# 计算智能——蚁群算法

作者 柯良军

西安交通大学 电信学院

October 9, 2015

# 目录

# 蚂蚁觅食行为



# 蚂蚁觅食行为



- 蚂蚁是一种社会性昆虫

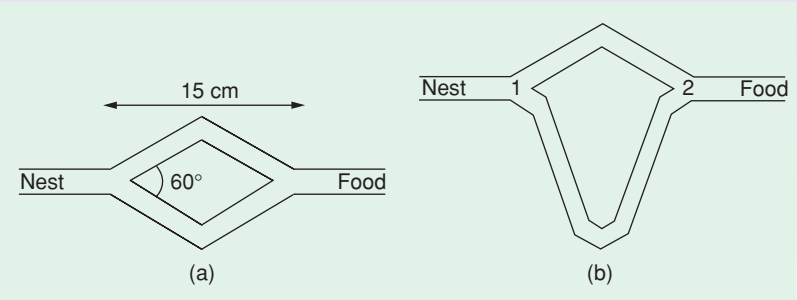
# 蚂蚁觅食行为

- 蚂蚁是一种社会性昆虫
- 蚂蚁食物搜索行为具有非常高的协作性。

# 蚂蚁觅食行为

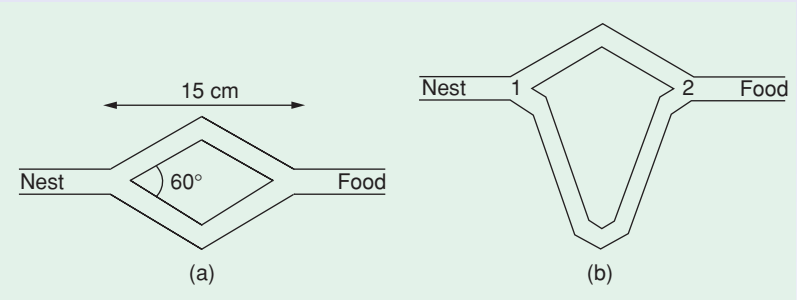
- 蚂蚁是一种社会性昆虫
- 蚂蚁食物搜索行为具有非常高的协作性。
- 当蚂蚁在蚁穴和食物源之间行走时，在路上释放一种称为**信息素**的物质

# Goss实验



# Goss实验

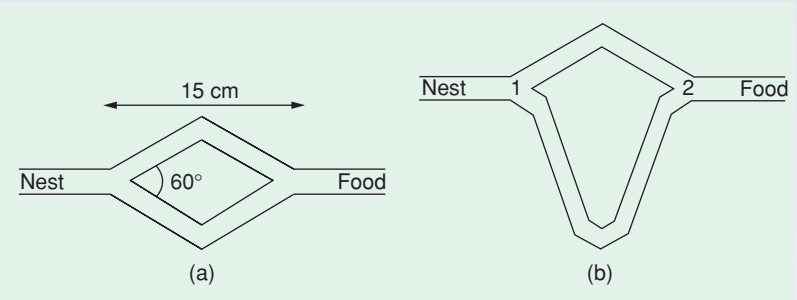
在蚁穴与食物源之间架设二分支桥。



# Goss实验

在蚁穴与食物源之间架设二分支桥。

每次移动时，蚂蚁只能选择两个分支其中之一往返于蚁穴与食物源。

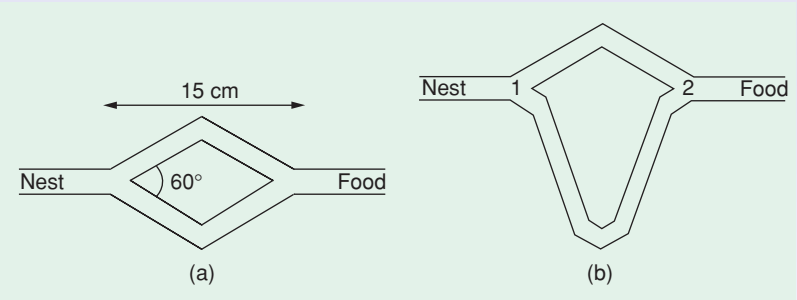


# Goss实验

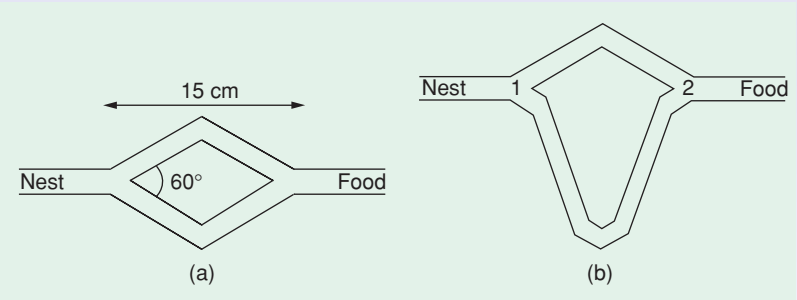
在蚁穴与食物源之间架设二分支桥。

每次移动时，蚂蚁只能选择两个分支其中之一往返于蚁穴与食物源。

实验观察发现，在很短时间内，大多数蚂蚁将会选择路径较短的桥。



# Goss实验



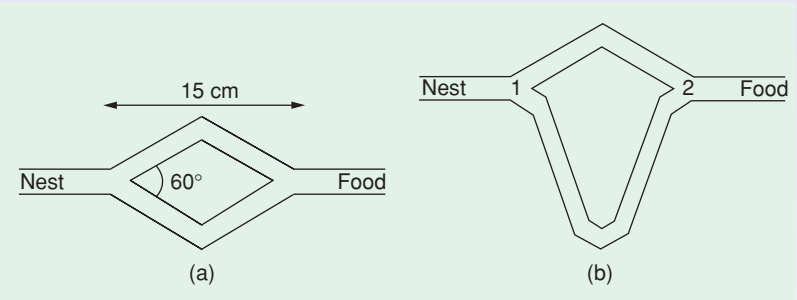

在蚁穴与食物源之间架设二分支桥。

每次移动时，蚂蚁只能选择两个分支其中之一往返于蚁穴与食物源。

实验观察发现，在很短时间内，大多数蚂蚁将会选择路径较短的桥。

同时发现，蚁群选择短分支的概率随着两个分支之间的长度比例增加而增大。

# Goss实验




在蚁穴与食物源之间架设二分支桥。

每次移动时，蚂蚁只能选择两个分支其中之一往返于蚁穴与食物源。

实验观察发现，在很短时间内，大多数蚂蚁将会选择路径较短的桥。

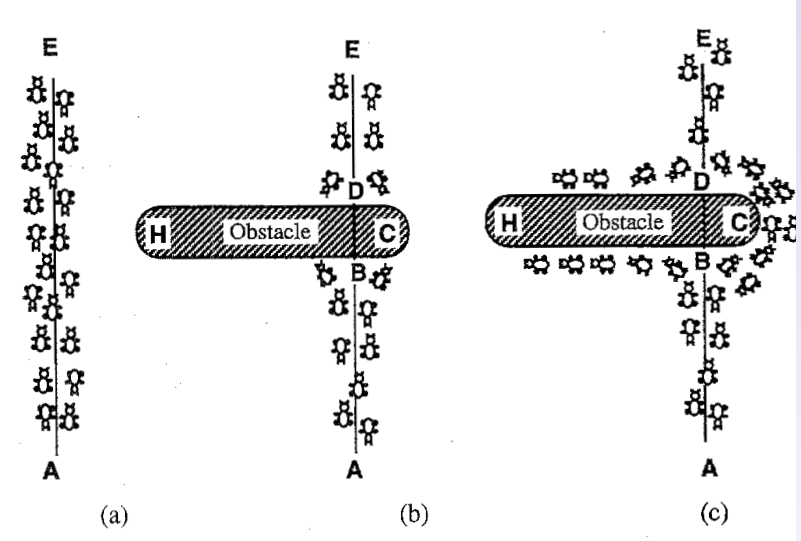
同时发现，蚁群选择短分支的概率随着两个分支之间的长度比例增加而增大。

蚂蚁协作：

一是通讯个体释放信息素体现其对所处环境的物理状态的修正；

二是信息素只能被访问该局部状态的通讯体感应到。这种以环境状态的物理修正为媒介，而且只能被局部接收的通讯方式称为”Stigmergy”。

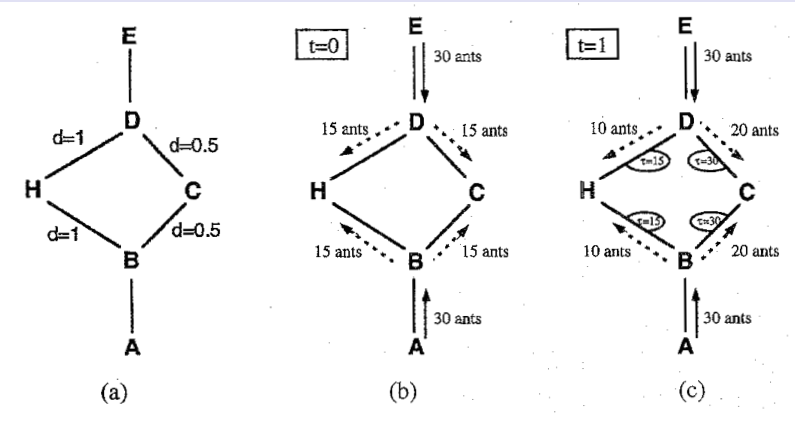
# Dorigo实验




随着信息素的积累，蚂蚁趋向于选择右边的路径

# 觅食模型




E
D
H
C
B
A
d=1
d=0.5
d=1
d=0.5
(a)
t=0
30 ants
15 ants
15 ants
15 ants
15 ants
30 ants
(b)
t=1
30 ants
10 ants
20 ants
τ=15
τ=30
τ=15
τ=30
10 ants
20 ants
30 ants
(c)

# 人工蚂蚁的设计



人工蚂蚁绝大部分行为特征来源于真实蚂蚁，它们的**共同特征**主要有以下几点：

- **人工蚂蚁和真实蚂蚁都是一群相互合作的个体**

  它们可以通过全局范围内相互合作找出问题较好的解。虽然每只人工蚂蚁构造可行解的行为可能较复杂，但是高质量的解都是整个蚁群合作的结果。

# 人工蚂蚁的设计



人工蚂蚁绝大部分行为特征来源于真实蚂蚁，它们的**共同特征**主要有以下几点：

- **人工蚂蚁和真实蚂蚁都是一群相互合作的个体**

  它们可以通过全局范围内相互合作找出问题较好的解。虽然每只人工蚂蚁构造可行解的行为可能较复杂，但是高质量的解都是整个蚁群合作的结果。

- **人工蚂蚁和真实蚂蚁都通过信息素进行间接通讯**

  真实蚂蚁在经过的路径上留下信息素，而人工蚂蚁改变其经过路径上存储的数字信息，该信息记录了蚂蚁当前解和历史解的性能状态，而且可以被以后经过该路径的蚂蚁读写。蚁群通过这种交流方式改变了当前人工蚂蚁所经过的路径周围的环境，同时也改变了整个蚁群所存储的历史信息。另外，蚁群算法中还有一个信息素挥发机制，它模拟真实信息素挥发，逐渐改变信息素。挥发机制使得人工蚂蚁和真实蚂蚁一样可以逐渐忘记历史遗留信息，从而可使人工蚂蚁在进行路径选择时不局限于以前蚂蚁所存留的经验。

# 人工蚂蚁的设计



人工蚂蚁绝大部分行为特征来源于真实蚂蚁，它们的**共同特征**主要有以下几点：

- **人工蚂蚁和真实蚂蚁都是一群相互合作的个体**

  它们可以通过全局范围内相互合作找出问题较好的解。虽然每只人工蚂蚁构造可行解的行为可能较复杂，但是高质量的解都是整个蚁群合作的结果。

- **人工蚂蚁和真实蚂蚁都通过信息素进行间接通讯**

  真实蚂蚁在经过的路径上留下信息素，而人工蚂蚁改变其经过路径上存储的数字信息，该信息记录了蚂蚁当前解和历史解的性能状态，而且可以被以后经过该路径的蚂蚁读写。蚁群通过这种交流方式改变了当前人工蚂蚁所经过的路径周围的环境，同时也改变了整个蚁群所存储的历史信息。另外，蚁群算法中还有一个信息素挥发机制，它模拟真实信息素挥发，逐渐改变信息素。挥发机制使得人工蚂蚁和真实蚂蚁一样可以逐渐忘记历史遗留信息，从而可使人工蚂蚁在进行路径选择时不局限于以前蚂蚁所存留的经验。

# 人工蚂蚁的设计



- **人工蚂蚁和真实蚂蚁都利用了正反馈机制**

  人工蚂蚁受真实蚂蚁觅食行为的启发，它利用了正反馈机制（自催化机制）和解的隐性评价（implicit solution evaluation）[39]。解的隐性评价意味着路径越短则可以更快地被人工蚂蚁经过，从而较短的路径接收了更多的信息素。而正反馈意味着路径越短则信息素越多，从而更多的蚂蚁选取较短地路径。如果合理地利用，正反馈可以成为基于群体的优化算法一个强有力的机制。但是，在应用正反馈时要避免早熟收敛。

# 人工蚂蚁的设计



- **人工蚂蚁和真实蚂蚁都利用了正反馈机制**

  人工蚂蚁受真实蚂蚁觅食行为的启发，它利用了正反馈机制（自催化机制）和解的隐性评价（implicit solution evaluation）[39]。解的隐性评价意味着路径越短则可以更快地被人工蚂蚁经过，从而较短的路径接收了更多的信息素。而正反馈意味着路径越短则信息素越多，从而更多的蚂蚁选取较短地路径。如果合理地利用，正反馈可以成为基于群体的优化算法一个强有力的机制。但是，在应用正反馈时要避免早熟收敛。

- **人工蚂蚁和真实蚂蚁都有一个共同的任务，并且只能局部移动、**

  人工蚂蚁和真实蚂蚁一样都有着共同的任务，那就是寻找连接起点（蚁穴）到终点（食物源）的最短（最小代价）路径。真实蚂蚁不能跳跃，它们只能沿着相邻区域的状态行走。人工蚂蚁也只能一步步地沿着问题的相邻状态移动。

