关于求解难组合优化问题的蚁群优化算法

About ACO Algorithm For Hard Combinatorial Optimization Problem

李有梅[1,2] 王文剑[1,2] 徐宗本[1]

(西安交通大学理学院 西安710049)[1](山西大学计算机科学系 太原030006)[2]

Abstract Ant colony optimization(ACO)is a class of constructive metaheuristic algorithm, which took inspiration from the observation of ant colonies foraging behavior and have been applied to the solution of difficult combinatorial optimization problems. The algorithm constructs solutions on the basis of information provided both by problem specific characteristic and by previously constructed solutions. This paper overviews recent work on Ant Colony Optimization. It is composed of three parts . The first part overviews the related background and the basic biological findings on real ants; the second frames the ACO approach; while the third focus on the most important aspects and performance of the ACO metaheuristics. A brief conclusion is given in the last.

Keywords Ant colony optimization, Metaheuristics, Combinatorial optimization

1. 引言

组合优化问题在规划、调度、资源分配、决策等工程问题中有着非常广泛的应用。在问题规模较小时,可以使用分支定界法或动态规划方法等来求解。当问题规模增大时,解的数目虽然有限,但呈指数增长,要在合理时间内求得准确的最优解实际上已不可能。为此,人们设计了各种启发式算法。近年来,最重要和最有希望的一个研究领域是构造"师法自然"的启发式。它们类比社会系统、物理系统、生物系统等的运行机制,设计算法在问题的解空间中进行非确定性搜索。典型的有遗传算法(GA)、模拟退火(SA)、人工神经网络(ANN)。这些算法由于其自适应性,对难组合优化问题的求解取得了好的结果,被广泛应用于工程优化和控制中。本文将要介绍的蚁群优化算法,由于其较强的自适应性和对问题状态的学习能力,正逐步成为一种新的有潜力的优化算法。

1991年,M . Dorigo 等人,受蚂蚁搜索食物行为的启发,提出了一种新的算法,Ant System ,后来以此为基础逐步完善成蚁群优化算法,一种新的启发式算法框架(Ant Colony Optimization, A New Meta-heuristics,简记为 ACO)。Metaheuristics 是一类启发式算法的总的框架描述,它利用收集到的关于问题的信息,指导算法在解空间进行非确定性搜索。

蚁群作为一种社会昆虫,其食物搜索行为具有非常高的协作性。研究人员在对蚂蚁行为的观察中发现,它们总能找到一条巢穴到食物源之间的最短路,这条最短路并不是单个蚂蚁所能找到的,它是蚂蚁群体协作的结果。当蚂蚁在食物源和巢穴之间行走时,会在路上存储一种称为信息素的物质,这些物质形成一条指示轨迹,蚂蚁可以感知到环境中这种物质的存在及其强度,并在选择移动时,倾向于选择信息素密度高的方向。最终蚂蚁会从初始的随机路径搜索,逐步稳定到最短路径上来。昆虫对环境的刺激物作出反映,并产生新的刺激物,这些刺激物既作用于自己,也对群体中的其他个体产生影响,利用这种间接的通信方式,就形成了一种行为协作方式。这种以环境状态的物理修正为媒介,而且信息只能被局部接收的通信方式称为 STIGMERG COMMUNICATION。

ACO 算法将这一协作机制模型化,将问题的求解表述为最短路搜索。但这种方法并不适用于那些能用经典算法(如动态规划或标号法)求解的多项式问题。它能够对那些规模巨大的或者问题状态随时间改变的难组合问题给出较满意的解。ACO 算法由于它所引入的独特的通信协作机制,日益引起人们的重视。特别值得一提的是 Dorigo 维护的站点: http://irida. uib. ac. be/ants98/ants98. htm 提供许多关于蚁群算法方面的资料。

