1}^{N_c+1}J(i,j,k,l)$$

注意高代价意味着低健康，按照适应度值对所有细菌进行排序，令适应度函数值大的半数的细菌死亡，而其他的细菌进行自我复制(包括位置θ与适应度函数值J)

步骤7：如果$k<N_{\text{re}}$，则进入步骤4继续进行复制循环

步骤8：对每个细菌$i=1,2,...,S$，按照概率P_{ed}进行驱散，如果P_{ed}小于随机值则保留该细菌，否则该细菌死亡，并在解空间任意生成新的向量。如果$l<N_{\text{ed}}$，则进入步骤3继续进行循环，否则循环结束，同时输出细菌的最优位置与适应度函数值。

细菌觅食算法的改进

- **基于参数的改进**

（1）根据自适应增量调节原理，自适应调节步长等参数

（2）改进菌群规模、步长参数、迭代终止条件

- **基于解空间的改进**

（1）将菌群个体置于量子空间中进行描述

（2）缩小解空间

- **算法机制的改进**

（1）改进大肠杆菌之间的相互作用机制

（2）对趋向操作进行改进

- **与其他算法融合**

（1）引进遗传算法的交叉和变异操作

（2）引进免疫算法的克隆选择思想

（3）与粒子群算法结合

（4）与和声搜索算法结合

（5）与差分进化算法结合

内容提要

- 粒子群优化算法
- 蚂蚁种群优化算法
- 细菌觅食算法
- 人工免疫系统算法

人工免疫系统算法的发展概况

- 二十世纪七十年代，诺贝尔奖获得者、美国生物学家、医学家、免疫学家Jerne提出了免疫系统的网络假说，给出了免疫网络的数学框架，开创了独特性网络理论，Perelson基于独特性网络理论进一步的给出了免疫系统的数学框架
- 1986年，Farmer等人率先基于免疫网络假说构造了免疫系统的动态模型，提出了一些学习算法的构造思想，并探讨了免疫系统与其它人工智能方法的联系，开始了人工免疫系统的研究
- 1989年，Varela讨论了免疫网络以某种方式收敛的思想以及免疫系统能够通过产生不同的抗体和变异适应新环境的思想。
- 随后，有很多学者从生物免疫系统中抽取隐喻(metaphor)机制，用于人工免疫算法的模型设计、算法实现和工程应用
- 1996年12月，在日本首次举行了基于免疫系统的国际专题讨论会，首次提出了“人工免疫系统 (AIS)”的概念
- 随后，人工免疫系统进入了兴盛发展时期

人工免疫系统算法的发展概况

- D. Dasgupta等系统分析了人工免疫系统和人工神经网络的异同，认为在组成单元及数目、交互作用、模式识别、任务执行、记忆学习、系统鲁棒性等方面是相似的，而在系统分布、组成单元间的通信、系统控制等方面是不同的，并指出自然免疫系统是人工智能方法灵感的重要源泉
- D. Dasgupta等认为人工免疫系统已经成为人工智能领域的理论和应用研究热点，相关论文和研究成果逐年增加
- 1997开始，IEEE System, Man and Cybernetics国际会议每年都组织了AIS专题研讨会(Workshop)，并成立了“人工免疫系统及应用分会”
- IEEE Trans. on Evolutionary Computation在2001年和2002年相继出版了AIS专辑
- 其他的国际会议，如GECCO(Genetic and Evolutionary Computation Conference)等，也将人工免疫算法作为大会的主题之一
- 第一届人工免疫系统国际学术会议ICARIS(1st International Conference on Artificial Immune Systems)也于2002年在英国Kent大学召开

- 人工免疫系统已经被用于解决许多不同的工程问题，如日本学者Ishida在1990年用免疫系统解决传感器网络故障诊断问题，美国学者Forrest在1994年将免疫系统用于计算机安全和病毒检测
- 此后，越来越多的学者开始关注人工免疫系统理论的发展和应用，并且推动了AIS在信息安全、机器学习、最优化理论、模式识别、故障诊断、图像处理、自动控制、数据挖掘、数据分析、机器人控制等领域的应用