# 人工蚂蚁的设计



- **人工蚂蚁和真实蚂蚁都利用了正反馈机制**

  人工蚂蚁受真实蚂蚁觅食行为的启发，它利用了正反馈机制（自催化机制）和解的隐性评价（implicit solution evaluation）[39]。解的隐性评价意味着路径越短则可以更快地被人工蚂蚁经过，从而较短的路径接收了更多的信息素。而正反馈意味着路径越短则信息素越多，从而更多的蚂蚁选取较短地路径。如果合理地利用，正反馈可以成为基于群体的优化算法一个强有力的机制。但是，在应用正反馈时要避免早熟收敛。

- **人工蚂蚁和真实蚂蚁都有一个共同的任务，并且只能局部移动、**

  人工蚂蚁和真实蚂蚁一样都有着共同的任务，那就是寻找连接起点（蚁穴）到终点（食物源）的最短（最小代价）路径。真实蚂蚁不能跳跃，它们只能沿着相邻区域的状态行走。人工蚂蚁也只能一步步地沿着问题的相邻状态移动。

- **人工蚂蚁和真实蚂蚁都采用随机的、近视的（myopic）状态转移策略**

  与真实蚂蚁类似，人工蚂蚁在构造解时按照一种随机决策策略在相邻状态间移动，而且它们只是利用了局部信息而没有利用前瞻性来预测未来状态。因此，它们所采用的策略在时间和空间上都是局部的。这个策略既与问题特有先验信息有关又与由蚂蚁引起的环境局部改变有关。

# 人工蚂蚁的设计



人工蚂蚁还具有真实蚂蚁所不具备的行为特征，主要有以下几点：

- **人工蚂蚁生活在一个离散空间中**

  它们的移动是由一个离散状态到另一个离散状态的跃迁；而真实蚂蚁是在一个连续的空间爬行。

# 人工蚂蚁的设计



人工蚂蚁还具有真实蚂蚁所不具备的行为特征，主要有以下几点：

- **人工蚂蚁生活在一个离散空间中**

  它们的移动是由一个离散状态到另一个离散状态的跃迁；而真实蚂蚁是在一个连续的空间爬行。

- **人工蚂蚁具有一定的记忆能力**

  它能保存蚂蚁过去的状态。而真实蚂蚁没有表现出这方面的能力。

# 人工蚂蚁的设计



人工蚂蚁还具有真实蚂蚁所不具备的行为特征，主要有以下几点：

- **人工蚂蚁生活在一个离散空间中**

  它们的移动是由一个离散状态到另一个离散状态的跃迁；而真实蚂蚁是在一个连续的空间爬行。

- **人工蚂蚁具有一定的记忆能力**

  它能保存蚂蚁过去的状态。而真实蚂蚁没有表现出这方面的能力。

- **人工蚂蚁释放信息素的时机可以依据具体问题而定**

  而真实蚂蚁是在移动的同时释放信息素。

# 人工蚂蚁的设计



人工蚂蚁还具有真实蚂蚁所不具备的行为特征，主要有以下几点：

- **人工蚂蚁生活在一个离散空间中**

  它们的移动是由一个离散状态到另一个离散状态的跃迁；而真实蚂蚁是在一个连续的空间爬行。

- **人工蚂蚁具有一定的记忆能力**

  它能保存蚂蚁过去的状态。而真实蚂蚁没有表现出这方面的能力。

- **人工蚂蚁释放信息素的时机可以依据具体问题而定**

  而真实蚂蚁是在移动的同时释放信息素。

- **人工蚂蚁可以增加一些额外的本领**

  为了提高系统的优化性能，如预测能力、局部优化、回退等。而这些本领是真实蚂蚁所不具备的

# 人工蚂蚁的设计



人工蚂蚁还具有真实蚂蚁所不具备的行为特征，主要有以下几点：

- **人工蚂蚁生活在一个离散空间中**

  它们的移动是由一个离散状态到另一个离散状态的跃迁；而真实蚂蚁是在一个连续的空间爬行。

- **人工蚂蚁具有一定的记忆能力**

  它能保存蚂蚁过去的状态。而真实蚂蚁没有表现出这方面的能力。

- **人工蚂蚁释放信息素的时机可以依据具体问题而定**

  而真实蚂蚁是在移动的同时释放信息素。

- **人工蚂蚁可以增加一些额外的本领**

  为了提高系统的优化性能，如预测能力、局部优化、回退等。而这些本领是真实蚂蚁所不具备的

总之，蚁群算法是真实蚁群觅食行为的**一种抽象**；同时，为了能有效解决实际优化问题，它赋予了人工蚂蚁一些真实蚂蚁所**不具备的本领**。

# 组合优化问题的描述

组合优化问题通常可以由三元组$(S, f, \Omega)$ 表示，其中$S$是候选解的集合；$f$是目标函数，对于任意$s \in S$有目标函数值$f(x)$；$\Omega$ 是约束条件集合。其优化目标是寻找全局最优可行解。

# 组合优化问题的描述



组合优化问题通常可以由三元组$(S, f, \Omega)$ 表示，其中$S$是候选解的集合；$f$是目标函数，对于任意$s \in S$有目标函数值$f(x)$；$\Omega$ 是约束条件集合。其优化目标是寻找全局最优可行解。组合优化问题$(S, f, \Omega)$ 可以映射为具有如下特征的问题：

（1）有限集合$\varsigma = \{c_1, \cdots, c_{NC}\}$ 表示优化问题的组成元素（component）。

（2）根据所有的可能序列$x =< c_i, c_j, \cdots, c_k, \cdots >$定义问题的有限状态集合$\chi$，其中$c_i, c_j, \cdots, c_k$ 是$\varsigma$ 的元素。序列的长度定义为$|x|$，表示序列中元素的个数，最大值$l < \infty$ 。

（3）候选解集合$S$是有限状态集合$\chi$ 的子集，即$S \subseteq \chi$。

（4）可行状态集合$\widetilde{\chi} \subseteq \chi$ ，由满足约束条件集合$\Omega$ 的序列$x \in \chi$ 构成。

（5）非空集合$S^*$是最优解组成的集合，$S^* \subseteq \widetilde{\chi}$ 且$S^* \subseteq S$。

# 组合优化问题的描述



组合优化问题通常可以由三元组$(S, f, \Omega)$ 表示，其中$S$是候选解的集合；$f$是目标函数，对于任意$s \in S$有目标函数值$f(x)$；$\Omega$ 是约束条件集合。其优化目标是寻找全局最优可行解。组合优化问题$(S, f, \Omega)$ 可以映射为具有如下特征的问题：

（1）有限集合$\varsigma = \{c_1, \cdots, c_{NC}\}$ 表示优化问题的组成元素（component）。

（2）根据所有的可能序列$x =< c_i, c_j, \cdots, c_k, \cdots >$定义问题的有限状态集合$\chi$，其中$c_i, c_j, \cdots, c_k$ 是$\varsigma$ 的元素。序列的长度定义为$|x|$，表示序列中元素的个数，最大值$l < \infty$ 。

（3）候选解集合$S$是有限状态集合$\chi$ 的子集，即$S \subseteq \chi$。

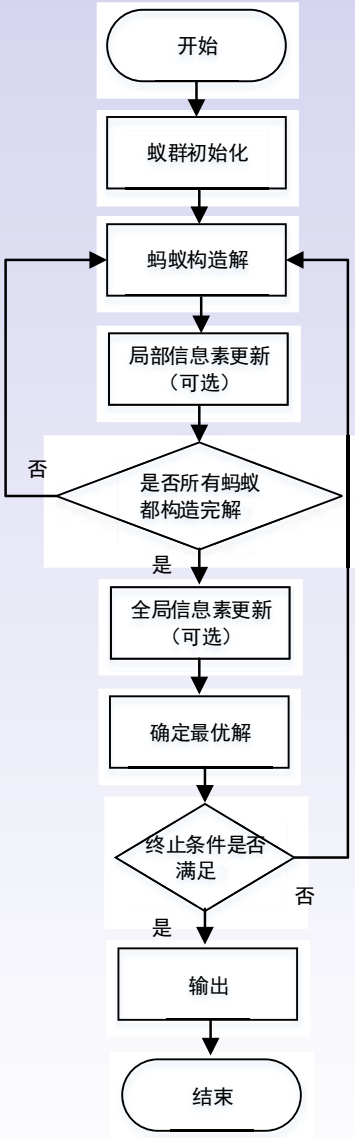
（4）可行状态集合$\widetilde{\chi} \subseteq \chi$ ，由满足约束条件集合$\Omega$ 的序列$x \in \chi$ 构成。

（5）非空集合$S^*$是最优解组成的集合，$S^* \subseteq \widetilde{\chi}$ 且$S^* \subseteq S$。

通过以上的描述，组合优化问题可以通过图的形式表示为$G = (\varsigma, L)$，其中，$\varsigma$表示结点集合，$L$ 表示所有结点的连接弧集合。从而，待求解的问题转化为在一个图中搜索最小代价路径问题。问题的解对应于$G$上的可行序列（可行路径）。它的最优解即为$G$上满足约束条件的最短路径，即目标函数的最优解。

# 蚁群算法的主要流程




开始
蚁群初始化
蚂蚁构造解
局部信息素更新
（可选）
是否所有蚂蚁
都构造完解
否
是
全局信息素更新
（可选）
确定最优解
终止条件是否
满足
否
是
输出
结束

# 人工蚂蚁的特点

蚂蚁系统（AS）是最早的蚁群算法，也是基本蚁群算法。在基本蚁群算法中，人工蚂蚁具有以下特征：

（1）蚂蚁在城市根据信息素和启发信息选择下一个城市$j$。

（2）在从城市$i$到城市$j$移动过程中或是在完成一次循环后，蚂蚁在边$(i,j)$上释放一定量的信息素$\tau(i,j)$。

（3）为了满足问题的约束条件，在一次循环中，不允许蚂蚁选择已经访问过的城市。

# 状态转移规则

在初始时刻，蚂蚁随机选取一个结点。然后蚂蚁从一个结点移动到另一个结点，直到经过所有的结点。设第$k$只蚂蚁当前所在的结点为$i$，则从该结点移到结点$j$的概率为

$$p_{ij} = \begin{cases} \frac{\tau(i,j)^{\alpha}\eta(i,j)^{\beta}}{\sum_{u \in N_k^i} \tau(i,u)^{\alpha}\eta(i,u)^{\beta}} & \text{如果} j \in N_k^i \\ 0 & \text{否则} \end{cases} \tag{1}$$

其中$\eta(i,j)$是弧$(i,j)$上的启发信息。在TSP问题中，$\eta(i,j)$ 一般选取为弧长的倒数。$N_k^i$是由所有未经过的点组成的集合，$\alpha, \beta$分别表示信息素和启发信息相对权重参数，它们控制$\tau(i,j)$和$\eta(i,j)$在决策中所占的比重。由上式可知，那些具有较多的信息素且较短的弧段，被选择的概率较大。

# 信息素更新规则



经过$n-1$次选择，蚂蚁完成一个回路，也就是问题的一个可行解。当蚂蚁原路返回时，在它所经过的弧段上留下信息素，用$\Delta\tau_{ij}^k$表示第$k$只蚂蚁在弧段$(i,j)$上存放的量。它的大小与它构造的可行解$T_k$质量有关。设$L_k$表示该回路的长度，显然，$L_k$越小，解的质量越好，在所经过的弧段上留下的信息素$\Delta\tau_{ij}^k$越大。任意弧段$(i,j)$上信息素总的改变量为

$$\Delta\tau(i,j)=\sum_{k=1}^{m}\Delta\tau_{ij}^k \tag{2}$$

其中$m$是蚂蚁数。

# 信息素更新规则

根据具体算法的不同，$\Delta\tau_{ij}^k$的表达形式可以不同。Dorigo[2]曾给出三种不同的模型，分别称为Ant Cycle System、Ant Quantity System、Ant Density System。在Ant Cycle System中$\Delta\tau_{ij}^k$为：

$$\Delta\tau_{ij}^k = \begin{cases} \frac{Q}{L_k} & \text{如果第k只蚂蚁在本次循环中经过弧}(i,j) \\ 0 & \text{否则} \end{cases} \tag{3}$$

其中$Q$为常数；$L_k$表示第$k$只蚂蚁在本次循环中所走路径的长度。

# 信息素更新规则



在Ant Quantity System中$\Delta\tau_{ij}^{k}$为：

$$\Delta\tau_{ij}^{k}=\begin{cases}\frac{Q}{d_{ij}} & \text{如果第k只蚂蚁在本次循环中经过弧}(i,j)\\ 0 & \text{否则}\end{cases} \tag{4}$$

# 信息素更新规则

在Ant Density System中$\Delta\tau_{ij}^k$为：

$$\Delta\tau_{ij}^k = \begin{cases} Q & \text{如果第k只蚂蚁在本次循环中经过弧}(i,j) \\ 0 & \text{否则} \end{cases} \tag{5}$$

其中$d_{ij}$为城市$i$到城市$j$的距离。后两种算法与Ant Cycle System的区别在于，蚂蚁每走一步都要更新信息素的强度，而不是等到所有蚂蚁完成一个解的构造以后。在Ant Quantity System中，信息素强度的增量为$Q/d_{ij}$，此时增量会随着城市之间距离的减小而增大。在Ant Density System中，从城市i到城市j的蚂蚁在路径上释放的信息素强度是一个与弧长无关的常数$Q$。Ant Cycle System利用全局信息来更新信息素，它在求解TSP问题时性能较好。

# 信息素更新规则



此外，基本蚁群算法引入信息素挥发机制。设信息素的保持系数为$\rho$，则信息素挥发系数为$1-\rho$。信息素按下式调整：

$$\tau(i,j) \leftarrow \rho\tau(i,j) + \Delta\tau(i,j) \tag{6}$$

至此一次迭代结束，进入下一次迭代。重复进行上述过程，直到满足某个停止条件。

# 基本蚁群算法的优势



- **它具有较强的全局搜索能力**

  在算法中，一群蚂蚁通过相互协作来更好地适应环境，以获得更好的性能；使用蚂蚁群体而不是单只蚂蚁，使得算法找到全局最优解的概率增加；另外，使用概率规则而不是确定性规则指导搜索，使得算法可能逃离局部最优。而传统优化算法对初值、迭代步长的选择较敏感，一旦陷入局部最优就很难逃离。