国内对 ACO 算法的研究刚刚开始引起人们的注意。这篇文章的目的正是希望读者能够认识蚁群优化算法及其特性,并展开进一步的研究。

2. ANT SYSTEM 和组合优化问题

ANT SYSTEM(AS)是第一个受蚁群行为激发而设计的算法,而且是以后许多 ACO 算法推广的基础。为了能清楚地理解 ACO 启发式算法框架,我们以 TSP 问题为例,具体介绍 AS 及其扩充 MMAS (MIN-MAX ANT SYATEM)和 ACS (ANT COLONY SYSTEM)。

TSP 是一经典的难组合优化问题,因其典型性已成为各种启发式搜索、优化算法的间接比较标准。TSP 问题一般定义如下:

给定 N 个结点(用来代表城市)和连接结点的弧段的集合 E,对任意 $i\in N, j\in N, (i,j)\in E$,用 d_{ij} 表示弧段的长度,即城市 i 和 j 之间的距离,TSP 问题就是在图 G=(N,E)上找一条最短的 HAMILTON 回路,即一条闭回路,过每个结点一次且只一次,其长度为组成它的各弧段长度的和。

2.1 人工蚂蚁模型

AS 中引入了一群简单智能体,我们称之为蚂蚁,这群蚂蚁相互协作去求一条最短 HAMILTON 回路。蚂蚁,一方面它是现实蚂蚁的抽象,另一方面,根据实际需要丰富了它的能力,这些能力现实蚂蚁并不具备。归纳起来有如下的一些特征:

与实际蚂蚁的相似性:

1)每个人工蚂蚁可以逐步构造问题的可行解,正象单个蚂蚁可以发现食物源和巢穴之间的路,但好的解(对应最短路)是一群蚂蚁协作的结果。

2)人工蚂蚁利用数值存储关于问题状态的信息,模拟现实蚂蚁留下的信息素。这些数值保存了蚂蚁过去的行为信息,

是关于问题的局部的状态信息。

3)真实的信息素会挥发,对人工信息素也引入了挥发机制。

4)真实蚂蚁走过相邻的区域,人工蚂蚁一步步走过问题的相邻状态。

5)人工蚂蚁和真实蚂蚁一样,感知所处局部状态的信息素密度和其他启发信息,按照某种概率决策规则在相邻状态间移动。

与真实蚂蚁的不同之处:

1)它存在于离散的状态中,它们的运动是从一个状态到另一个状态的转移。

2)它有内部状态,有记忆性,可存储过去的行为。

3)它在某一确定的时刻存放信息素,信息素的密度与它所发现的解的质量有关。

2.2 蚂蚁的路线选择规则

对 TSP 问题,给定城市之间的关系图 G=(N,E),每一个结点(城市)i 为一个局部状态,任一个状态 j 若与 i 有弧段相连,称 j 为 i 的邻接状态。蚂蚁从当前状态移动到另一个相邻状态,直到发现一条满足条件的回路为止。

AS 运行时,在任意时刻 t,系统生成 m 只蚂蚁,它们随机分布在结点 i 上。设第 k 只蚂蚁当前所在的结点为 i,如何选择下一个要移向的结点是算法的关键。首先,j 必须是 i 的相邻结点,记 i 的相邻结点集为 N^i。其次要满足约束条件,每个结点只能被访问一次。因此,引入禁忌表 $Tabu_k(i)$记录第 k 只蚂蚁到目前为止已走过的结点,以便阻止它们被再一次访问。在状态 i,可被访问移向的结点集记为 $N_k^i=N^i-Tabu_k(i)$,它中一般有多个元素,蚂蚁采用某种概率决策规则选择下一个结点 j。我们以 $p_{ij}^k(t)$表示第 k 只蚂蚁 t 时刻从 i 移向 j 的概率。$p_{ij}^k(t)$的大小,受到结点 i 处的局部状态信息的影响。它的局部状态信息有两种,一是和问题相关的启发信息 η_{ij},它用来表示从 i 移动到 j 的期望程度。在 TSP 问题中,η_{ij}可取为两相邻结点间距离 d_{ij}的倒数。d_{ij}越小,移向 j 的可能性越大。另一种信息是过去 t−1个时刻收集的关于问题状态信息的描述,即信息素 $\tau_{ij}(t)$,$\tau_{ij}(t)$的大小代表信息素密度的高低,从而影响 $p_{ij}^k(t)$的取值。我们将这两种信息综合保存在 i 的路线决策表 $A_i=[a_{ij}(t)]$(共$|N^i|$维)中。

