基于变异和动态信息素更新的蚁群优化算法*

朱庆保[1+], 杨志军[2]

[1](南京师范大学 计算机科学系,江苏 南京 210097)

[2](爱丁堡大学 电子工程系, EH9 3JL,英国)

An Ant Colony Optimization Algorithm Based on Mutation and Dynamic Pheromone Updating

ZHU Qing-Bao[1+], YANG Zhi-Jun[2]

[1](Department of Computer Science, Nanjing Normal University, Nanjing 210097, China)

[2](Department of Electronics, University of Edinburgh, EH9 3JL, UK)

+ Corresponding author: Phn: E-mail: qbzhu@email.njnu.edu.cn, http://mcs.njnu.edu.cn

Received 2002-12-19; Accepted 2003-06-20

Zhu QB, Yang ZJ. An ant colony optimization algorithm based on mutation and dynamic pheromone updating. *Journal of Software*, 2004,15(2):185~192.

http://www.jos.org.cn/1000-9825/15/185.htm

Abstract: Despite the numerous applications of ACO (ant colony optimization) algorithm in optimization computation, it remains a computational bottleneck that the ACO algorithm costs too much time in order to find an optimal solution for large-scaled optimization problems. Therefore, a quickly convergent version of the ACO algorithm is presented. A novel strategy based on the dynamic pheromone updating is adopted to ensure that every ant contributes to the search during each search step. Meanwhile, a unique mutation scheme is employed to optimize the search results of each step. The computer experiments demonstrate that the proposed algorithm makes the speed of convergence hundreds of times faster than the latest improved ACO algorithm.

Key words: ant colony optimization; nearest neighbour; dynamic pheromone updating; mutation algorithm

摘 要: 尽管蚁群优化算法在优化计算中已得到了很多应用,但在进行大规模优化时,其收敛时间过长仍是应用该算法的一个瓶颈.为此,提出了一种高速收敛算法.该算法采用一种新颖的动态信息素更新策略,以保证在每次搜索中,每只蚂蚁都对搜索做出贡献;同时,还采取了一种独特的变异策略,以对每次搜索的结果进行优化.计算机实验结果表明,该算法与最新的改进蚁群优化算法相比,其收敛速度提高了数十倍乃至数百倍以上.

关键词: 蚁群优化;最近邻居;动态信息素更新;变异算法

中图法分类号: TP301 **文献标识码**: A

* Supported by the Natural Science Foundation of the Education Section of Jiangsu Province of China under Grant No. 01KJB520007 (江苏省教育厅自然科学基金)

作者简介: **朱庆保**(1955－),男,山东沂源人,教授,主要研究领域为人工智能,智能控制;**杨志军**(1962－),男,副研究员,主要研究领域为人工神经网络,模式识别.

蚁群优化(ant colony optimization,简称 ACO)算法是模拟自然界中真实蚁群的觅食行为而形成的一种模拟进化算法,是 20 世纪 90 年代意大利的 M. Dorigo 等学者提出的[1~3].受到其取得了较好的实验结果的影响,蚁群优化算法激起了其他学者的研究热情,并取得了很多研究和应用成果[4~8].近 10 年来的研究结果已经表明:蚁群算法用于组合优化具有很强的发现较好解的能力,具有分布式计算、易于与其他方法相结合、鲁棒性强等优点,在动态环境下也表现出高度的灵活性和健壮性.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[9~13].例如,文献[9]提出了一种 MMAS(max-min ant system)算法,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,以加快收敛速度.文献[10]则提出了一种新的改进的信息素更新策略:其一,局部信息素修改时,挥发系数动态改变;其二,全局信息素更新时,则将蚂蚁所走路的较短的那些路径上的信息加强,而较差的那些路径上的信息减弱.文献[11]提出了一种基于蚂蚁进化规则的算法.文献[12]主要提出了一种相遇算法,其基本思想是在求解 TSP 问题中,用两只蚂蚁共同完成对一条路径的搜索,以使搜索速度提高.文献[13]提出了一种变异策略,以加快局部搜索.还有不少其他文献作了改进,例如,引入交叉算子以提高搜索多样性、引入分支因子 r 作为衡量群体多样性的指标,当 r 低于某一值时,对各路径上的信息作动态调整,以期望克服停滞现象等等.这些研究对算法有一定程度的改进,但对提高收敛速度的效果不是特别明显,速度慢仍然是制约蚁群算法在大规模优化问题中的应用的瓶颈.为此,我们基于 TSP 问题的优化应用,研究了一种基于最近邻居选择、动态信息素更新和变异策略的高速收敛算法,简称 NDMACO 算法(ACO algorithm based on the nearest neighbor node choosing, dynamic pheromone updating and mutation).该算法以最近的邻居节点选择和动态信息更新策略来加速全局收敛,以一种独特的变异策略来加快局部寻优,使收敛速度大幅度地提高.实验结果表明,本文提出的算法与其他最新的改进算法相比,其收敛速度提高数百倍以上,结果也大幅度优于其他改进蚁群优化算法.