- **它具有潜在的并行性**

  所有蚂蚁同时独立地在解空间中搜索，非常适合于并行实现，因此它本质上是一种高效的并行搜索算法。一方面蚂蚁的搜索行为是独立自主的，不需要集中控制；另一方面，即使一只或者几只蚂蚁作出不好的选择，整个蚁群系统仍然能够保持正常功能。这种分布式并行模式大大提高整个算法的运行效率、鲁棒性。

- **不依赖于优化问题本身的数学性质**

  如连续性、可导性以及目标函数和约束条件的精确数学描述等等。

- **它具有学习能力**

  在复杂的、不确定的、时变的环境中，通过自我学习不断提高蚂蚁的适应性。

# 基本蚁群算法的不足



- **与遗传算法等相比，该算法一般需要较长的搜索时间**

  这是因为蚁群中个体的移动是随机的，虽然通过信息的交流能够向着最优路径进化，但是当群体规模较大时，很难在较短时间内从复杂无章的路径中找出一条较好的路径，而解的构造过程也会占用大量的计算时间。这一缺点是蚁群算法本身决定的，很难有本质上的改进，但可通过采用局部搜索等方法来提高算法收敛性能，减少算法搜索到满意解的时间。

- **容易出现停滞现象**

  停滞现象是指当算法搜索到一定程度后，所有蚂蚁不能构造新的解，以致不能对搜索空间做进一步探索的现象。蚂蚁总是倾向沿着信息素强度高的弧段移动，由信息素更新规则可见，未被经过的弧段与包含在优解中的弧段相比，信息素强度的差异越来越大，它们被选择的概率也就越来越小，从而导致算法有时只能在信息素更新中的优解附近进行搜索。这种信息素更新规则实现了正反馈机制，但是停滞现象是这种方式要避免的一个不足之处。

- **有些优化问题难以用构造图描述**

  虽然构造图在一定程度上扩展了蚁群算法的应用范围，但许多较复杂的实际问题仍然难以用构造图描述。

# 基本蚁群算法的改进途径

针对基本蚁群算法的不足，可从以下方面来改进：

- 信息素释放方式
- 信息素更新规则
- 路径选择策略
- 参数的选取
- 并行实现和计算效率
- 融合其他算法

# 基本蚁群算法的改进途径



人们提出了许多改进的蚁群算法，包括：
**蚁群系统（ACS）、基于优化排序的蚂蚁系统、最大最小蚂蚁系统（MMAS）**等。

# 基本蚁群算法的改进途径



人们提出了许多改进的蚁群算法，包括：
**蚁群系统**（ACS）、**基于优化排序的蚂蚁系统**、**最大最小蚂蚁系统（MMAS）**等。

它们与AS的主要区别在于状态转移规则和信息素更新规则。

# 基本蚁群算法的改进途径



人们提出了许多改进的蚁群算法，包括：
**蚁群系统**（ACS）、**基于优化排序的蚂蚁系统**、**最大最小蚂蚁系统**（**MMAS**）等。

它们与AS的主要区别在于状态转移规则和信息素更新规则。

**ACS**和**MMAS**是两类应用广泛的蚁群算法。

# 蚁群系统-状态转移规则



在ACS中，状态转移规则如下：一只位于结点$i$的蚂蚁按照下式给出的规则选取下一个将要到达的城市

$$j = \begin{cases} argmax_{u \in J(i)}\{\tau(i,u)^{\alpha}\eta(i,u)^{\beta}\} & \text{如果} q \leq q_0 \\ S & \text{否则} \end{cases} \tag{7}$$

其中$q$是一个[0,1]之间的服从均匀分布的随机数，$q_0$是一个参数（$q_0 \in [0,1]$），$S$是按照式（1-1）给出的概率分布选出的一个随机变量,$J(i)$是可选城市集合。

# 蚁群系统-状态转移规则



在ACS中，状态转移规则如下：一只位于结点$i$的蚂蚁按照下式给出的规则选取下一个将要到达的城市

$$j = \begin{cases} argmax_{u \in J(i)}\{\tau(i,u)^{\alpha}\eta(i,u)^{\beta}\} & \text{如果} q \leq q_0 \\ S & \text{否则} \end{cases} \tag{7}$$

其中$q$是一个[0,1]之间的服从均匀分布的随机数，$q_0$是一个参数（$q_0 \in [0,1]$），$S$是按照式（1-1）给出的概率分布选出的一个随机变量,$J(i)$是可选城市集合。

上式给出的状态转移规则称为**伪随机比例状态转移规则**(pseudo random-proportional state transition rule)。和**随机状态转移规则**(random-proportional state transition rule) 一样，都倾向于选择较短的且有较多信息素的边作为移动方向。

# 蚁群系统-状态转移规则



在ACS中，状态转移规则如下：一只位于结点$i$的蚂蚁按照下式给出的规则选取下一个将要到达的城市

$$j = \begin{cases} argmax_{u \in J(i)}\{\tau(i,u)^{\alpha}\eta(i,u)^{\beta}\} & \text{如果} q \leq q_0 \\ S & \text{否则} \end{cases} \tag{7}$$

其中$q$是一个[0,1]之间的服从均匀分布的随机数，$q_0$是一个参数（$q_0 \in [0,1]$），$S$是按照式（1-1）给出的概率分布选出的一个随机变量,$J(i)$是可选城市集合。

上式给出的状态转移规则称为**伪随机比例状态转移规则**(pseudo random-proportional state transition rule)。和**随机状态转移规则**(random-proportional state transition rule) 一样，都倾向于选择较短的且有较多信息素的边作为移动方向。

参数$q_0$ 决定了探索和开发的相对重要性：当一只位于结点$i$的蚂蚁按照式给出的规则选取下一个将要到达的城市时，它先产生一个随机数$q$；如果$q \leq q_0$，则依据式选取最好边，否则依据随机状态转移规则选取一条边。

# 蚁群系统-全局信息素更新规则



在ACS中，只有全局最优的蚂蚁才被允许释放信息素。该策略以及伪随机比例规则的使用，使得蚂蚁具有更强的开发能力：蚂蚁的搜索主要集中在当前迭代为止所找出的最好路径的邻域内。

# 蚁群系统-全局信息素更新规则



在ACS中，只有全局最优的蚂蚁才被允许释放信息素。该策略以及伪随机比例规则的使用，使得蚂蚁具有更强的开发能力：蚂蚁的搜索主要集中在当前迭代为止所找出的最好路径的邻域内。

在每只蚂蚁都构造完一个解之后，全局信息素更新规则按照下式执行：

$$\tau(i,j) \leftarrow \rho\tau(i,j) + \Delta\tau(i,j) \tag{8}$$

其中：

$$\Delta\tau_{ij} = \begin{cases} (L_{gb})^{-1} & \text{如果}(i,j) \in \text{全局最优路径} \\ 0 & \text{否则} \end{cases} \tag{9}$$

其中$\omega$是信息素挥发参数$(0 < \omega < 1)$，$L_{gb}$是全局最优路径长度。

# 蚁群系统-全局信息素更新规则



在ACS中，只有全局最优的蚂蚁才被允许释放信息素。该策略以及伪随机比例规则的使用，使得蚂蚁具有更强的开发能力：蚂蚁的搜索主要集中在当前迭代为止所找出的最好路径的邻域内。

在每只蚂蚁都构造完一个解之后，全局信息素更新规则按照下式执行：

$$\tau(i,j) \leftarrow \rho\tau(i,j) + \Delta\tau(i,j) \tag{8}$$

其中：

$$\Delta\tau_{ij} = \begin{cases} (L_{gb})^{-1} & \text{如果}(i,j) \in \text{全局最优路径} \\ 0 & \text{否则} \end{cases} \tag{9}$$

其中$\omega$是信息素挥发参数$(0 < \omega < 1)$，$L_{gb}$是全局最优路径长度。

只有那些属于全局最优路径的弧段上的信息素才得到增强。

# 蚁群系统-全局信息素更新规则



在ACS中，只有全局最优的蚂蚁才被允许释放信息素。该策略以及伪随机比例规则的使用，使得蚂蚁具有更强的开发能力：蚂蚁的搜索主要集中在当前迭代为止所找出的最好路径的邻域内。

在每只蚂蚁都构造完一个解之后，全局信息素更新规则按照下式执行：

$$\tau(i,j) \leftarrow \rho\tau(i,j) + \Delta\tau(i,j) \tag{8}$$

其中：

$$\Delta\tau_{ij} = \begin{cases} (L_{gb})^{-1} & \text{如果}(i,j) \in \text{全局最优路径} \\ 0 & \text{否则} \end{cases} \tag{9}$$

其中$\omega$是信息素挥发参数$(0 < \omega < 1)$，$L_{gb}$是全局最优路径长度。

只有那些属于全局最优路径的弧段上的信息素才得到增强。

也可以当前迭代的最优解来更新信息素。在这种策略中，上式中用当前迭代的最优路径长度$L_{ib}$代替$L_{gb}$。

# 蚁群系统-全局信息素更新规则



在ACS中，只有全局最优的蚂蚁才被允许释放信息素。该策略以及伪随机比例规则的使用，使得蚂蚁具有更强的开发能力：蚂蚁的搜索主要集中在当前迭代为止所找出的最好路径的邻域内。

在每只蚂蚁都构造完一个解之后，全局信息素更新规则按照下式执行：

$$\tau(i,j) \leftarrow \rho\tau(i,j) + \Delta\tau(i,j) \tag{8}$$

其中：

$$\Delta\tau_{ij} = \begin{cases} (L_{gb})^{-1} & \text{如果}(i,j) \in \text{全局最优路径} \\ 0 & \text{否则} \end{cases} \tag{9}$$

其中$\omega$是信息素挥发参数$(0 < \omega < 1)$，$L_{gb}$是全局最优路径长度。

只有那些属于全局最优路径的弧段上的信息素才得到增强。

也可以当前迭代的最优解来更新信息素。在这种策略中，上式中用当前迭代的最优路径长度$L_{ib}$代替$L_{gb}$。

这两种类型对蚁群系统性能影响差别很小，全局最优的性能要稍好一些。

# 蚁群系统-局部信息素更新规则



在构造解时，蚂蚁应用下式给出的**局部更新规则**对它们所经过的边来更新信息素：

$$\tau(i,j) \leftarrow (1-r)\tau(i,j) + r\Delta\tau(i,j) \tag{10}$$

其中$r(0 < r < 1)$ 是一个参数。

# 蚁群系统-局部信息素更新规则



在构造解时，蚂蚁应用下式给出的**局部更新规则**对它们所经过的边来更新信息素：

$$\tau(i,j) \leftarrow (1-r)\tau(i,j) + r\Delta\tau(i,j) \tag{10}$$

其中$r(0 < r < 1)$ 是一个参数。

当$\Delta\tau(i,j) = \tau_0$时（其中$\tau_0$是一个常数），算法能在较短的时间内获得较好的解。

**注：** 在TSP 问题中，$\tau_0 = (nL_{nn})^{-1}$，其中$L_{nn}$是由最近邻域启发算法求得路径的长度，$n$是城市的数目。

# 蚁群系统-候选表策略



状态转移规则的计算量会很大

# 蚁群系统-候选表策略



**状态转移规则的计算量会很大**

蚁群系统是一种构造式随机算法，在每一步，如果蚂蚁在选择下一个城市时考虑所有可选的城市。

# 蚁群系统-候选表策略



**状态转移规则的计算量会很大**

蚁群系统是一种构造式随机算法，在每一步，如果蚂蚁在选择下一个城市时考虑所有可选的城市。

**候选表**

# 蚁群系统-候选表策略



**状态转移规则的计算量会很大**

蚁群系统是一种构造式随机算法，在每一步，如果蚂蚁在选择下一个城市时考虑所有可选的城市。

**候选表**

候选表是一个表，它记录从当前城市出发偏好程度较高的城市（preferred cities）。
只要候选表还有未访问过的城市，蚂蚁就会按照状态转移规则从候选表中选取一个城市。当候选表所有城市都被访问过时，蚂蚁才会考虑不在候选表中的城市。

# 最大最小蚂蚁系统



Stützle和Hoos在2000年提出MMAS。它与AS的差异主要体现在信息素更新规则上：

**1）它采用精英策略来更新信息素。**

具体而言，在每个蚂蚁构造完一个解之后，只增加最优解对应弧段的信息素。这个解可能是当前最优解（best-so-far solution），也可能是当前迭代的最优解（iteration-best solution）。

# 最大最小蚂蚁系统



Stützle和Hoos在2000年提出MMAS。它与AS的差异主要体现在信息素更新规则上：

**1）它采用精英策略来更新信息素。**

具体而言，在每个蚂蚁构造完一个解之后，只增加最优解对应弧段的信息素。这个解可能是当前最优解（best-so-far solution），也可能是当前迭代的最优解（iteration-best solution）。

利用当前最优解来更新信息素，可使蚁群获得较强的开发能力，而利用当前迭代的最优解来更新信息素，可使蚁群获得较强的探索能力。

# 最大最小蚂蚁系统



Stützle和Hoos在2000年提出MMAS。它与AS的差异主要体现在信息素更新规则上：

**1）它采用精英策略来更新信息素。**

具体而言，在每个蚂蚁构造完一个解之后，只增加最优解对应弧段的信息素。这个解可能是当前最优解（best-so-far solution），也可能是当前迭代的最优解（iteration-best solution）。

利用当前最优解来更新信息素，可使蚁群获得较强的开发能力，而利用当前迭代的最优解来更新信息素，可使蚁群获得较强的探索能力。

当只使用当前最优解时，搜索可能会过快地集中到这个解的周围，从而限制了对新解的搜索，甚至可能会陷于局部最优解。而利用当前迭代的最优解来更新信息素可以减少这样的风险。

# 最大最小蚂蚁系统



Stützle和Hoos在2000年提出MMAS。它与AS的差异主要体现在信息素更新规则上：

**1）它采用精英策略来更新信息素。**

具体而言，在每个蚂蚁构造完一个解之后，只增加最优解对应弧段的信息素。这个解可能是当前最优解（best-so-far solution），也可能是当前迭代的最优解（iteration-best solution）。