在 AS 中,$a_{ij}(t)$按下式取值:

$$a_{ij}(t)=\frac{(\tau_{ij}(t))^{\alpha}(\eta_{ij})^{\beta}}{\sum_{l\in N^i}((\tau_{il}(t))^{\alpha}(\eta_{il})^{\beta}},\quad \forall\ j\in N^i \tag{1}$$

这里 α、β 表示相对权重参数。控制 η_{ij}和 $\tau_{ij}(t)$在决策中所占的比重。

对于蚂蚁 k,在 t 时刻选择从当前结点 i 移到结点 j,$j\in N_k^i$ 的概率为:

$$p_{ij}^k(t)=a_{ij}(t)/\sum_{l\in N_k^i}a_{il}(t)=(\tau_{ij}(t))^{\alpha}(\eta_{ij})^{\beta}/\sum_{l\in N_k^i}(\tau_{il}(t))^{\alpha}(\eta_{ij})^{\beta} \tag{2}$$

2.3 信息存储和挥发

经过|N|−1步选择,所有蚂蚁完成一个回路,我们得到关于问题的 m 个可行解。从中学习关于问题状态的信息,调整信息素的值。当蚂蚁原路返回时,在它所经过的弧段上留下信息素,用 $\Delta\tau_{ij}^k(t)$表示第 k 只蚂蚁在弧段(i,j)上存放的量。它的大小,与它发现的可行解 $T_k(t)$质量有关,设 $L_k(t)$表示该回路的长度,显然 $L_k(t)$越小,解的质量越好,在所经过的弧段上留下的信息素 $\Delta\tau_{ij}^k(t)$越大。取

$$\Delta\tau_{ij}^k(t)=\begin{cases}1/L_k(t) & \forall\ (i,j)\in T_k(t)\\ 0 & \text{其他}\end{cases} \tag{3}$$

则弧段(i,j)上信息素总的改变量为

$$\Delta\tau_{ij}(t)=\sum_{k=1}^{m}\Delta\tau_{ij}^k(t) \tag{4}$$

此外,AS 引入信息素挥发机制。若挥发比例为 ρ,则保留下来的信息素比例为(1−ρ)。总的信息素按(5)式调整:

$$\tau_{ij}(t)\leftarrow(1-\rho)\tau_{ij}(t)+\Delta\tau_{ij}(t) \tag{5}$$

至此一次迭代结束。重新生成 m 只蚂蚁和调整路线决策表 A_i,重复进行上述过程,直到求得满意的解或迭代到一定次数为止。文[1]给出 AS 算法的详细描述。

2.4 对 AS 的扩充

由于 AS 的实验结果很令人鼓舞,随后出现了许多算法,对 AS 进行扩充和完善。Stützle 和 Hoose 在1997年提出 MMAS。与 AS 的重要差异体现在对信息素的处理上:

1)引入一个总控模块,称之为精灵模块(daemon)。它收集 m 只蚂蚁构成的解,并挑选出最优解。利用最优解调整计算信息素的改变量。只有当前最优解经过的弧段才能有一个改变量 $\Delta\tau_{ij}(t)$,其他弧段的改变量为零。这种信息素调整方式称为离线调整。

2)信息素的取值限制在区间$[\tau_{\min},\tau_{\max}]$,利用一种平滑机制,缩小弧段上信息素分布的相对差距,这有利于蚂蚁探索新的解,避免停滞现象和产生局部最优解。

在其他方面,MMAS 与 AS 相同。详细内容参看文[3]。

ACS 是 Dorigo 和 Gambardella 于1996年提出的。它基于 AS,但又有重要差别:

1)精灵模块离线调整信息素,算法的每一迭代步结束时,精灵模块寻找生成历史最优解的蚂蚁,在其所经过的弧段上调整信息素。同时在这一模块内加入局部优化技术,如3-opt 搜索程序,去改进蚂蚁生成的解,然后再离线调整信息素。规则为

$$\Delta\tau_{ij}(t)=1/L^+,\forall\ (i,j)\in T^+ \tag{6}$$

T^+是从开始到目前为止的最优解,L^+为其长度。

2)蚂蚁采用与 AS 不同的决策规则,设 q 是一个均匀分布在[0,1]上的随机变量,$q_0\in[0,1]$是一可调参数,第 k 只蚂蚁在结点 i 按下述规则选择移向结点 j:

如果 $q\leqslant q_0$,则

$$P_{ij}^k(t)=\begin{cases}1 & \text{如果 } j=\underset{l\in N_k^i}{\operatorname{argmax}}a_{il}(t)\\ 0 & \text{其他}\end{cases}$$

如果 $q\geqslant q_0$ 则按公式(2)计算 $P_{ij}^k(t)$ (7)

这个规则有两个参数。当 $q\leqslant q_0$,决策规则利用已知的关于问题的信息,选择 $a_{ij}(t)$最大的结点移动。若 $q\geqslant q_0$,它进行有偏向性的搜索。调整 q_0可以调节系统在解空间进行新的随机搜索的程度。q_0越大,在已获得的最优解附近进行搜索的可能性越大,对解空间的探索就越小,很容易陷入局部最优。因此,需要在搜索(Exploration)和利用(Exploitation)之间权衡,以保证得到满意的解。

3)ACS 同时进行逐步信息素调整,增加生成的解的多样性。

4)ACS 还采用候选表策略。当 N_k^i 很大时,不考察候选表以外的结点,提高求解的速度和性能。关于 ACS 请参看文[3]。

在最近的几年中,又相继出现了许多仿蚁群行为的算法。1997年,Dicaro 和 Dorigo 提出 ANTNET 算法用于网络路由

问题(Network Routing),1998年 Gambardella 提出 HAS 算法求解运输线路问题(VRP,Vehicle Routing Problem)等等。

3. ACO 启发式算法框架的模型与实现

目前有很多仿蚁群行为而设计的算法,这些算法对实际问题产生了成功的应用。纵观这些用于不同问题的算法,Dorigo 和 Dicaro 给出了此类算法的一个总的框架描述,称之为蚁群优化算法(ANT COLONY OPTIMIZATION,ACO)。本节将概括给出 ACO 的算法描述和设计分析。

3.1 ACO 算法的描述

ACO 算法的核心是要把一个组合优化问题表示为一个图上的最短路搜索问题。抽象地讲,一个组合优化问题可由下述方式来定义:

1)给定一个有限的基本元素集 C,$|C|=N_c$。

2)给定元素间的所有可能的连接组成的有限集 $L=\{l_{c_i c_j} | c_i, c_j \in C\}$,$|L| \leqslant N_c^2$。

3)对每一个连接 $l_{c_i c_j}$,定义连接费用 $J_{c_i c_j}$,费用可能和时间参数 t 相关。

4)约束条件集 $\Omega=\Omega(C,L,t)$。

给定组合优化问题以上述方式定义的图表示 G=(C,L,Ω),优化问题的解就可以表示为 G 上的一可行序列(可行路),它的最优解即为 G 上的满足约束条件的最短路。