人工免疫系统的研究范围

- 根据生物免疫系统原理发展新的算法或对现有的免疫算法进行改进，如阴性选择算法、克隆选择算法、免疫遗传算法、免疫优化算法，以及为了完成特定任务而设计的基于免疫原理的算法等
- 用免疫系统自身特性建立新型智能机器学习方法。免疫系统具有动态保持自组织记忆能力，并允许信息遗忘和具有内容可访记忆，这些进化学习机制和学习外界物质的自然防御机制可用于发展新型机器学习系统和人工免疫系统
- 人工免疫系统理论研究。基于数学、非线性科学、复杂系统、混沌、计算智能等工具深入研究人工免疫系统的机制，加强包括一般的原理、方法、定义、定理、数学模型和描述在内的理论研究，加强人工免疫系统的收敛性和稳定性分析，建立完整的人工免疫系统的理论体系

人工免疫系统的研究范围

- 根据生物免疫系统原理发展新的人工免疫模型，包括人工免疫网络模型和人工免疫系统模型两类。各种免疫网络学说，如独特型网络、互联耦合免疫网络、免疫反馈网络和对称网络等，可被借鉴用于人工秒你网络(Artificial Immune Network, AIN)认知模型中，基于免疫网络理论构建具有更强的自组织、自学习、自适应的混合智能网络
- 将免疫系统与神经网络、模糊技术、遗传算法等结合，建立融合的计算方法，用于解决其他智能计算方法难以解决的复杂问题。例如，引入DNA计算技术，实现免疫系统基于DNA分子和片段的慢速学习和免疫系统抵御抗原的自适应快速学习机制，使得人工免疫系统具有较强的抗干扰能力和稳定性；用免疫系统抗体多样性的遗传机制改进遗传算法的搜索优化过程；借鉴神经网络和遗传算法的优化思想优化免疫网络
- 免疫工程应用研究

人工免疫系统的特性

- **多样性**：免疫系统抗体库的多样性特征能及时对不同类型的入侵抗原进行有效的保证和消除
- **容错性**：免疫系统在分类和响应中突发的一些比较小的信息处理错误不会使整个信息处理结果造成严重影响
- **分布自律性**：免疫系统没有集中控制系统，它是由许多局部的并且相互作用的基本信息单元联合起来达到对全局的保护
- **动态稳定性**：免疫系统要消除各种外来的不断变化的入侵抗原，并保持整个系统的稳定
- **自适应鲁棒性**：免疫系统具有非常强的自我学习能力，并且通过学习使得自身能够随环境不断变化而不断改变和完善的一个自适应型的鲁棒进化系统
- **保证收敛**：收敛速度快，即产生满足要求的最优解所用时间较短

生物体液免疫的机理的启示

- 抗原识别：免疫系统能够识别出抗原并根据不同抗原的特性生成不同的浆细胞来产生抗体
- 根据亲和力来选择浆细胞：若产生的抗体与抗原的亲和度高则保留，否则筛掉
- 存在记忆细胞：B细胞分化为浆细胞和记忆细胞，记忆细胞保存亲和度高的抗体信息
- 促进和抑制抗体的产生：能产生亲和度高抗体的浆细胞被促进，反之则被抑制
- 通过交叉变异产生下一代抗体