1 蚁群优化算法

经研究发现:蚂蚁在觅食过程中能够在所经过的路径上留下一种称为信息素的物质,而且蚂蚁在觅食过程中能够感知这种物质的存在及其强度,并以此指导自己的运动方向,它们倾向于朝着该物质强度高的方向移动.因此,由大量蚂蚁组成的集体觅食行为就表现出一种信息正反馈现象:某一路径越短,该路径上走过的蚂蚁就越多,所留下的信息素强度也就越大,后来者选择该路径的概率因此就越大.蚂蚁个体之间通过这种信息交流来选择最短路径并达到搜索食物的目的.蚁群优化算法就是模拟蚁群这一觅食行为的优化算法.

M. Dorigo 等人已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 AS 算法和 ACS 算法定义为 ACO(ant colony optimization)[7]算法.

1.1 基本蚁群系统模型(AS算法)

为了便于理解,仍以求解平面上 n 个城市的 TSP 问题为例,对于其他问题,对此模型稍作修改就可以应用.为了模拟实际蚂蚁的行为,首先引进如下记号:设 m 是蚁群中蚂蚁的数量,$d_{ij}(i,j=1,2,...,n)$表示城市 i 和城市 j 之间的距离,$b_i(t)$表示 t 时刻位于城市 i 的蚂蚁的个数,则有

$$m=\sum_{i=1}^{n}b_i(t)\ .$$

$\tau_{ij}(t)$表示 t 时刻在城市 i,j 连线上残留的信息量.初始时刻,各条路径上信息量相等,设$\tau_{ij}(0)=C(C$ 为常数).蚂蚁 $k(k=1,2,...,m)$在运动过程中,根据各条路径上的信息量决定转移方向.$p_{ij}^k(t)$表示在 t 时刻蚂蚁 k 由城市 i 转移到城市 j 的概率:

$$p_{ij}^k=\begin{cases}\dfrac{[\tau_{ij}(t)]^\alpha[\eta_{ij}]^\beta}{\sum_{k\notin tabu_k}[\tau_{ik}(t)]^\alpha[\eta_{ik}]^\beta}, & j\notin tabu_k\\ 0, & j\in tabu_k\end{cases}\tag{1}$$

其中:η_{ij} 为先验知识或称为能见度,在 TSP 问题中为城市 i 转移到城市 j 的启发信息,一般取$\eta_{ij}=1/d_{ij}$;α为在路径 ij 上残留信息的重要程度;β为启发信息的重要程度;与实际蚁群不同,人工蚁群系统具有记忆功能,$tabu_k(k=1,$

2,...,m)用以记录蚂蚁 k 当前所走过的城市,称为禁忌表(下一步不允许选择的城市).集合 $tabu_k$ 随着进化过程作动态调整.

经过 n 个时刻,所有蚂蚁都完成了一次周游.它们本次周游的禁忌表将满,此时应清空,将当前蚂蚁所在城市置入 $tabu_k$,准备下一次周游.这时,计算每一只蚂蚁所走过的路径 L_k,并保存最短路径 $L_{k_{\min}}$($L_{k_{\min}}=\min L_k$,k=1,...,m).