针对上述表示,图1给出 ACO 启发式算法框架的伪代码。

```
1  ACO 主程序模块()
2   While(结束标准不满足)
3    调用模块
4    蚂蚁生成和活动模块()
5    信息素挥发模块()
6    精灵控制模块()
7   调用完毕
8  While 循环结束
9  主模块结束

10 蚂蚁生成和活动模块()
11  While(资源可利用时)
12   调用蚂蚁生成模块()
13   调用蚂蚁活动模块()
14   While 循环结束
15  本模块结束

16 蚂蚁活动模块()
17 蚂蚁初始化()
18 M=调整蚂蚁记忆()
19 While(当前状态≠目标状态)
20  A=读入局部路线决策表()
21  P=计算状态转移概率(A;M;,)
22  下一个状态=实施选择策略(P;,)
23  移动到下一个状态
24  IF(采用逐步信息素调整)
25   在访问的弧段上存储信息素
26   调整路线决策表
27  IF 模块结束
28  M=调整蚂蚁记忆状态()
29 While 循环结束
30 IF(采用在线延迟调整)
31  计算解的适应值
32  调整存储信息素()
33  调整路线决策表()
34 IF 模块结束
35 蚂蚁生命周期结束
36 蚂蚁活动模块结束.
```

图1 ACO 主要模块示意程序

粗略地讲,ACO 算法中,蚂蚁的行为可总结如下:一群蚂蚁同步或异步在问题的相邻状态之间移动,它们利用包含在每个状态中的局部信息,采用随机决策规则选择移动方向,逐步构造出问题的可行解。一旦蚂蚁完成了解的构造,就根据获得的解的信息调整信息素。精灵模块的引入,从全局角度实现对问题解的集中评价并设计响应的行为,它的功能不能由单个蚂蚁来完成,如发现全局最优解或本步迭代的最优解,或激活局部优化程序等。用不同的方法对 ACO 算法总框架的各部分进行具体化,可以生成不同的 ACO 算法。

3.2 ACO 算法的设计分析

从上节的描述中我们可以看出,当给出问题的合适表示方式,也即明确定义了问题的状态和状态之间的相邻关系,影响 ACO 算法性能的主要是信息素调整规则、状态转移规则和相关参数的设置。这一节我们分别就以下几个方面进行详细讨论。

3.2.1 *局部启发信息* η_{ij}是关于问题的先验信息。大多数 ACO 算法将局部启发信息与信息素相结合去生成 i 的路线决策表 A_i。(见(1)式)。利用 η_{ij}可以显著改善算法的性能。但结合方式并不局限于(1)式。在文献中还可看到如下的结合方式:

$$\alpha_{ij}(t)=\frac{\alpha\tau_{ij}(t)+(1-\alpha)\eta_{ij}}{\sum_{l\in N_i}\alpha\tau_{il}(t)+(1-\alpha)\eta_{il}} \tag{8}$$

公式中的参数 α、β 起到调整 η_{ij}与 $\tau_{ij}(t)$之间相对重要性的效果。

3.2.2 *信息素* $\tau_{ij}(t)$*的调整机制* ACO 算法的一个主要部分就是蚂蚁间的相互通信,一群蚂蚁协作发现问题的一个好的解。信息素 $\tau_{ij}(t)$便是蚂蚁之间的通信管道。通过对逐步迭代时得到的解的评价,修正对问题状态的信息素表示。信息素的调整机制极大影响着算法的性能。不管采用哪种调整方式,应存储多少信息素 $\Delta\tau_{ij}^{k}(t)$是一个非常关键的问题,否则会导致局部最优或出现停滞现象。

3.2.3 *随机决策规则* (2)式给出的决策规则称为随机比例决策规则。(7)式给出的决策规则称为伪随机比例决策规则。弧段上的信息素是已获得的对问题状态的描述,是后验信息。在(2)式中,α 越大,蚂蚁越倾向选择其他蚂蚁使用的弧段,也体现了蚂蚁之间的协作;α 越小,先验信息 η_{ij}所占的比重相对增加,规则越接近贪婪规则。因此要选择合适的随机决策规则,以协调搜索与利用之间的关系。