生物免疫和免疫算法概念之间的对应关系

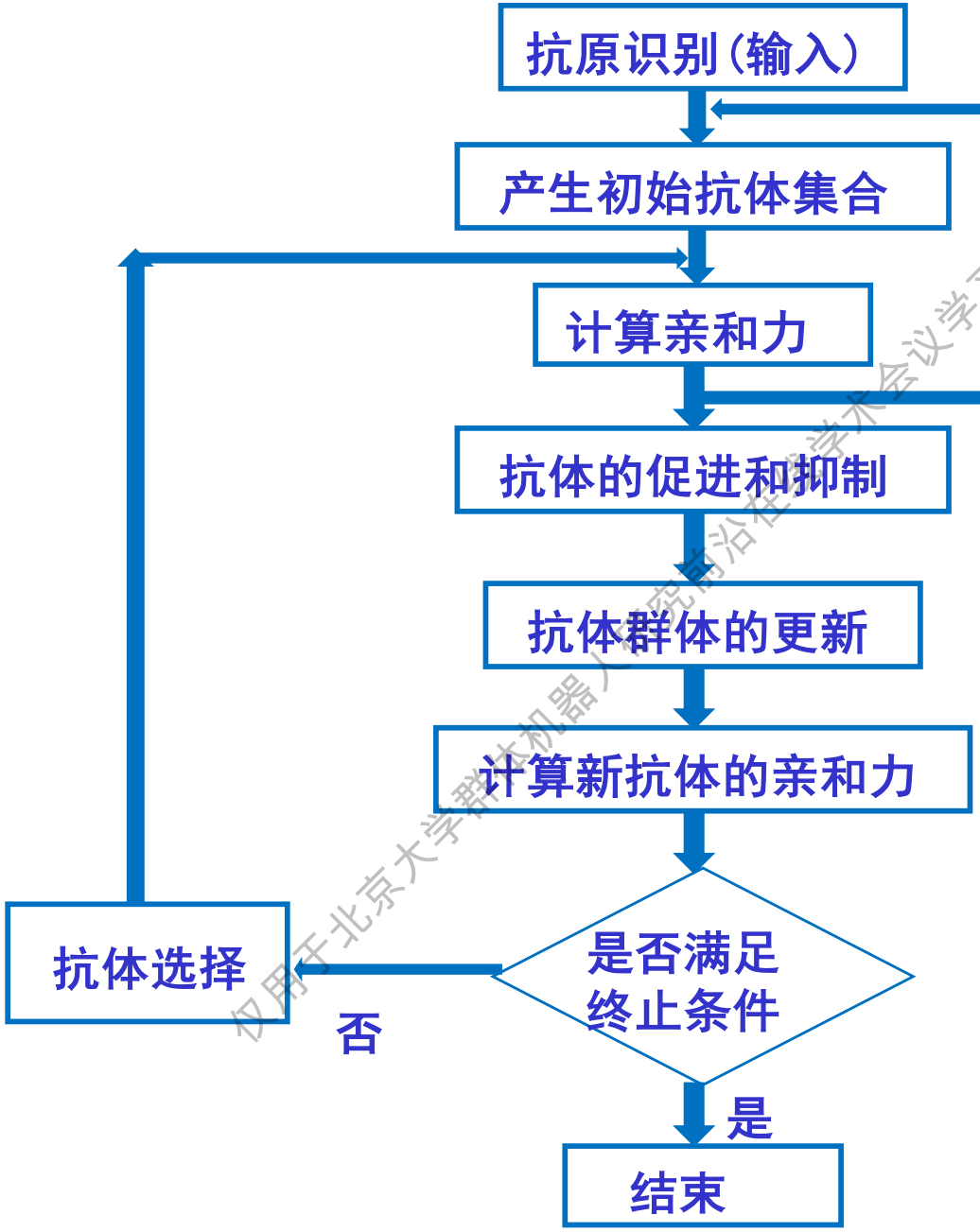
生物免疫系统	免疫算法
抗原	要解决的优化问题
抗体	优化问题的可行解
亲和度	可行解的质量
细胞活化	免疫选择
细胞分化	个体克隆
亲和度成熟	变异
克隆抑制	优秀个体选择
动态稳态维持	种群刷新

人工免疫算法的基本原理

人工免疫算法是一种受生物免疫系统基本机制启发而设计的新型智能优化算法。基本免疫算法从体细胞理论和网络理论得到启发，实现了类似于生物免疫系统的抗原识别、细胞分化、记忆和自我调节的功能。

它将抗原和抗体分别对应于优化问题的目标函数和可行解，把抗原和抗体的亲和力看作可行解与目标函数的匹配程度，用抗体之间的亲和力保证可行解的多样性，通过计算抗体期望生存率来促进较优抗体的遗传和变异，用记忆细胞单元保存择优后的可行解来抑制相似可行解的继续产生并加速搜索到全局最优解。同时，当相似问题再次出现时，它能较快产生适应新问题的次优解甚至最优解。