随着时间的推移,以前留下的信息逐渐消逝,用参数 1−ρ表示信息消逝程度,蚂蚁完成一次循环以后,各路径上的信息量要根据式(2)作调整:

$$\tau_{ij}(t+1)=(1-\rho)\tau_{ij}(t)+\rho\Delta\tau_{ij} \tag{2}$$

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

$$\Delta\tau_{ij}^{k}=\begin{cases}\dfrac{Q}{L_k}, & \text{当第}k\text{只蚂蚁在时刻}t\text{和}t+1\text{之间经过}ij\text{时}\\ 0, & \text{其他}\end{cases},$$

其中,$\Delta\tau_{ij}^{k}$ 表示第 k 只蚂蚁在本次循环中留在路径 ij 上的信息量,$\Delta\tau_{ij}$ 表示本次循环中路径 ij 上的信息量的增量,Q 为常数,L_k 表示第 k 只蚂蚁在本次循环中所走过的路径的长度.

一般设置周游次数计数器 NC,当达到设定值时结束.最短路径为 $L_{\min}=\min L_{k_{\min}}(l=1,2,...,NC)$.

1.2 ACS算法

ACS 算法也是一种 ACO 算法,算法描述如下:

Step 1. 初始化.将每个边上的信息素初始化为一个很小的常数值;将 m 只蚂蚁随机地分配到 n 个城市(或 n 个节点,以下同),同时,出发点城市设置到禁忌表中.

Step 2. 下一个节点的选择.

每只蚂蚁按式(3)或式(1)选择下一个城市,并修改禁忌表.

$$j=\begin{cases}\arg\max\limits_{j\notin tabu_k}\{[\tau_{ij}(t)][\eta_{ij}(t)]^{\beta}, & \text{if } q\le q_0\\ S, & \text{otherwise}\end{cases} \tag{3}$$

其中:0≤q_0≤1,是初始设定的参数;q 是一个随机数,$q\in[0,1]$;S 是根据式(1)决定的随机变量(在 ACS 算法中,式(1)中的α取消).

很显然,该策略增强了搜索的多样性,以避免过早地陷于搜索停滞.

Step 3. 信息素局部更新.

每只蚂蚁选择一个城市(走完一个边)以后,按式(4)更新该边上的信息素.

$$\tau_{ij}(t+1)=(1-\rho)\tau_{ij}(t)+\rho\Delta\tau_{ij}^{k} \tag{4}$$

$$\Delta\tau_{ij}^{k}=\begin{cases}\dfrac{Q}{l_{jb}}, & \text{当蚂蚁}k\text{走过城市}ij\text{时}\\ 0, & \text{否则}\end{cases},$$

其中,l_{jb} 是蚂蚁 k 从开始城市到当前城市已走过的路径长度.其余参数与式(2)相同.

Step 4. 计算最佳路径.

当 m 只蚂蚁走完所有城市以后,按式(5)计算最佳路径长度并保留;

$$l_{\min}=\min l_k,\ k=\{1,2,...,m\} \tag{5}$$

其中,l_{km} 是第 k 只蚂蚁所走的路径长度.

Step 5. 信息素全局更新

当所有蚂蚁走完全部城市以后,仅对最佳路径上的信息素按式(6)进行更新.

$$\tau_{ij}^{\text{new}}=(1-\alpha)\tau_{ij}^{\text{old}}+\alpha\Delta\tau_{ij} \tag{6}$$

$$\Delta\tau_{ij}^{k}=\begin{cases}\dfrac{1}{l_k}, & \text{if } ij\in global\text{-}best\text{-}tour\\ 0, & \text{otherwise}\end{cases},$$

其中,α为全局信息素挥发系数,l_k为最佳路径的长度,$ij\in global\text{-}best\text{-}tour$表示蚂蚁$k$所走的城市$ij$属于最佳路径.

Step 6. 设置的搜索次数如果未完,则清空禁忌表,重复上述过程.