3.2.4 *参数的设置* 不同的 ACO 算法其中参数的个数是不同的。目前也没有理论结果可用来指导参数的设置。只能通过大量的实验对参数的取值进行研究。

3.2.5 *与其他优化算法相结合* ACO 算法很容易与其他算法相结合。一般情况下,ACO 算法直接利用蚂蚁构造的解去调整信息素水平,当然也可以利用这些解作为初始解,使用象 TS、SA 、K-OPT 等算法对其进行优化。用优化后的解去调整信息素水平,已有实验表明了它的有效性。

3.3 有关的实验结果

目前对 ACO 算法的实验结果在文献中有很多报道。M.-Dorigo 在文[4]中比较了 ACS 与其他启发式的计算性能,实例是对称 TSP 问题。目前对算法的研究多数停留在实验阶段。从实验结果看,算法的性能可以和其他的通用求解算法的性能相并提,有时还要好一些。这里就不再列出,可参考文[1,2,4,5]。

4. ACO 算法的特点及存在的问题

ACO 算法是一个新的非常有前途的研究领域,它是人工生命与运筹学的交叉。它有许多独特的性质,还有许多需要完善的地方。本节就 ACO 算法的特点和存在的问题展开讨论。

4.1 与演化算法的比较

我们已经看到，ACO 算法是基于群体的仿生算法，本质上是并行的。它与基于群体的演化算法有很大相似性。它们都利用群体来收集关于问题的知识，再据此来指导随机生成新的群体。但它们之间也存在重要差异，因为在演化算法中，知识被保存在当前的群体中，而蚁群算法则通过信息素方式存储过去学得的知识，实现长期记忆。在演化算法中，Baluja 和 Caruana 的增量学习算法（Population Based Incremental Learning，PBIL）[6]与 ACO 更为接近。在 PBIL 中，引入一个实向量，称为生成向量，用来指导生成新的群体，反过来，再利用群体中的解来调整生成向量中各分量的值。很明显，生成向量起到与 ACO 中的信息素相似的作用，但主要差别仍存在，即生成向量中各分量的度量是独立的，因此它只适应解分量可分离的问题情形。

从算法的描述中，我们可以看到，ACO 通过存储信息素方式指导选择方向，利用概率选择机制实现了解的变异，同时问题的约束在解的构造过程中得到了满足。这些特点解决了演化算法中对杂交算子和变异算子设计上的困难，避免了约束处理上的麻烦。ACO 算法使得问题的求解思想更简洁和易于实现。

4.2 算法的特点

蚁群优化算法作为一种基于群体的仿生算法，除具有一般模拟演化算法所具有的全局最优性、并行性、稳健性以外，还有如下两个特征：

4.2.1 *个体间独特的通信协作方式* ACO 算法的成功在于它借助信息素 $\tau_{ij}(t)$，实现了一种有效的学习机制。信息素是蚂蚁之间通信的管道，通过它实现了群体知识的利用。信息素的改变反映的正是蚂蚁对问题状态认识的改变，也就是对问题认识的逐步深入。信息素对问题的状态信息实现了长期记忆，所以可分阶段对问题进行求解，而不需要每次重新开始，也可以从中导出其他有用的结论。

4.2.2 *与问题实例信息的耦合* $\tau_{ij}(t)$与 η_{ij}的结合是算法成功的另一个要素。η_{ij}的构造方式对算法的性能产生着影响，η_{ij}构造得越精细，反映问题的特征便越多，算法的性能也就越好。在 TSP 问题中，η_{ij}可以简单地定义为距离的倒数，$\eta_{ij}=1/d_{ij}$，它也可以是利用某种下界估计法对选择(i，j)转移的部分解的费用估计。Maniezzo 在1998年提出的 ANTS 算法中就使用了这种方法。因此就先验的启发信息而言，算法有很大的拓展空间，可以将问题有关的已知的数学结论引入到算法中，以提高算法的性能。