利用当前最优解来更新信息素，可使蚁群获得较强的开发能力，而利用当前迭代的最优解来更新信息素，可使蚁群获得较强的探索能力。

当只使用当前最优解时，搜索可能会过快地集中到这个解的周围，从而限制了对新解的搜索，甚至可能会陷于局部最优解。而利用当前迭代的最优解来更新信息素可以减少这样的风险。

**混合策略**：在迭代过程中增加使用当前最优解进行信息素更新的频率，能有效提高算法性能。

# 最大最小蚂蚁系统



2）信息素的取值限制在区间$[\tau_{min}, \tau_{max}]$。

# 最大最小蚂蚁系统



**2）信息素的取值限制在区间**$[\tau_{min}, \tau_{max}]$**。**

它通过将超出这个范围的值强制设置为$\tau_{min}$或$\tau_{max}$，避免不同弧段的信息强度差异过大，从而达到避免停滞的目的。

# 最大最小蚂蚁系统



**3）它将信息素初始化为$\tau_{max}$。**

在MMAS中，信息素的值在第一次迭代之后都被设置为$\tau_{max}(1)$。

# 最大最小蚂蚁系统



**3）它将信息素初始化为$\tau_{max}$。**

在MMAS中，信息素的值在第一次迭代之后都被设置为$\tau_{max}(1)$。 通过将信息素的初始值设置为某个非常大的数值来实现。

# 最大最小蚂蚁系统



**3）它将信息素初始化为$\tau_{max}$。**

在MMAS中，信息素的值在第一次迭代之后都被设置为$\tau_{max}(1)$。 通过将信息素的初始值设置为某个非常大的数值来实现。

其作用是：蚂蚁在算法的初始阶段能够具有更好的探索能力。

# 最大最小蚂蚁系统



**4）MMAS还利用信息素的平滑机制来提高其性能。**
当MMAS已经收敛或接近收敛时，这种机制将信息素作如下调整:

$$\tau(i,j)^* \leftarrow (1-\delta)\tau(i,j) + \delta(\tau_{max} - \tau(i,j)) \tag{11}$$

其中$0 < \delta < 1$，$\tau(i,j)$ 与$\tau(i,j)^*$是信息素调整前后的信息素值。

# 最大最小蚂蚁系统

4）**MMAS还利用信息素的平滑机制来提高其性能。**
当MMAS已经收敛或接近收敛时，这种机制将信息素作如下调整:

$$\tau(i,j)^* \leftarrow (1-\delta)\tau(i,j) + \delta(\tau_{max} - \tau(i,j)) \tag{11}$$

其中$0 < \delta < 1$，$\tau(i,j)$ 与$\tau(i,j)^*$是信息素调整前后的信息素值。

平滑处理机制的基本思想是通过增加选择有着较低信息素值的解元素的概率来**提高搜索新解的能力**。

# 最大最小蚂蚁系统

4）**MMAS还利用信息素的平滑机制来提高其性能。**
当MMAS已经收敛或接近收敛时，这种机制将信息素作如下调整:

$$\tau(i,j)^* \leftarrow (1-\delta)\tau(i,j) + \delta(\tau_{max} - \tau(i,j)) \tag{11}$$

其中$0 < \delta < 1$，$\tau(i,j)$ 与$\tau(i,j)^*$是信息素调整前后的信息素值。

平滑处理机制的基本思想是通过增加选择有着较低信息素值的解元素的概率来**提高搜索新解的能力**。

由于$\delta < 1$，它能避免完全丢失在算法运行过程中所积累的信息。当$\delta = 1$时，它相当信息素的重新初始化。而当$\delta = 0$时，该机制不发挥作用。

# 最大最小蚂蚁系统



**4）MMAS还利用信息素的平滑机制来提高其性能。**
当MMAS已经收敛或接近收敛时，这种机制将信息素作如下调整:

$$\tau(i,j)^* \leftarrow (1-\delta)\tau(i,j) + \delta(\tau_{max} - \tau(i,j)) \tag{11}$$

其中$0 < \delta < 1$，$\tau(i,j)$ 与$\tau(i,j)^*$是信息素调整前后的信息素值。

平滑处理机制的基本思想是通过增加选择有着较低信息素值的解元素的概率来**提高搜索新解的能力**。

由于$\delta < 1$，它能避免完全丢失在算法运行过程中所积累的信息。当$\delta = 1$时，它相当信息素的重新初始化。而当$\delta = 0$时，该机制不发挥作用。

平滑机制的作用:

- 改善算法的探索能力;
- 降低MMAS 对信息素下限的敏感程度;
- 有利于在全局范围内搜索新的解，同时避免过早收敛于局部优解。

# 有限级信息素蚁群算法



**动机：**

在蚁群算法中，信息素直接影响着整个搜索过程。在信息素更新时利用到目标函数值。

# 有限级信息素蚁群算法



**动机：**

在蚁群算法中，信息素直接影响着整个搜索过程。在信息素更新时利用到目标函数值。

但目标函数值的变化规律难以预知，这给算法的参数设置带来很大的困难。

# 有限级信息素蚁群算法



**动机：**

在蚁群算法中，信息素直接影响着整个搜索过程。在信息素更新时利用到目标函数值。

但目标函数值的变化规律难以预知，这给算法的参数设置带来很大的困难。

对于两个成常数比例的目标函数，即使基本蚁群算法采用完全相同的参数，算法性能却可能不同。

# 有限级信息素蚁群算法–算法描述



MMAS和ACS具有以下共同特点：

- 参数的设置是依赖问题的，几乎没有规律可循，而且算法对参数比较敏感。

# 有限级信息素蚁群算法–算法描述



MMAS和ACS具有以下共同特点：

- 参数的设置是依赖问题的，几乎没有规律可循，而且算法对参数比较敏感。
- 在算法的运行过程中，信息素是按照指数下降的，这是算法易于早熟的一个原因。

# 有限级信息素蚁群算法–算法描述



MMAS和ACS具有以下共同特点：

- 参数的设置是依赖问题的，几乎没有规律可循，而且算法对参数比较敏感。
- 在算法的运行过程中，信息素是按照指数下降的，这是算法易于早熟的一个原因。
- 在算法的运行过程中，大量的信息素相同或相近。实际上，如果弧上的信息素相近，它们被选取的概率差异非常小，可以近似地把它们看成是等同的。

# 有限级信息素蚁群算法–主要流程



**主要思想：**

# 有限级信息素蚁群算法–主要流程



**主要思想：**

把信息素分成有限个级别，用完全不同的方式更新信息素，并且信息素的更新量与目标函数值无关。

# 有限级信息素蚁群算法–主要流程



**主要思想：**

把信息素分成有限个级别，用完全不同的方式更新信息素，并且信息素的更新量与目标函数值无关。

**算法的主要流程为：**

步骤1：设定参数，初始化信息素。

步骤2：按照路径选择规则构造问题的解。

步骤3：按照信息素更新规则更新信息素。

步骤4：判断停止条件是否满足，若满足，算法终止；否则返回到步骤2。

# 有限级信息素蚁群算法–信息素更新规则



信息素被分成有限个级别，不同的级别按照一定的映射关系对应不同实数值，这样相同级别的弧的信息素相同。

# 有限级信息素蚁群算法–信息素更新规则



信息素被分成有限个级别，不同的级别按照一定的映射关系对应不同实数值，这样相同级别的弧的信息素相同。

信息素更新通过级别的变动来实现，对于当前最优弧，提高其级别，对于其他的弧，降低其级别。

# 有限级信息素蚁群算法–信息素更新规则



信息素被分成有限个级别，不同的级别按照一定的映射关系对应不同实数值，这样相同级别的弧的信息素相同。

信息素更新通过级别的变动来实现，对于当前最优弧，提高其级别，对于其他的弧，降低其级别。

更新时只用加、减法。记$h(i,j)$ 是弧$(i,j)$上的级别，$g(x)$是单调增的正实函数，它实现从级别到信息素的映射关系。

# 有限级信息素蚁群算法–信息素更新规则



信息素更新规则如下：

（u1）$\forall(i,j): h(i,j) \leftarrow h(i,j) - r_1$；

（u2）如果$f(\widehat{w}) > f(w_t)$，则$\widehat{w} = w_t$；

（u3）对于$\widehat{w}$，$\forall(i,j) \in \widehat{w}, h(i,j) \leftarrow h(i,j) + 2$；

（u4）$\forall(i,j): h(i,j) \leftarrow \max(h(i,j), 1)$；

（u5）$\forall(i,j): h(i,j) \leftarrow \min(h(i,j), M)$；

（u6）$\forall(i,j): \tau(i,j) \leftarrow g(h(i,j))$；。

其中$\widehat{w}$是当前最优解，$w_t$是本次迭代的最优解，参数$r_1$、$r_2$是正整数，$r_1 < r_2$，称$r_1$为惩罚级别数，$r_1 = 1$，$r_2$为奖励级别数，参数$M$是最大级别数。$\tau_m ax$是信息素上界。约定$g(M) = \tau_m ax$，$g(1) = 1$。

# 有限级信息素蚁群算法–信息素更新规则



信息素更新规则如下：

（u1）$\forall(i,j): h(i,j) \leftarrow h(i,j) - r_1$；

（u2）如果$f(\widehat{w}) > f(w_t)$，则$\widehat{w} = w_t$；

（u3）对于$\widehat{w}$，$\forall(i,j) \in \widehat{w}, h(i,j) \leftarrow h(i,j) + 2$；

（u4）$\forall(i,j): h(i,j) \leftarrow \max(h(i,j), 1)$；

（u5）$\forall(i,j): h(i,j) \leftarrow \min(h(i,j), M)$；

（u6）$\forall(i,j): \tau(i,j) \leftarrow g(h(i,j))$；。

其中$\widehat{w}$是当前最优解，$w_t$是本次迭代的最优解，参数$r_1$、$r_2$是正整数，$r_1 < r_2$，称$r_1$为惩罚级别数，$r_1 = 1$，$r_2$为奖励级别数，参数$M$是最大级别数。$\tau_m ax$是信息素上界。约定$g(M) = \tau_m ax$，$g(1) = 1$。

$g(x)$一般选取为:

# 有限级信息素蚁群算法–信息素更新规则



信息素更新规则如下：

（u1）$\forall(i,j): h(i,j) \leftarrow h(i,j) - r_1$；

（u2）如果$f(\widehat{w}) > f(w_t)$，则$\widehat{w} = w_t$；

（u3）对于$\widehat{w}$，$\forall(i,j) \in \widehat{w}, h(i,j) \leftarrow h(i,j) + 2$；

（u4）$\forall(i,j): h(i,j) \leftarrow \max(h(i,j), 1)$；

（u5）$\forall(i,j): h(i,j) \leftarrow \min(h(i,j), M)$；

（u6）$\forall(i,j): \tau(i,j) \leftarrow g(h(i,j))$；。

其中$\widehat{w}$是当前最优解，$w_t$是本次迭代的最优解，参数$r_1$、$r_2$是正整数，$r_1 < r_2$，称$r_1$为惩罚级别数，$r_1 = 1$，$r_2$为奖励级别数，参数$M$是最大级别数。$\tau_m ax$是信息素上界。约定$g(M) = \tau_m ax$，$g(1) = 1$。

$g(x)$一般选取为:

线性函数:

$$\frac{\tau_m ax - 1}{M - 1}(x - 1) + 1$$

# 有限级信息素蚁群算法–信息素更新规则



信息素更新规则如下：

（u1）$\forall(i,j): h(i,j) \leftarrow h(i,j) - r_1$；

（u2）如果$f(\widehat{w}) > f(w_t)$，则$\widehat{w} = w_t$；

（u3）对于$\widehat{w}$，$\forall(i,j) \in \widehat{w}, h(i,j) \leftarrow h(i,j) + 2$；

（u4）$\forall(i,j): h(i,j) \leftarrow \max(h(i,j), 1)$；

（u5）$\forall(i,j): h(i,j) \leftarrow \min(h(i,j), M)$；

（u6）$\forall(i,j): \tau(i,j) \leftarrow g(h(i,j))$；。

其中$\widehat{w}$是当前最优解，$w_t$是本次迭代的最优解，参数$r_1$、$r_2$是正整数，$r_1 < r_2$，称$r_1$为惩罚级别数，$r_1 = 1$，$r_2$为奖励级别数，参数$M$是最大级别数。$\tau_m ax$是信息素上界。约定$g(M) = \tau_m ax$，$g(1) = 1$。

$g(x)$一般选取为:

线性函数:

$$\frac{\tau_m ax - 1}{M - 1}(x - 1) + 1$$

凹函数:

$$\sqrt{\frac{\tau_m ax^2 - 1}{M - 1}(x - 1) + 1}$$

# 有限级信息素蚁群算法–信息素更新规则



信息素更新规则如下：

（u1）$\forall(i,j): h(i,j) \leftarrow h(i,j) - r_1$；

（u2）如果$f(\widehat{w}) > f(w_t)$，则$\widehat{w} = w_t$；

（u3）对于$\widehat{w}$，$\forall(i,j) \in \widehat{w}, h(i,j) \leftarrow h(i,j) + 2$；

（u4）$\forall(i,j): h(i,j) \leftarrow \max(h(i,j), 1)$；

（u5）$\forall(i,j): h(i,j) \leftarrow \min(h(i,j), M)$；

（u6）$\forall(i,j): \tau(i,j) \leftarrow g(h(i,j))$；。

其中$\widehat{w}$是当前最优解，$w_t$是本次迭代的最优解，参数$r_1$、$r_2$是正整数，$r_1 < r_2$，称$r_1$为惩罚级别数，$r_1 = 1$，$r_2$为奖励级别数，参数$M$是最大级别数。$\tau_m ax$是信息素上界。约定$g(M) = \tau_m ax$，$g(1) = 1$。

$g(x)$一般选取为:

线性函数:

$$\frac{\tau_m ax - 1}{M - 1}(x - 1) + 1$$

凹函数:

$$\sqrt{\frac{\tau_m ax^2 - 1}{M - 1}(x - 1) + 1}$$

# 有限级信息素蚁群算法–信息素更新规则



**更新方式：** 通过函数$g(x)$、奖励级别数、惩罚级别数来协调控制级别的变化，进而控制信息素的变化。

# 有限级信息素蚁群算法–信息素更新规则



**更新方式：** 通过函数$g(x)$、奖励级别数、惩罚级别数来协调控制级别的变化，进而控制信息素的变化。

**作用1：** 使得蚁群能有效地在最优解的邻域搜索；另一方面，使得蚁群具有一定的探索新区域的能力。

# 有限级信息素蚁群算法–信息素更新规则



**更新方式：** 通过函数$g(x)$、奖励级别数、惩罚级别数来协调控制级别的变化，进而控制信息素的变化。

**作用1：** 使得蚁群能有效地在最优解的邻域搜索；另一方面，使得蚁群具有一定的探索新区域的能力。

**作用2：** 由于更新量与目标函数值无关，因此对于两个成常数比例的目标函数，如果算法采用完全相同的参数，算法性能将相同。

# 有限级信息素蚁群算法–信息素更新规则



**更新方式：** 通过函数$g(x)$、奖励级别数、惩罚级别数来协调控制级别的变化，进而控制信息素的变化。

**作用1：** 使得蚁群能有效地在最优解的邻域搜索；另一方面，使得蚁群具有一定的探索新区域的能力。

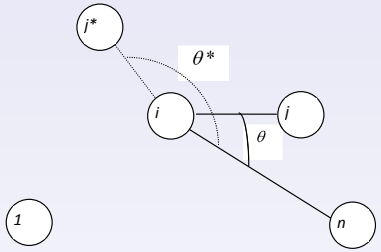
**作用2：** 由于更新量与目标函数值无关，因此对于两个成常数比例的目标函数，如果算法采用完全相同的参数，算法性能将相同。

有限级信息素蚁群算法的路径选择规则与基本蚁群算法一样。

# 信息素和启发信息的定义

根据团队定向问题的定义，它可以表示为一个构造图（construction graph），其顶点就是原问题的顶点。任意边$i, j$都赋有一定量的信息素$\tau(i, j)$，用于表征从一个顶点转移到下一个顶点的偏好（desirablity）。

假设蚂蚁当前位置为$i$，启发信息$\eta(i, j)$用来表征一种先验的偏好。注意到团队定向问题的优化目标是在规定时间内，使得团队的总收益最大化，因此蚂蚁更加偏好于那些具有较高收益且离$i$较近的点。




图：$\angle jin$的示意.

# 信息素和启发信息的定义



这一观察启发我们考虑点$j$的以下性质：

1）收益$r_j$；

2）$i$ 和$j$之间的距离$c_{ij}$；

3）如果所有点都位于同一平面且任意两点直线可达，此时可以考虑$\angle jin$的度数，即$\arccos w_{ij}$，其中$w_{ij} = (c_{ij}^2 + {}_{in}^2 - c_{jn}^2)/2c_{ij}c_{in}$。

在下图中，给出了$\angle jin$的示意图。点$j$和$j*$对应的角度分别为$\theta$和$\theta^*$，且$\theta < \theta^*$。由图可见，如果选取$j^*$，则使车辆偏离点$n$，而如果选取$j$，则使车辆偏向于点$n$。考虑到时间约束，点$j$（从局部来看）优于点$j*$。依据这些性质，启发信息的定义如下：

$$\eta(i,j) = \frac{r_j}{c_{ij}} exp(\gamma w_{ij}) \tag{12}$$

其中$\gamma$是一参数（$\gamma \geq 0$），它决定$w_{ij}$的重要性。当$\gamma = 0$，此时不考虑$w_{ij}$

# 解的构造



由约束可知，违反下式的点一定是不可行点。

$$c_{i1} + c_{in} \leq T_{max}(2 \leq i \leq n-1) \tag{13}$$

如果事先去掉这些不可行点，就可以减少需要考虑的点数。为方便起见，不失一般性，假定所有的点都满足式（5-10）。

# 解的构造

**约束的处理**是构造解的一个关键点。一种可行方法是将原问题转化为无约束问题，常见的方法有罚函数法（包括内点法、外点法）。但是罚因子的选取是个难点。而蚁群算法是一类构造性算法，它能在每个构造步去掉不可行点，这为处理约束提供了方便。

**在构造解的过程中，蚂蚁相当于一个具有简单智能的决策者**。它负责为每个车辆选取一条可行路径（即在规定时间内从起点出发到达终点的路径）。具体而言，在每个构造步，蚂蚁选取一车辆并为该车辆选取一个可行点，直到所有的车辆都到达终点。

# 解的构造



本文考虑以下方法：

1) 串行法。在这种方法中，蚂蚁先给一个车辆选择一条可行路径，然后再给下一个车辆选择一条可行路径，直到所有车辆都安排了一条可行路径。

2）同步法。在这种方法中，蚂蚁按照一定次序给每个车辆选择一可行点，这一过程反复进行，直到所有车辆都安排了一条可行路径。考虑两种安排车辆的次序：一种是确定性次序，即给车辆选择点的次序是固定不变的；另一种是随机性次序，即给车辆选择点的次序是随机的。相应地，这两种次序对应的方法分别称为确定性同步法和随机性同步法。需要注意的是，如果某个车辆没有可行点，则在以后的构造过程中不再考虑。

3）同时法。在这种方法中，蚂蚁从连接每个车辆当前所在点到所有可行点的边中选择一个边。该方法同时确定一车辆和一个可行点。

# 解的构造

考虑一个总点数$n$为7，且车辆数$m$为2的团队定向问题。下图分别给出了四种构造方法的示意效果。在每个图中，边上的数字表示构造步的序号。

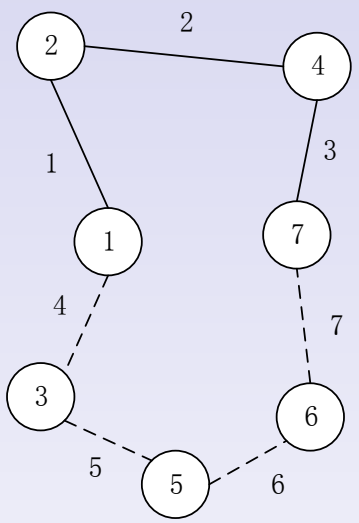
# 解的构造

为了在蚁群算法的框架下描述这四种方法，采用以下记号：$u_i$：在第$k$个构造步，第$i$个车辆所处的点$(1 \leq i \leq m)$；$C_{u_i}$：由所有未被经过且满足式（5-11）的点组成的集合。即

$$\forall v \in C_{u_i} L(t_i) + c_{u_i v} + vn \leq T_{max} \tag{14}$$

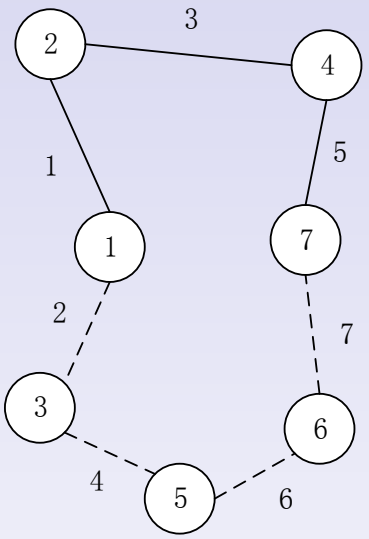
其中$L(t_i)$是第$i$个车辆经过的（未完成）路径$t_i$的长度。如果$C_{u_i}$是空集，即没有可行点，则终点被选取并且第$i$条路径完成。$v_k$：在第$k$构造步选取的点；$q_k$：在第$k$构造步选取的车辆。

# 解的构造



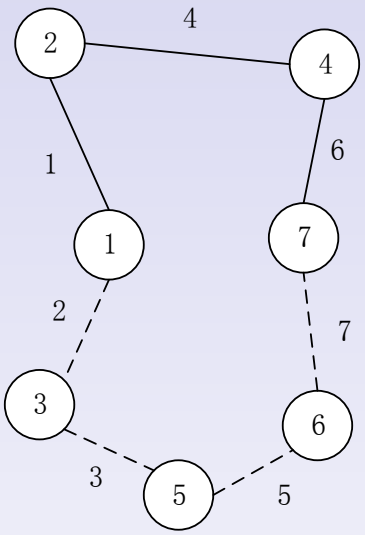

图：串行法示意图（点数为7，车辆数为2）。实线是车辆1的路径，虚线是车辆2的路径。边上的数字表示构造的次序.

# 解的构造



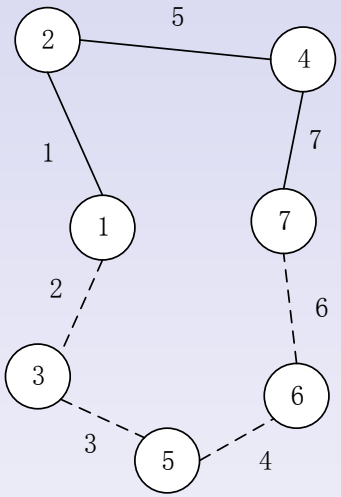

图：确定性同步法示意图（点数为7，车辆数为2）。实线是车辆1的路径，虚线是车辆2的路径。边上的数字表示构造的次序.

# 解的构造




图：随机性同步法示意图（点数为7，车辆数为2）。实线是车辆1的路径，虚线是车辆2的路径。边上的数字表示构造的次序.

# 解的构造




图：同时法示意图（点数为7，车辆数为2）。实线是车辆1的路径，虚线是车辆2的路径。边上的数字表示构造的次序.

# 解的构造

在串行法、确定性同步法和随机性同步法中，蚂蚁按照以下概率选择下一点：

$$P(v_{k+1}=v, q_{k+1}=j|C_{u_i}, 1\le i\le m, q_k, \tau)= \begin{cases} \frac{\tau(u_j,v)^\alpha \eta(u_j,v)^\beta}{\sum_{w\in C_{u_j}} \tau(u_j,w)^\alpha \eta(u_j,w)^\beta} & \text{如果} v\in C_{u_j} \\ 0 & \text{否则} \end{cases} \tag{15}$$

其中，$\alpha$和$\beta$是参数。注意到在串行法中车辆$q_k$和$q_{k+1}$是相同的，而在两种同步法中它们可能是不同的。虽然这个概率决策规则与基本蚁群算法的随机概率规则形式上有所不同，但是，它在选取下一个移动方向时都偏好于选取启发信息和信息素较大的边，其基本思想与基本蚁群算法的随机概率规则类似。

# 解的构造

在同时法（见图5）中，蚂蚁按照式（5-13）给出的概率选择下一个点：

$$P(v_{k+1}=v,q_{k+1}=j|C_{u_i},1\le i\le m,q_k,\tau)= \tag{16}$$
$$\begin{cases}\frac{\tau(u_j,v)^\alpha\eta(u_j,v)^\beta}{\sum_{i=1}^m\sum_{w\in C_{u_i}}\tau(u_i,w)^\alpha\eta(u_i,w)^\beta} & \text{如果}v\in C_{u_j}\\ 0 & \text{否则}\end{cases}$$

由于在每个构造步只有一个车辆和一个点被选取，式（5-13）可以通过保存$\sum_{w\in C_{u_i}}\tau(u_i,w)^\alpha\eta(u_i,w)^\beta(1\le i\le m)$来快速地计算。假定车辆$j$和点$v$被选取，则对于车辆$i(i\ne j)$，它所对应的可行点集变为$C_{u_i}\ \{v\}$。

# 信息素的更新规则



一旦所有的蚂蚁都构造了一个解，按照最大最小蚂蚁系统给出的规则更新信息素（即图5-1中PheromoneUpdate）：

$$\tau(u,v)^{l+1} \leftarrow \rho\tau(u,v)^{l} + \Delta\tau(u,v), \tag{17}$$

$$\text{如果}\tau(u,v)^{l+1} < \tau_{min}\text{则}\tau(u,v)^{l+1} = \tau_{min} \tag{18}$$

$$\text{如果}\tau(u,v)^{l+1} > \tau_{max}\text{则}\tau(u,v)^{l+1} = \tau_{max} \tag{19}$$

其中$\tau(u,v)^{l}$是边$(u,v)$在第$l$代时的信息素。如果$(u,v)$在第$l$代被最优蚂蚁经过，则$\Delta(u,v)$ 等于$F(s_{best})$，否则，$\Delta(u,v)=0$。最优解$s_{best}$可能是本次迭代获得的最优解$s_{ib}$也可能是当前最优解$s_{gb}$。是质量函数（quality function）[113]，它由下式给出：

$$F(x) = \sum_{i=2}^{n-1}\sum_{k=1}^{m} r_i y_{ik} / \sum_{i=2}^{n-1} r_i \tag{20}$$

# 信息素的上下界

$\tau_{max}$和$\tau_{min}$分别是信息素的上下界。设置上下界的目的是为了避免停滞。它们按照下式设置为：

$$\tau_{max} = \frac{F(s_{gb})}{(1-\rho)} \tag{21}$$

$$\tau_{min} = (1 - \sqrt[n]{P_{best}})/((avg-1)\sqrt[n]{P_{best}})\tau_{max}$$

其中$avg$等于$n/2$，$P_{best}$是参数（$0 \leq P_{best} \leq 1$）。

# 局部搜索

局部搜索常用来改善蚁群算法的性能，但是在应用中要折中考虑解质量和计算时间。虽然目前已有一些有效的局部搜索，本文利用文献中的局部搜索。其基本思想是：先用2-opt法缩短解中每条路径的长度，再插入尽可能多的可行点。这个过程反复进行，直到路径长度不能缩短或者不能再插入可行点时终止。在算法中，局部搜索对每个构造解进行改进，它在所有蚂蚁都构造完解之后且在信息素还未更新之前执行。

# 测试数据说明



本节用实验来分析文中所提出的算法并比较四种构造法。所有代码用C＋＋实现，运行环境为：3GHZ CPU，1G Ram，windows XP操作系统。采用文献[145]中的387个测试算例，它们分别属于7个数据集。这些数据集的顶点数目分别为32、21、33、100、66、64、102。在每个数据集中，每个点的坐标和对应的收益不同。每个数据集由三组数据子集构成，每个数据子集的车辆数分别为2、3、4。在数据子集中，每个算例的区别仅在于。每个数据子集用表示，其中表示该数据子集所在数据集的序号，表示车辆数。每个算例用表示，其中表示该算例所在数据集中的序号，表示车辆数，表示该算例在相应数据子集中的序号。本文用混合策略更新信息素：每隔5代，当前最优解被用来更新信息素，在其他代中，都用本次迭代获得的最优解来更新信息素。对于每个算例，测试10 次。