2 NDMACO 算法

2.1 NDMACO算法的基本原理

2.1.1 最近节点选择策略

考察 TSP 问题,商人要遍历 n 个城市(每个城市只走一次),且使总路径最短.在这一过程中有两个值得注意的问题:(1) 根据 ACS 算法,商人从当前城市选择下一个城市的概率大小主要依据信息素和两城市间的距离长短,因此,对于一个有很多城市的地区,当商人从四周不同边部的城市出发时,他所选择的路径不相同的概率最大;(2) 在一个较复杂的、有若干城市的地图上,在一条遍历所有城市的最短路径中,城市 i 连接到城市 j, j 不可能是离城市 i 最远的那些城市.

根据上述第(1)点,可以将 m 只蚂蚁较均匀地分配在 $m(m\le n)$个边部城市,由 m 只蚂蚁完成从不同的边部城市为出发点的搜索.由于不同出发点构成的路径不同,各蚂蚁重复搜索的概率较小,增加了搜索的多样性,可以加快搜索速度.

在 ACS 算法中,当从城市 i 选择下一个城市 j 时,需要将 n–1 个城市与禁忌表比较,再计算 n–1 个城市的转移概率,需要较长的计算时间.根据上述第(2)点,将下一个城市的选择范围局限于离当前城市 i 较近的部分城市,仅对这部分城市的转移概率进行计算即可.当城市的选择范围限于 n/20 个城市时,这一计算过程的速度可以提高近 20 倍.

2.1.2 信息素动态更新策略

根据实验,当蚂蚁从一个具有很多城市的复杂地图的边部城市出发时,它在出发点附近的局部区域走的路径往往是较优的,走到地图中心区,路径错综复杂,与其他蚂蚁走的路径重叠增多,互相影响加大,走出较优路径的概率降低.根据这一实验结果,为了避免中心区信息素堆积过多以及体现众多蚂蚁的共同影响,每只从边部城市出发的蚂蚁留下的信息素应随着向中心区的延伸而逐渐减弱,当走到最远处时,已渗透到另一边部其他蚂蚁的领地,为了不过大地干扰其他蚂蚁开始走出的较优结果,其信息素应减到最小.由于蚂蚁间的相互联系以及对搜索的主要贡献是留下信息素,因此,信息素的更新策略是决定收敛速度的关键之一.

2.1.3 最优个体变异策略

实验表明,应用上述两种策略改进后的蚁群算法的全局搜索能力相当强,可迅速收敛到一个较优解.但从较优解到最优解所花费的收敛时间所占比重很大.利用变异算法局部寻优能力强的特点,在收敛到一定代数时,对最优个体进行变异,可大大加快局部寻优速度.

综合以上几点,研究了一种基于求解 TSP 问题的最近邻居节点选择、动态信息更新和变异策略的高速收敛蚁群优化算法,简称 NDMACO 算法.

2.2 NDMACO算法步骤

根据第 2.1 节的原理,NDMACO 算法步骤如下:

Step 1. 初始化.将 n 个城市分别按 x,y 坐标排序,均匀地在地图周围找出 N 个边部城市,$N=n/h$,$h\in[1,20]$,n 越大,h 取值可越大,若 n 较小时,h 取较小的值.将 m 只蚂蚁分别固定地放置在 N 个边部城市上(一般取 $m=N$,每个城市放置一只蚂蚁),并设置到禁忌表中(对于不能按 x,y 坐标排序的城市,则每个城市放置一只蚂蚁).

分别以 n 个城市为起点按距离排序,生成 n 个排序表.同时,设置代数计数器 NC=MAX,其余初始化与 ACS 算法相同.

Step 2. 对于每一只蚂蚁,以当前城市 i 为中心,按最近邻居原则选取离蚂蚁所在当前城市 i 最近的 n/w 个城

市作为下一个城市 j 的选择范围(根据城市 i 的距离排序表选择前 n/w 个城市),$n/w\in[5,20]$.