人工免疫算法的流程图



应用免疫算法求解实际问题时，常将抗原、抗体、它们之间的亲和性对应于优化问题的目标函数、优化解、解与目标函数的匹配程度

人工免疫算法的基本步骤

步骤1：**抗原的识别阶段**。免疫系统确认抗原入侵。输入所求解问题的目标函数和各种约束条件作为免疫算法的抗原，并选择亲和度函数；

步骤2：**初始抗体群体的产生阶段**。激活记忆细胞产生抗体，清除以前出现过的抗原，从包含最优抗体(最优解)的数据库中选择出来一些抗体。确定抗体的编码方式，在解空间中用随机方法产生N个候选解作为初始抗体；

步骤3：**亲和力的计算**。分别计算抗原和抗体之间的亲和力并排序；

步骤4：**记忆细胞的分化**。将与抗原亲和力高的抗体加入到记忆细胞，并执行免疫操作。由于记忆细胞有限，新产生的与抗原亲和力更高的抗体替换较低亲和力的抗体；

步骤5：**抗体的促进和抑制。**高亲和力的抗体受到促进，密度高的抗体受到抑制，通常通过计算抗体存活的期望值来实施；

人工免疫算法的基本步骤

步骤6：**新抗体的产生(抗体群体的更新)**。对未知抗原的响应，产生新淋巴细胞。构造人工免疫算子，通过人工免疫算子(如交叉和变异)的作用，产生进入下一代的抗体；

步骤7：**计算新抗体的亲和力**。如果新抗体中有与抗原匹配的抗体，或已满足预定的程序终止条件则终止程序，否则进入下一步；

步骤8：**抗体选择**。按照优胜劣汰的自然选择机制，在原有的N个有效抗体和新产生的若干个抗体中选择出N个与抗原匹配的较好的抗体构成新的抗体群。在进行选择操作时，应依据抗体之间的排斥力限制进入新抗体群中的相同抗体数目，以保持抗体群中抗体的多样性，增强抗体群的免疫力，防止算法收敛于局部最优解

步骤9：**终止记忆细胞的迭代**。在达到指定阈值的时候终止记忆细胞的生成和选取；

一般的人工免疫算法和遗传算法的区别

一般的人工免疫算法同遗传算法在生物学上有相似之处，如免疫系统的变异就是进化的一种特殊现象、都能提高群体的多样性

- 免疫算法起源于宿主和宿源之间的内部竞争，它所相互作用的环境既包括外部环境也包括内部环境；而遗传算法起源于个体和自私基因之间的外部竞争
- 免疫算法假设免疫元素相互作用（抗原与抗体，抗体与抗体），即每个免疫细胞等个体可以互相作用；而遗传算法不考虑个体之间相互作用
- 免疫算法中基因可以由个体自己选择；遗传算法中基因由环境选择

一般的人工免疫算法和遗传算法的区别

- 免疫算法中，基因（可理解为抗体）组合是为了获得多样性，一般不用交叉算子，因为免疫算法中基因是在同一代个体进行进化；而遗传算法后代个体基因通常是父代交叉的结果，交叉用于混合基因
- 免疫算法的生物学基础是多样性的，大致为：免疫网络模型、克隆选择和阴性选择。遗传算法由单一的生物学理论（进化论）发展而来
- GA是具有很强全局搜索能力的随机化搜索算法，特别适合求出问题的的近似最优解。但GA也存在一些不足，尤其当群体分布不均匀时易出现不成熟收敛。免疫算法最后产生的抗体为问题的最优解，相比遗传算法增加了抗体促进与抑制的步骤，这使得抗体不断朝着有利的方向进化，剔除退化的抗体

信息领域的研究范围这么大，我希望每个学生能够找到一个感兴趣的点，并且成为你今后赖以谋生一辈子的饭碗

谢谢聆听