# 参数设置

在测试算法之前，首先确定算法的参数。考虑到计算时间与计算结果，最大运行代数NC为2000，蚂蚁数为20。$Nni = 250$。数值结果表明当$\alpha$，$\beta$都取为0.5 时，算法性能较好。

# 四种构造法的比较

表给出了四种构造方法在两个规模最大数据集（即第4个数据集）中的计算结果。由于在其他数据集中四种构造方法的差异不大，详细计算结果在附录中给出。由结果可见，串行法和同时法最好。确定性同步法与随机性同步法的计算结果相当，前者略好。在大多数算例中，串行法的最大值优于其他三种方法。在第4数据集中，同时法的平均值较好。

# 四种构造法的比较



表：四种构造方法在21个数据子集中的平均计算结果

| | 串行法 | | 确定性同步法 | | 随机性同步法 | | 同时法 | |
|---|---|---|---|---|---|---|---|---|
| 数据子集 | 最大值 | 平均值 | 最大值 | 平均值 | 最大值 | 平均值 | 最大值 | 平均值 |
| p4.2.a | 206 | 206.0 | 206 | 206.0 | 206 | 206.0 | 206 | 206.0 |
| p4.2.b | 341 | 338.7 | 341 | 338.0 | 341 | 338.7 | 341 | 340.2 |
| p4.2.c | 452 | 447.9 | 452 | 448.8 | 452 | 449.4 | 452 | 448.0 |
| p4.2.d | 531 | 527.5 | 531 | 528.7 | 530 | 528.4 | 531 | 528.2 |
| p4.2.e | 618 | 596.9 | 600 | 595.6 | 600 | 597.0 | 613 | 599.5 |

对每一种方法，表中给出了每一组的最大值、平均值。由表的结果可见，在数据集1中，四种方法的计算结果完全相同，且在每一次测试中都找到最好解。在其他数据集中，结果再次表明，串行法获得的结果最好，同时法次之。另外实验发现，串行法需要的计算时间较大，但是，它能在51.1s内（平均意义下）求解每个算例。因此，串行法能在解质量（solution quality）和计算时间之间较好地折中。以后，仅讨论串行法。

# 实验分析



表：四种构造方法在21个数据子集中的平均计算结果

| | 串行法 | | 确定性同步法 | | 随机性同步法 | | 同时法 | |
|---|---|---|---|---|---|---|---|---|
| 数据子集 | 最大值 | 平均值 | 最大值 | 平均值 | 最大值 | 平均值 | 最大值 | 平均值 |
| 7.2 | 642.7 | 637.4 | 641.5 | 637.7 | 641.0 | 637.1 | 641.2 | 637.4 |
| 7.3 | 599.9 | 597.1 | 599.4 | 595.5 | 598.6 | 594.6 | 599.2 | 596.0 |
| 7.4 | 519.1 | 517.0 | 518.2 | 516.3 | 518.4 | 516.2 | 518.4 | 516.3 |

# 实验分析



本文所提出的算法与下列算法进行了比较：

CGW: Chao等[145]提出的五步法;

TMH: Tang 和Miller-Hooks [150]提出的禁忌算法;

GTP: Archetti等[151]提出的基于罚方法的禁忌算法;

GTF: Archetti等提出的基于可行方法的禁忌算法;

FVF: Archetti等快速变邻域算法;

SVF: Archetti等慢变邻域算法，它与FVF的主要区别在于参数的选取。

由于上述文献中只给出了相应算法求得的每个算例的最大收益（最大值），本文只比较最大收益。

# 四种构造法的比较



在数值实验时，路径长度的精度是一个必须考虑的问题。在CGW中，Chao等人没有采用实数精度，而是把路径长度四舍五入到小数点后第一位。而其他文献没有明确指出它们的数值精度。显然地，当采用实数精度时所得到的结果可能较差。为了与现有算法相比较，本文中的算法主要考虑实数精度。

对于每个算法，每个数据子集的（平均）最大总收益在表4中给出。可见ACO-TOP和SVF的计算结果最好，GTF和FVF次之，GTP比以上算法结果略差，而CGW和TMH的结果最差。ACO-TOP和SVF分别在15个数据子集中获得优于其他算法的结果。另外ACO-TOP能找到347个算例已知最大收益，并且它能找到12个算例新的最大总收益（详细结果在表5-6中给出）。由以上比较结果可见，ACO-TOP具有较好的性能。

# 实验分析



表：7种算法的计算结果

| 数据子集 | ACO-TOP | CGW | TMH | GTP | GTF | FVF | SVF |
|---|---|---|---|---|---|---|---|
| 7.2 | 642.7 | 637.4 | 641.5 | 637.7 | 641.0 | 637.1 | 641.2 |
| 7.3 | 599.9 | 597.1 | 599.4 | 595.5 | 598.6 | 594.6 | 599.2 |
| 7.4 | 519.1 | 517.0 | 518.2 | 516.3 | 518.4 | 516.2 | 518.4 |

表??给出了每个算法的平均计算时间。注意到CGW 在SUN 4/730 Workstation 25 MHz上运行，TMH 在DEC Alpha XP1000 computer 667 MHz运行，而其他算法在2.8GHZ PC上运行。这些算法的运行环境不一致，难以直接比较时间。注意到SVF运行的PC性能与本章算法运行的PC接近，它在两个规模最大的数据集中的运行时间都远远超过一分钟。SVF 在第4个数据集中的计算时间为1118s（将近20分钟），而在第7个数据集中的计算时间为911s（将近15分钟）。ACO-TOP在第4个数据集中耗时最长（51.1s）

，但是它能在一分钟内求解每个问题，因此本文提出的算法可以在合理的时间内获得令人满意的解。

# 实验分析

表：7种算法在每个数据集上消耗的计算时间

| 数据集 | ACO-TOP | CGW | TMH | GTP | GTF | FVF | SVF |
|---|---|---|---|---|---|---|---|
| Set1 | 7.9 | 15.4 | N.A. | 10.0 | 5.0 | 1.0 | 22.0 |
| Set2 | 3.8 | 0.9 | N.A. | 0.0 | 0.0 | 0.0 | 1.0 |
| Set3 | 8.5 | 15.4 | N.A. | 10.0 | 9.0 | 1.0 | 19.0 |
| Set4 | 51.1 | 934.8 | 796.7 | 612.0 | 324.0 | 121.0 | 1118.0 |
| Set5 | 25.2 | 193.7 | 71.3 | 147.0 | 105.0 | 30.0 | 394.0 |
| Set6 | 20.3 | 150.1 | 45.7 | 96.0 | 48.0 | 20.0 | 310.0 |
| Set7 | 44.7 | 841.4 | 432.6 | 582.0 | 514.0 | 90.0 | 911.0 |

# 问题说明

多维背包问题是一个典型的子集问题。在子集问题中优化目标是从原物品集中选取一组最优可行物品。

注：其他子集问题有约束满足问题、bin packing and cutting stock、加权边k- 势树问题以及最大团问题等。

与旅行商问题相比，子集问题关注于**选取哪些物品**而**不注重物品被选取的次序**。

# 构造图

在求解多维背包问题时，先将问题建模为一个构造图，图的顶点对应于每个物品，每个边对应于两个顶点（物品）之间的连线。蚂蚁在这个图中行走以构造问题的解。

# 信息素释放方式

假定$s$是一个解，且$S = \{o_1, o_2, \cdots, o_{|S|}\}$是被选物品集，其中$|S|$是$S$的基数。它们采用的信息素释放方式如下：

- 把信息素释放在被选物品上。在这种方式中，信息素用来表征对物品的偏好。它的基本思想是增加被选物品的偏好使得这些物品在以后的解构造过程中更可能被选中。

# 信息素释放方式

假定$s$是一个解，且$S = \{o_1, o_2, \cdots, o_{|S|}\}$是被选物品集，其中$|S|$是$S$的基数。它们采用的信息素释放方式如下：

- 把信息素释放在被选物品上。在这种方式中，信息素用来表征对物品的偏好。它的基本思想是增加被选物品的偏好使得这些物品在以后的解构造过程中更可能被选中。
- 把信息素释放到连接每对先后被选物品$(o_u, o_{u+1}$的边上。其基本思想是当上一个被选物品是$o_u$ 时，增大选择$o_{u+1}$ 的可能性。在这种方式中，信息素表示从一个顶点（物品）出发到下一个特定顶点（物品）的偏好。

# 信息素释放方式

假定$s$是一个解，且$S = \{o_1, o_2, \cdots, o_{|S|}\}$是被选物品集，其中$|S|$是$S$的基数。它们采用的信息素释放方式如下：

- 把信息素释放在被选物品上。在这种方式中，信息素用来表征对物品的偏好。它的基本思想是增加被选物品的偏好使得这些物品在以后的解构造过程中更可能被选中。
- 把信息素释放到连接每对先后被选物品$(o_u, o_{u+1}$的边上。其基本思想是当上一个被选物品是$o_u$ 时，增大选择$o_{u+1}$ 的可能性。在这种方式中，信息素表示从一个顶点（物品）出发到下一个特定顶点（物品）的偏好。
- 把信息素释放到连接每对被选取物品的边上。其基本思想是增加同时选取$S$中任意两个物品的可能性。

# 信息素释放方式



假定$s$是一个解，且$S = \{o_1, o_2, \cdots, o_{|S|}\}$是被选物品集，其中$|S|$是$S$的基数。它们采用的信息素释放方式如下：

- **把信息素释放在被选物品上**。在这种方式中，信息素用来表征对物品的偏好。它的基本思想是增加被选物品的偏好使得这些物品在以后的解构造过程中更可能被选中。
- 把信息素释放到连接每对先后被选物品$(o_u, o_{u+1}$的边上。其基本思想是当上一个被选物品是$o_u$ 时，增大选择$o_{u+1}$ 的可能性。在这种方式中，信息素表示从一个顶点（物品）出发到下一个特定顶点（物品）的偏好。
- 把信息素释放到连接每对被选取物品的边上。其基本思想是增加同时选取$S$中任意两个物品的可能性。

# 启发信息的定义



动态启发信息，即在算法运行过程中启发信息依赖于当前已构造的部分解，其数学表达式为：

$$\eta_{S_k}(j) = \frac{p_j}{\sum_{i=1}^{m} r_{ij}/d_{S_k}(i)} \tag{22}$$

其中$d_{S_k}(i) = b_i - \sum_{g \in S_k} r_{ig}$，$S_k$是已被选中的物品组成的集合。

# 启发信息的定义



三种静态启发信息，即启发信息在算法运行过程中保持不变。它们分别如下：

$$\text{第一种启发信息：}\ \eta_j = p_j^{d_1} \tag{23}$$

其中$d_1$是参数。

$$\text{第二种启发信息：}\ \eta_j = \begin{cases} p_j^{d_1}/s_j^{d_2} & if\, s_j \neq 0 \\ p_j^{d_1} & K \end{cases} \tag{24}$$

其中$s_j = \max\limits_{1 \le i \le m}(r_{ij})$, $d_1$和$d_2$是参数。

$$\text{第三种启发信息：}\ \eta_j = \begin{cases} p_j^{d_1}/s_j^{d_2} & if\, s_j \neq 0 \\ p_j^{d_1} & K \end{cases} \tag{25}$$

其中$s_j = \sum_{i=1}^{m} r_{ij}$, $d_1$和$d_2$是参数。

# 启发信息的定义



启发信息是下式定义的拟效用率(pseudo-utility ratio)：

$$\eta(j) = \frac{p_j}{\sum_{i=1}^{m} w_i r_{ij}} \tag{26}$$

其中$w_i$是第$i$个约束在原问题松弛线性规划的影子价格（对偶变量），分母表示物品$j$的紧致性（tightness）。

# 启发信息的定义

启发信息是下式定义的拟效用率(pseudo-utility ratio):

$$\eta(j) = \frac{p_j}{\sum_{i=1}^{m} w_i r_{ij}} \tag{26}$$

其中$w_i$是第$i$个约束在原问题松弛线性规划的影子价格（对偶变量），分母表示物品$j$的紧致性（tightness）。
可见，蚂蚁偏好于利润高且紧致性小的物品。

# 解的构造



在构造解时，蚂蚁用一个维向量来标记物品是否被选取，它对应于一个解。这个向量的每个分量都初始化为0。在第构造步，蚂蚁依据下式定义的概率来选择物品

$$P(c_k = j|\tau) = \begin{cases} \frac{\tau(i,j)^\alpha \eta(i,j)^\beta}{\sum_{u \in U_k} \tau(i,u)^\alpha \eta(i,u)^\beta} & \text{如果} j \in U_k \\ 0 & \text{否则} \end{cases} \tag{27}$$

其中$U_k(1 \leq k \leq n)$是由满足约束且未被选择的物品组成的集合。$\alpha, \beta$是非负参数，它们控制信息素和启发信息的相对重要性。由上式可知，蚂蚁偏好于那些具有较高信息素和启发信息的物品。假设物品$j$被选取，则向量的第$j$个分量设置成1。构造过程在$U_k$为空集时终止。

# 解的构造

在解构造过程中，可选物品集的确定对算法速度有重要影响。因为$r_{ij} \geq 0$，所以一旦某个物品违反了约束，它在以后的构造过程都是不可选的，这就是说$U_k \supset U_{k+1}$。假设在第$k$构造步，如果物品$o \in U_k$满足$r_{io} \leq b_i - \sum_{j=1}^{n} r_{ij}x_j (i = 1, \cdots, m)$，则该物品就是可选物品。

# 解的构造

在解构造过程中，可选物品集的确定对算法速度有重要影响。因为$r_{ij} \geq 0$，所以一旦某个物品违反了约束，它在以后的构造过程都是不可选的，这就是说$U_k \supset U_{k+1}$。假设在第$k$构造步，如果物品$o \in U_k$满足$r_{io} \leq b_i - \sum_{j=1}^{n} r_{ij}x_j (i = 1, \cdots, m)$，则该物品就是可选物品。