从 n/w 个城市中找出 z 个未走过的城市 $\{j_1,j_2,...,j_p,...,j_z\}$,即 $j_p\notin tabu_k$.

Step 3. 在 z 个城市中,按式(7)选择节点 j:

$$\begin{cases} j=\arg\max\limits_{j\notin tabu_k}\{[\tau_{ij}(t)][\eta_{ij}(t)]^{\beta}, & \text{if } q\le q_0 \\ p_{ij}^{k}=\dfrac{[\tau_{ij}(t)][\eta_{ij}]^{\beta}}{\sum\limits_{k\notin tabu_k}[\tau_{ik}(t)][\eta_{ik}]^{\beta}}, j\notin tabu_k, & \text{else} \end{cases} \tag{7}$$

其中,q 为随机数($0<q\le 1$),当 $q>q_0$ 时,计算 z 个城市的转移概率 p_{ij}^{k},并根据赌轮盘规则选择城市 j.

将 j 加入禁忌表 $tabu_k$.

Step 4. 局部信息素动态更新.

当每只蚂蚁走完一个边以后,即按式(8)进行局部信息更新.

$$\tau_{ij}^{\text{new}}=(1-\rho)\tau_{ij}^{\text{old}}+\rho\Delta\tau_{ij} \tag{8}$$

$$\Delta\tau_{ij}=\frac{Q_1}{L_z},$$

$$L_z=\sum_{l_1}^{x}d_{l_1}^{1}+\sum_{l_2}^{x}d_{l_2}^{2}+...+\sum_{l_m}^{x}d_{l_m}^{m}=\sum_{k=1}^{m}\sum_{l=1}^{x}d_l^k,$$

$$d_l\subseteq d_{ij},\ l=1,2,...,x,\ \forall i,j=1,2,3,...,n.$$

其中,Q_1 为一个较大的常数,x 是蚂蚁 k 在本次周游中已走过的城市边数;d_l 是蚂蚁 k 已走过的边的边长,m 是蚂蚁总数,L_z 就是所有蚂蚁在本次周游中已走过的边的累加总长.

该策略保证了各蚂蚁所留信息素的均衡性,保证了对搜索的均衡贡献和相互协作,体现出了群体的力量,这是本文的算法能够大幅度地提高收敛速度的关键之一.

Step 5. 当 m 只蚂蚁选择完节点 j 以后,令 $i_{\text{new}}=j$; $j_{\text{new}}=j_{\text{old}}+1$;返回 Step 2,开始选择下一个节点,直到所有蚂蚁完成一次周游为止.

Step 6. 当 m 只蚂蚁完成一次周游以后,计算 l_k 和 $l_{k_{\min}}$,并保留 $l_{k_{\min}}$ 和本次最优路径表 $tabu_l.k=1,2,...,m$; $l_{k_{\min}}=\min l_k$;l_k 是蚂蚁 k 完成一次周游的路径长度.

Step 7. 变异运算.令 $NC=NC-1$,若 MAX-$NC<M$,转 Step 8,否则进行下面的变异运算,M 为设定的开始变异界限值.变异策略如下:

以蚂蚁 k 走过的一周路径表为染色体个体,设由 Step 6 得到的最优个体为 $a_1,a_2,a_3,...,a_i,...,a_n,a_1$,其中 a_i 为城市编号.适应度函数为

$$l_z=\sum_{l=1}^{n}d_l,\ \text{其中 } d_l \text{ 为边长.}$$

变异算法的伪代码如下:

```
for (i=2;i<n;i++)
  { a_i 与 a_{i+1} 交换,评价适应度 l_z
    if (l_z<l_{k_min}) {用 l_z 替换 l_{k_min},更新本次最优路径表,结束变异}
    else 反变异
   }
if (i>=n)
  { for (i=2;i<n-1;i++)
    a_i 与 a_{i+2} 交换,评价适应度 l_z
    if (l_z<l_{k_min}) {用 l_z 替换 l_{k_min},更新本次最优路径表,结束变异}
    else 反变异
   }
```

如果仍然没有得到更优路径,将原个体依次循环一个城市号,原个体变为 $a_n,a_1,a_2,...,a_i,...,a_{n-1},a_n$,重复上述过程一次.