4.3 算法的缺陷

蚁群优化算法表现出两个主要缺陷：

1. 与其他方法相比，该算法一般需要较长的搜索时间。蚁群算法的复杂度可以反映这一点。解的构造过程会占用大部分的计算时间。

2. 容易出现停滞现象。所谓停滞就是蚂蚁停止去搜索新的解，尽管利用了随机搜索规则。这是因为蚂蚁倾向沿着信息素密度高的弧段移动。在调整规则中我们看到，不被使用的弧段与包含在好解中的弧段相比，与它们的信息素密度的差异越来越大，被选择的概率减小，导致算法只能在目前的最优解附近搜索，即加强对所学知识的利用。这种学习方式实现了正反馈机制或强化学习机制。停滞现象是这种学习方式要避免的一个缺陷。所以在算法的求解过程中，需要在扩大搜索和加强利用之间折衷。信息素挥发机制或其他信息素差异平滑的策略的使用都有利于开拓新的搜索空间，跳出局部最优状态。

4.4 关于 ACO 算法的研究方向

ACO 算法是一种新型的模拟演化算法，其研究刚刚开始，远未象 GA、SA 那样形成系统的分析方法和一定的数学基础，有许多问题有待进一步研究，主要有以下三个方面：

4.4.1 *适用问题的确认* 发掘 ACO 相关和适用的研究领域是首要的一个研究方向。ACO 算法目前最成功的应用是在组合优化问题中。但它显然不会限于这一领域。ACO 算法与机器学习有密切的联系[7]。将 ACO 算法引入机器学习、自动控制和工程设计，将极大地促进算法的研究和发展。

4.4.2 *算法设计与执行策略* ACO 算法有很大的可扩展性和适应性，它的性能有赖于执行策略的选择。一是局部启发信息 η_{ij}的构造策略；二是信息素与局部启发信息的结合策略；三是状态转移策略，要保证足够的搜索空间以跳出局部最优；四是信息素调整策略，它直接影响搜索的集中程度，要避免陷入停滞状态。

4.4.3 *算法的基础理论研究* 蚂蚁的行为特性为算法的设计提供了好的启示，数值实验也说明了这一点。但算法性能的优劣需要理论上的保证。文[8]通过引入问题实例的构造图，将可行解编码表示为构造图上的路，并证明了在一定条件下，算法以一个可以任意接近 1 的概率收敛到全局最优解。这是一个较弱的结论。我们能否得到类似其它随机全局优化搜索技术的收敛结论，就象模拟退火算法所实现的那样？收敛速度是算法牵涉到的另一个重要问题，参数的选择影响算法的性能。指导参数设置的原则是什么？试验方法显然不如人意。能否得到一个最优解判别准则？好的最优解判别准则可以避免在解空间继续进行盲目搜索。另外，需要对算法模型本身进行更加深入的探讨，拓展它的适用范围。

结束语 本文对 ACO 算法做了详细介绍，希望读者能认识 ACO 算法及其协作方式的特点。目前，对蚁群优化算法的理论基础研究仅仅是一个开端，有必要从更高的高度与更广阔的视野审视 ACO 算法，并为未来探讨出一条新路。

参 考 文 献

1 Dorigo M，Maniezzo V，Colorni A. Ant system optimization by a colony of cooperating agents . IEEE Transactions on system，man，cybernetics，1996，26(1)

2 Dorigo M，Gambardella L M. Ant algorithm for discrete optimization. Artificial life，1999，5(3)

3 New ideas in optimization . chapter 2，3，4，McGraw-Hill，1999

4 Dorigo M，Ganbardella L M. Ant colony system：A cooperating learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation，1997，1(1)

5 Colorni A，Dorigo M，et al. Heuristics from nature for hard combinatorial optimization . International transactions in optimization research，1999，3(1)

6 Harik G，Lobo F，Goldbrg D. The compact genetic algorithm. IEEE Transactions on Evolutionary Computation，1999，3.1(4)

7 Dorigo M，Gambardella L M. A ...dy of some properties of ANT-Q. In：H. M. Voigt etc，Eds. Proc. 4th Int. Conf. On parallel solving from nature，Berlin：Springer-Verlag，656～665

8 Gutjahr W J. A graph-based ant system and its convergence. Future Generation Computer System，2000，16：873～888