注意到保存$b_i - \sum_{j=1}^{n} r_{ij}x_j (i = 1, \cdots, m)$**可提高计算速度。**

# 信息素的更新



在每个蚂蚁构造完一个解后，只使用最优解更新信息素。具体而言，信息素按照以下方式更新:

$$\forall u \in J, \tau(u)^{l+1} \leftarrow \rho\tau(u)^{l} + \Delta\tau(u), \tag{28}$$

$$\text{如果}\tau(u)^{l+1} < \tau_{min}\text{则}\tau(u)^{l+1} = \tau_{min} \tag{29}$$

$$\text{如果}\tau(u)^{l+1} > \tau_{max}\text{则}\tau(u)^{l+1} = \tau_{max} \tag{30}$$

其中$\tau(u)^{l}$是物品$u$在第$l$代时的信息素，$\rho$是信息素保持率（pheromone persistence）($1-\rho$是信息素的挥发率)。令$\widetilde{s}^{best}$为最优解，它可能是当前最优解$s^{gb}$也可能是本次迭代的最优解$s^{ib}$。如果$\widetilde{s}^{best}$的第$u$个分量等于1，则$\Delta\tau(u)$等于$g(\widetilde{s}^{best})$，其中$g(x) = 1/\sum_{j=1}^{n} p_j(1-x_j)$；否则$\Delta\tau(u)$为0。$\tau_{max}$和$\tau_{min}$分别是信息素的上下界。在信息素更新后，那些被最优蚂蚁选中的物品将接受更多的信息素，因此它们的偏好度（desirability）增加了。信息素的上界被设置为$g(s^{gb})/(1-\rho)$。以后详细讨论信息素下界的选取。

# 局部搜索



局部搜索能有效避免算法陷于局部极值点，因此它常用来改进蚁群算法。

# 局部搜索

局部搜索能有效避免算法陷于局部极值点，因此它常用来改进蚁群算法。

## 基本思想为：

用任两个未被选物品替换一个被选物品，每次（可行的）替换获得一定的收益，直到找到收益最大的替换。为了提高局部搜索的计算速度，可以对所有物品按照收益进行降序排列。由于在确定第二个未被选物品时，如果本次替换的收益小于当前最大收益，就不用尝试其他未被选物品。因此降序排列能起到减少计算量的目的。

# 局部搜索



局部搜索能有效避免算法陷于局部极值点，因此它常用来改进蚁群算法。

基本思想为：

用任两个未被选物品替换一个被选物品，每次（可行的）替换获得一定的收益，直到找到收益最大的替换。为了提高局部搜索的计算速度，可以对所有物品按照收益进行降序排列。由于在确定第二个未被选物品时，如果本次替换的收益小于当前最大收益，就不用尝试其他未被选物品。因此降序排列能起到减少计算量的目的。

它按照以下方式运行：在每一代，一旦某个蚂蚁构造了一个解，应用局部搜索来改进这个解。

# 信息素下界的选取

St ü tzle和Hoos建议将信息素的下界选取为

$$\tau_{min} = \epsilon \tau_{max} \tag{31}$$

其中$\epsilon = (1-\sqrt[n]{P_{best}})/((avg-1)\sqrt[n]{P_{best}})$, $avg$等于$n/2$, $P_{best}(0 < P_{best} < 1)$是参数。为简单起见，称这种方法为SH 法（SH method）。

在应用该方法时，关键在于选取合适的$P_{best}$。

# 信息素下界的选取

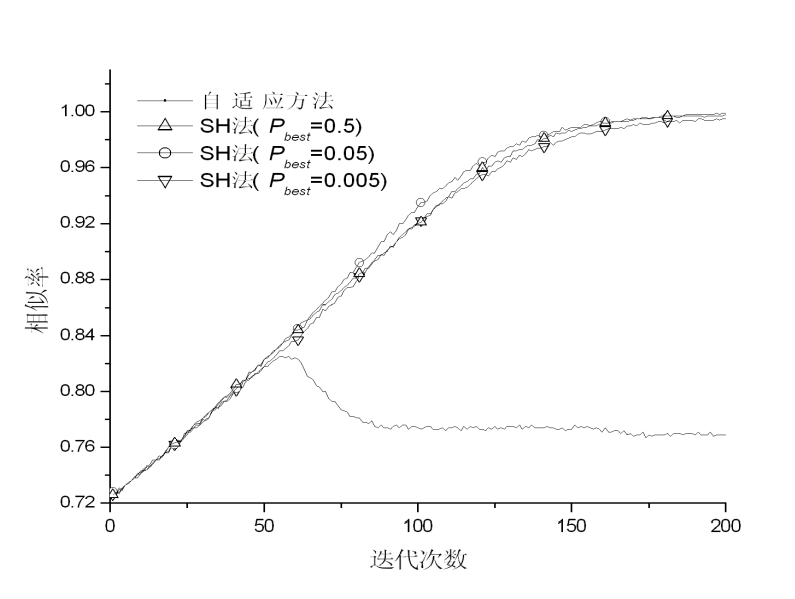


为了分析$P_{best}$对于蚂蚁搜索能力的影响，采用以下两个指标：

1）相似率（similarity ratio）：这个指标被广泛用于量化多样性，并且它已经在进化算法中应用。本文考虑文献中提出的指标：

$$\frac{\sum_{j=1}^{n}\sum_{i=1}^{n_a} s_j^i(\sum_{i=1}^{n_a} s_j^i - 1)}{(n_a-1)\sum_{j=1}^{n}\sum_{i=1}^{n_a} s_j^i} \tag{32}$$

由上式可见，当所有构造的解相同时，相似率为1；而如果蚂蚁构造的解完全不同时，相似率为0。




即使$P_{best}$取为0.005这样一个很小的量，相似率增加得很快，在150代达到0.98。

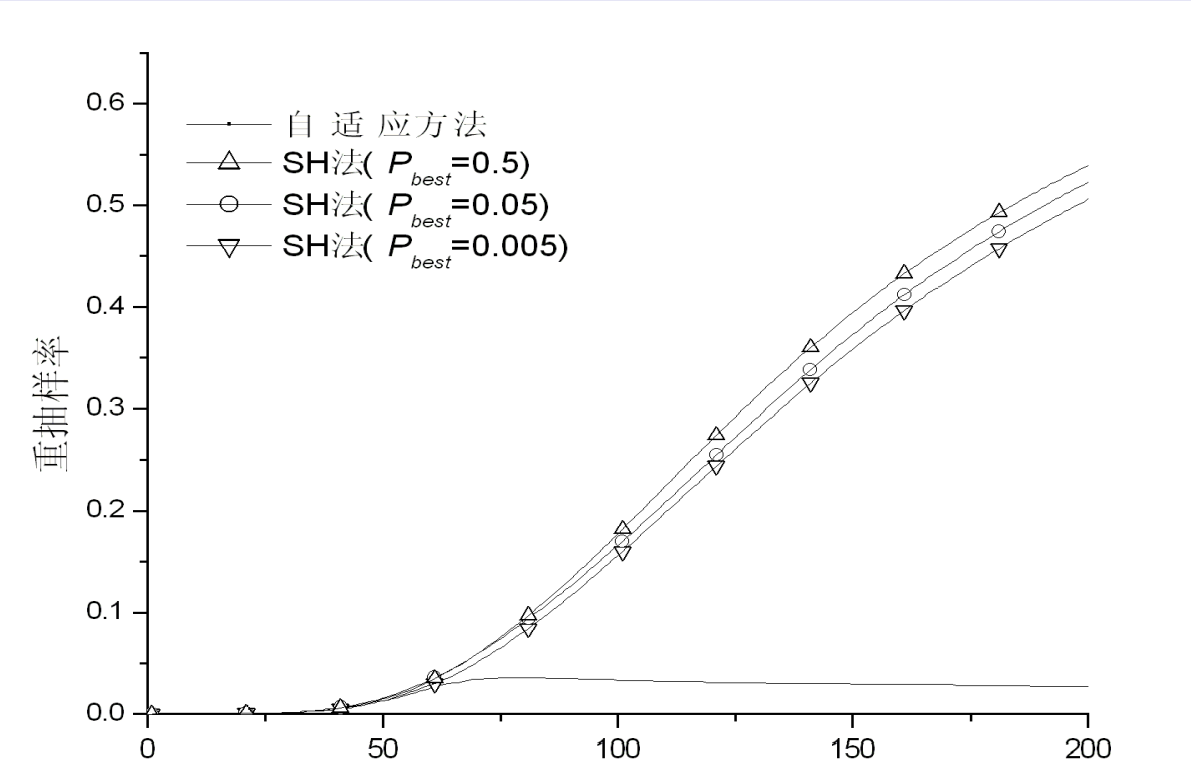
# 信息素下界的选取



2）重抽样率（re-sampling ratio）：它用来刻画算法在搜索空间中抽样的有效性。令$DiffNum$为到目前为止蚂蚁构造的不同解的数目，$TotalNum$为到目前为止蚂蚁构造的解的总数。重抽样率定义为

$$\frac{TotalNum - DiffNum}{TotalNum} \tag{33}$$

重抽样率接近于0意味着蚁群构造的解差异大，搜索能力强；而重抽样率接近于1意味着蚁群构造的解差异小，搜索接近停滞。



# 信息素下界的选取



2. 自适应方法

由信息素的释放方式可见，任意两个解之间的差异量即为解之间的Hamming距离（记为$d$）。设上一次迭代中用来更新信息素的最优解为，则平均差异量为

$$avg_D = \sum_{i=1}^{n_a} d(s^i, s^{best})/n_a \tag{34}$$

由于在多维背包问题中差异量是一个距离，因此也称平均差异量$avg_D$为平均距离（average distance），$n_a$是蚂蚁数，$s^i$是第$i$只蚂蚁构造的解。

# 信息素下界的选取



下面来分析平均差异量的渐近性质。如前所述，平均差异量是期望差异量的无偏估计。令$T_1 = \{\tau(u)^\alpha \eta(u)^\beta | s^{best}(u) = 1, u \in J\}$且$T_1 = \{\tau(u)^\alpha \eta(u)^\beta | s^{best}(u) = 0, u \in J\}$。假设$T_1$的最小值为$\tau_1$，$T_2$的最大值为$\tau_2$。一旦信息素被更新，$\tau_1/\tau_2$增加。特别地，当$\tau_1/\tau_2 \to \infty$，则在下一代构造最优解$s^{best}$的概率将趋近于1，即每个随机构造的解和之间的Hamming距离趋近于0。即下面的性质成立:

# 信息素下界的选取



下面来分析平均差异量的渐近性质。如前所述，平均差异量是期望差异量的无偏估计。令$T_1 = \{\tau(u)^\alpha \eta(u)^\beta | s^{best}(u) = 1, u \in J\}$且$T_1 = \{\tau(u)^\alpha \eta(u)^\beta | s^{best}(u) = 0, u \in J\}$。假设$T_1$的最小值为$\tau_1$，$T_2$的最大值为$\tau_2$。一旦信息素被更新，$\tau_1/\tau_2$增加。特别地，当$\tau_1/\tau_2 \to \infty$，则在下一代构造最优解$s^{best}$的概率将趋近于1，即每个随机构造的解和之间的Hamming距离趋近于0。即下面的性质成立:

**性质1.** 令$s$和$s^{best}$之间的Hamming距离的期望为$E(d(s, s^{best}))$, 则$E(d(s, s^{best})) \to 0$ 当且仅当$\tau_1/\tau_2 \to \infty$.

证明. 记$O = \{j \in J | s^{best}(j) = 1\}$，$K = |O|$。假设在构造$s$时，第$k$个被选中物品是$a_k (1 \le k \le K)$。注意到$E(d(s, s^{best})) \to 0$当且仅当$P(a_k \in O) \to 1 (1 \le k \le K)$。由于$s$ 按照3.5.2 中的过程构造，结论成立□

# 信息素下界的选取



**性质1表明如果期望差异量过小，则信息素的差异很可能过大。**在这种情形下，有必要减少信息素之间的差异。

根据3.2 节的自适应方法，信息素下界按照以下方法选取：

当找到一个新的$s^{gb}$时，$\tau_{min}$被初始化为一个很小的值。这可以通过设置$\tau_{min}$为$\epsilon\tau_{max}$且$P_{best}$被设置一个很大的数（如$P_{best} > 0.5$）来实现。此后，如果平均差异量过小，这意味着信息素差异可能过大。为了避免这种情形，可用如下方式来修正信息素的下界：

$$\text{如果} avg_D < \gamma, \text{则} \tau_{min} := \lambda\tau_{min} \tag{35}$$

其中$\gamma$是一个正参数（$1 < \gamma < n$），$\lambda(\lambda > 1)$是一个参数。

# 信息素下界的选取



与SH法不同的是，所提出的方法能在平均差异量过小时自适应地更新信息素的下界。在这种方法中，信息素下界被初始化为一个很小的值，这样在算法运行之初信息素的上下界差异较大。而一旦式（3-18）的前提条件满足，$\tau_{max}/\tau_{min}$逐步减少。而且，$\tau_{max}/\tau_{min}$能足够大以使信息素能起到引导蚂蚁搜索的作用。阈值$\gamma$决定信息素的上下界差异。当它设置为一个很大的值，信息素的上下界差异会在一定代数之后变得很小。而当它设置为0 时，信息素的下界不改变。

# 信息素下界的选取



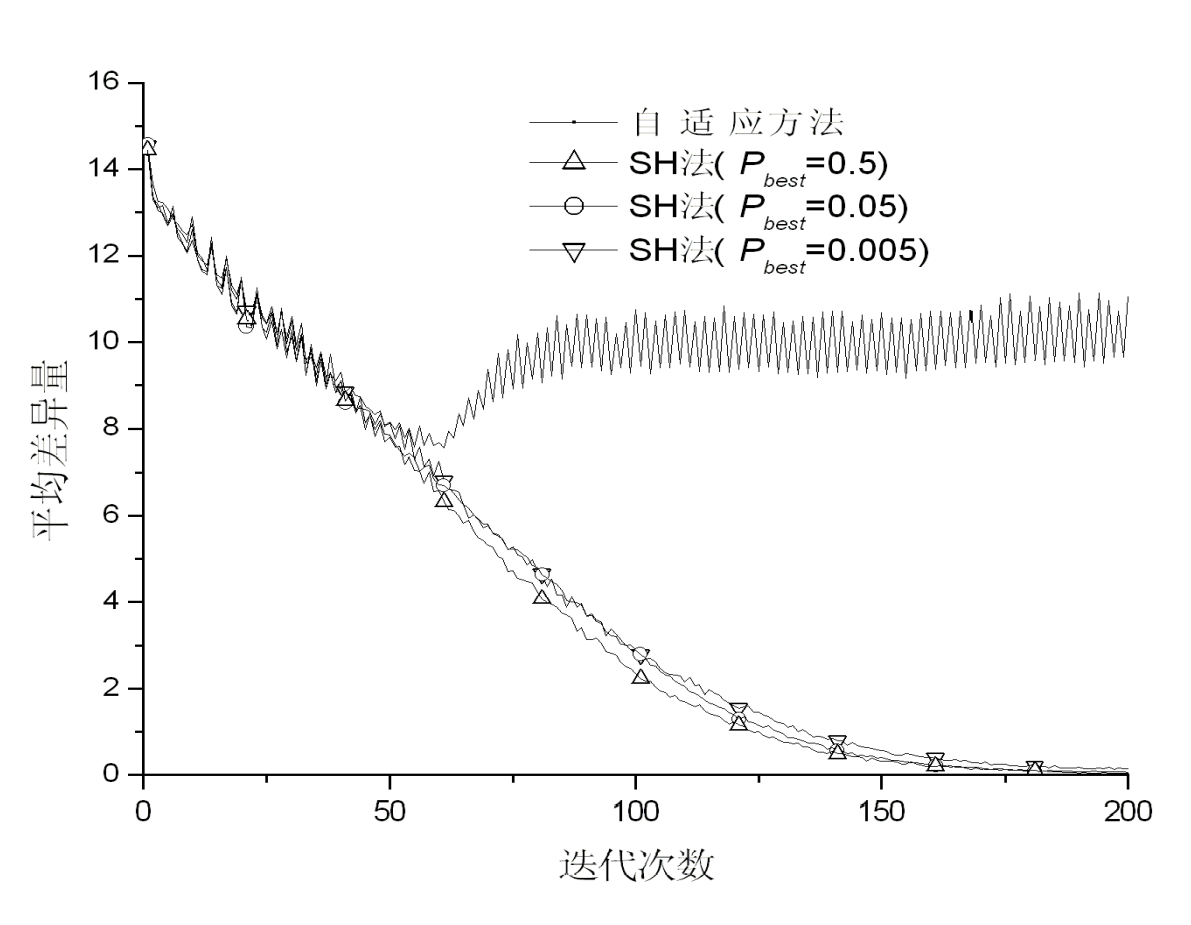



图: 平均差异量（average difference quantity）的演变（自适应方法（the adaptive method）和SH法在算例10.100.00上的平均结果)。参数设置为：$n_a = 50, \alpha = 1, \beta = 20, \rho = 0.95, \gamma = 8, \lambda = 2, P_{best} = 0.9$.

# 信息素下界的选取



相似率和重抽样率也用来分析自适应方法。在自适应方法中，这两个量都比SH法中的小，这表明蚂蚁有更好的搜索能力。图3-3给出了自适应方法和SH法的平均差异量变化曲线。对于自适应方法，平均差异量始终大于或接近于$\gamma$（其中$\gamma = 8$）。而对于SH法，平均差异量在开始的时候较大，而在150代以后，平均差异量很小（小于1），这再一次表明蚂蚁仅在很小的区域搜索。

# 实验数据及算法设计说明

为了检验AMMAS和局部搜索的性能，采用文献中提供的算例进行实验。每个问题表示为$m.n.z$，其中$m$和$n$分别是约束和物品的数目，$z$表示该算例在具有相同数目的物品和约束的算例组中的序号。

依据Stützle 和Hoos的建议，用混合策略来调度当前最优解和本次迭代最优解来更新信息素。令$f^{gb}$表示每隔$f^{gb}$代$s^{gb}$被用来更新信息素。调度策略为：在前9代，$f^{gb}$为3，从第10代到第24代，$f^{gb}$为2，从第25代起，$f^{gb}$为1。信息素的初始值为某个非常大的值。为了统计分析，对每个算例测试50次。

# 性能评价

对于随机算法，评价计算结果是一个难点。虽然算法的收敛性保证了算法在无穷大时间内总能找到最优解，但是实践中计算时间是有限的。本章采用文献中的方法，它先通过求解多维背包问题的线性规划松弛问题获得原问题一个上界，再通过比较随机算法求得的结果与问题的上界的差距（gap）进行评价。由于文献中的GA算法计算结果与问题上界的差距很小，因此GA是一个好的算法。所以与它进行比较，能判断AMMAS的计算有效性。

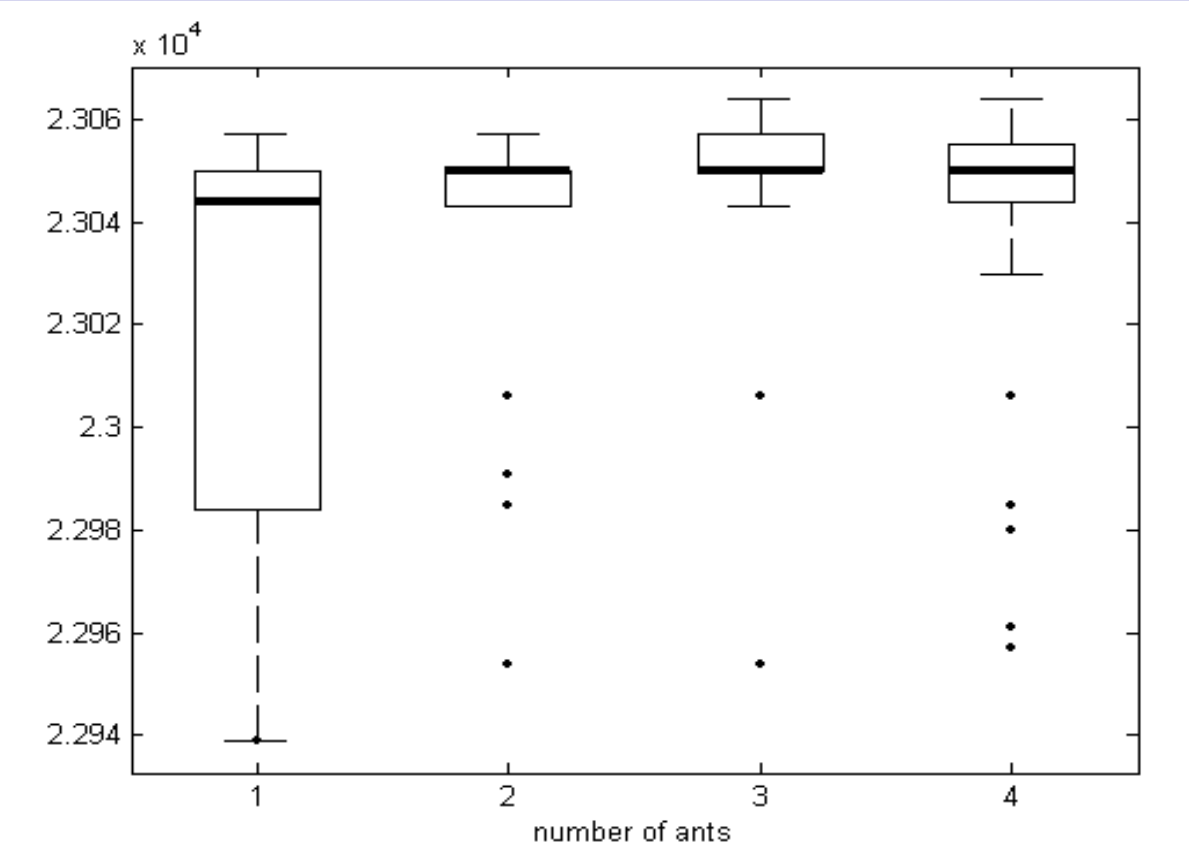
# 参数选取及作用



AMMAS在构造10000 个解之后停止。下面用数值实验分析参数的影响。在实验中，参数的缺省值为：$n_a = 50, \alpha = 1, \beta = 20, \rho = 0.95, \gamma = 8, \lambda = 2, P_{best} = 0.9$。在每次实验时，仅一个参数变化，其他参数保持不变，被测试的参数取值为$n_a$ ∈10, 20, 50, 100, $\alpha$ ∈0.5, 1, 2, 4, $\beta$ ∈1, 5, 10, 20, 40, $\rho$ ∈0.7, 0.8, 0.9, 0.95, 0.98, $\gamma$ ∈2, 4, 8, 16, $\lambda$ ∈1.5, 2, 2.5, 3, $P_{best}$ ∈0.6, 0.7, 0.8, 0.9。我们用10.100.00为例来进行分析。对于每个参数值，有50 个样本（即50个总收益）。

# 参数选取及作用






图显示的是蚂蚁数对于算法性能的影响。从左至右（即从第1个箱形图到第4个箱形图），四个箱形图依次给出的是蚂蚁数为10、20、50和100时的实验结果。**当蚂蚁数为50时，中位数最大并且最大值和最小值差距较小，此时计算结果分布较集中，表明算法能较好地收敛。**

# SH法和自适应方法的实验分析



**表：** 自适应方法与SH法的实验结果。前十个算例的物品数为100，约束数为5；中间十个的物品数为100，约束数为10；最后五个的物品数为500，约束数为5。平均值用四舍五入取整

| 算例 | 已知最大值 | SH法 | | | 自适应方法 | | |
|---|---|---|---|---|---|---|---|
| | | 最大值 | 平均值 | 标准差 | 最大值 | 平均值 | 标准差 |
| 5.100.00 | 24381 | 24381 | 24354 | 26.1 | 24381 | 24362 | 23.8 |
| 5.100.01 | 24274 | 24274 | 24268 | 15.9 | 24274 | 24273 | 6.2 |
| 5.100.02 | 23551 | 23551 | 23531 | 8.5 | 23551 | 23540 | 7.2 |
| 5.100.03 | 23534 | 23534 | 23477 | 13.1 | 23534 | 23482 | 14.9 |

# 与文献中的蚁群算法的比较

**表：** AMMAS、Ant-knapsack和$ACO_{LM}$实验结果。前十个算例的物品数为100，约束数为5；中间十个的物品数为100，约束数为10；最后五个的物品数为500，约束数为5。平均值用四舍五入取整

| 算例 | 已知最大值 | AMMAS | | | Ant-Knapsack | | |
|---|---|---|---|---|---|---|---|
| | | 最大值 | 平均值 | 标准差 | 最大值 | 平均值 | 标准差 |
| 5.100.00 | 24381 | 24381 | 24362 | 23.8 | 24381 | 24342 | 29.3 |
| 5.100.01 | 24274 | 24274 | 24273 | 6.2 | 24274 | 24247 | 38.5 |
| 5.100.02 | 23551 | 23551 | 23540 | 7.2 | 23551 | 23529 | 8.0 |
| 5.100.03 | 23534 | 23534 | 23482 | 14.9 | 23534 | 23462 | 32.6 |

# 与文献中的蚁群算法的比较



表6给出了三种算法的结果。由结果可见，AMMAS的结果最好，Ant-knapsack次之。

另一方面，AMMAS、ACOLM和Ant-knapsack构造解的总数分别是：10000、10000和造解的总数最少。因此，AMMAS能很快地找到高质量的解。

# 与文献中的蚁群算法的比较



最后分析混合AMMAS的性能。选择文献[121]中最大的90个算例来测试算法性能。这90个算例由500个物品和不同数目的约束构成。每30个算例为一组，每组用记号表示为。对于5.500、10.500和30.500，最大运行时间分别是100秒、200 秒、400 秒（CPU为2.8GHZ）。前期实验表明，算法能在相应的时间内获得的解是令人满意的。需要说明的是，为了公平比较混合AMMAS和AMMAS，将最大运行时间作为停止条件。

首先，分析局部搜索对算法的影响。表7给出了AMMAS和AMMAS+ls的实验结果，并且给出了不考虑信息素时（此时）AMMAS+ls的实验结果。由结果可见，局部搜索能有效改善AMMAS的性能，AMMAS+ls在所有算例中都取得最好的解；并且，使用信息素能引导蚂蚁找到更好的解，AMMAS的性能优于没有使用信息素的混合算法。

**表：** AMMAS+ls和AMMAS的实验结果。物品数为500，约束数为5。平均值用四舍五入取整

| 算例 | AMMAS+ls ($\alpha = 1$) | | AMMAS+ls ($\alpha = 0$) | | AMMAS | |
|---|---|---|---|---|---|---|
| | 最大值 | 平均值 | 最大值 | 平均值 | 最大值 | 平均值 |
| 5.500.00 | 120148 | 120111 | 119860 | 119648 | 120116 | 120056 |
| 5.500.01 | 117879 | 117841 | 117494 | 117360 | 117857 | 117786 |
| 5.500.02 | 121131 | 121097 | 120708 | 120526 | 121109 | 121043 |
| 5.500.03 | 120804 | 120776 | 120473 | 120293 | 120785 | 120715 |
| 5.500.04 | 122319 | 122303 | 121988 | 121812 | 122319 | 122254 |

# 与两个优秀算法的比较



表8给出了的三个算法的实验结果。与GA相比，AMMAS+ls在每一组都找到了更好的解；而与z*相比，AMMAS+ls在8组中找到了更好的解。

**表：** AMMAS+ls，GA和Z*的实验结果。计算结果用四舍五入取整

| 算例组 | Tightness率 | AMMAS+ls | GA | z* |
|---|---|---|---|---|
| 5.500 | 0.25 | 120629 | 120616 | 120623 |
| 5.500 | 0.5 | 219509 | 219503 | 219507 |
| 5.500 | 0.75 | 302362 | 302355 | 302360 |
| 10.500 | 0.25 | 118603 | 118566 | 118600 |
| 10.500 | 0.5 | 217309 | 217275 | 217298 |
| 10.500 | 0.75 | 302588 | 302556 | 302575 |
| 30.500 | 0.25 | 115541 | 115470 | 115547 |
| 30.500 | 0.5 | 216223 | 216187 | 216211 |
| 30.500 | 0.75 | 302406 | 302353 | 302404 |