Step 8. 按式(9)进行全局信息素更新.

$$\tau_{ij}^{\text{new}} = (1-\alpha)\tau_{ij}^{\text{old}} + \alpha\,\Delta\tau_{ij}^{k} \tag{9}$$

$$\Delta\tau_{ij}^{k} = \begin{cases} \sum_{l=1}^{n} \dfrac{Q}{d_l}, & \text{if } l \in global\text{-}best\text{-}tour \\ 0, & \text{otherwise} \end{cases},$$

$l \in global\text{-}best\text{-}tour$ 表示蚂蚁 k 所走过的城市边 l 属于最佳路径.

Step 9. 将本次周游得到的 $l_{k_{\min}}$ 与已得到的最优长度 l_d 比较,若有 $l_{k_{\min}}<l_d$,则用 $l_{k_{\min}}$ 替换 l_d,同时替换最优路径表.

Step 10. 设置的计数值 NC–1 不等于 0,清空并初始化禁忌表,重复上述过程,直到 NC–1=0 为止.

3 实验比较

为了验证、比较算法的效果,我们从 TSPLIB 下载了多个著名的 TSP 问题的范例进行实验,都取得了非常好的结果.为了与最新文献的算法进行比较,我们选择了几种与文献[10,11]相同的 TSP 问题的实验结果列入本文.由于对各参数的选择还没有指导理论,因此,本实验的参数均通过实验确定.文献[10]提出了一种动态权规则算法,其主要思想是:局部信息素修改时,挥发系数动态改变;另外,当全局信息素更新时,将蚂蚁所走路的较短的那些路径上的信息加强,而将较差的那些路径上的信息减弱.文献[11]提出了一种基于蚂蚁进化规则的算法.

3.1 收敛特性的实验比较

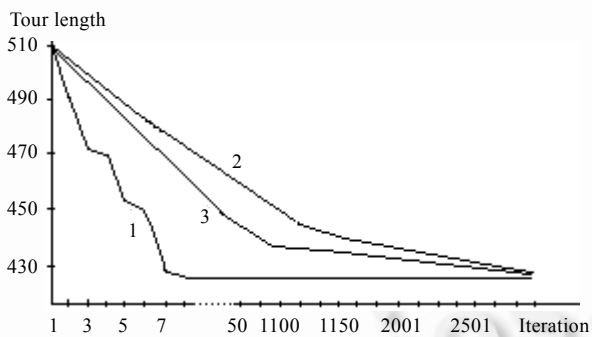

Fig.1 Comparison of convergent features of the optimized eil51

图 1 不同方法优化 eil51 收敛特性比较

图 1 是用不同方法优化 eil51 TSP 问题的收敛特性比较,其中,横坐标为蚂蚁周游次数,纵坐标为蚂蚁周游取得的最佳路径长度.曲线 1 为本算法收敛特性,曲线 2 为 ACS 算法的收敛特性,曲线 3 为文献[10]的改进算法结果.曲线 2 和曲线 3 摘自文献[10].实验结果表明,ACS 算法和文献[10]中的算法均需周游 2 500 次以上才收敛到较优值,本文提出的算法仅需 7 次即收敛到最优值.本实验的实验参数为 ρ=0.2,α=0.2,β=3,Q=20000,Q_1=50000,q_0=0.75,w=10,m=35.

图 2 是用不同算法优化 St70 TSP 问题的收敛特性比较.在图 2 中,曲线 1 为本文的算法,曲线 2 为 ACS 算法,曲线 3 为文献[11]的改进算法(曲线 2 和曲线 3 摘自文献[11]).实验结果表明,ACS 算法和文献[11]的改进算法需要 6 000 多次才能得到较优值,而本文的算法仅需要 20 多次即可得到最优值.本实验的实验参数为 ρ=0.2,α=0.2,β=3,Q=10000,Q_1=8000,q_0=0.8,w=10,m=50.

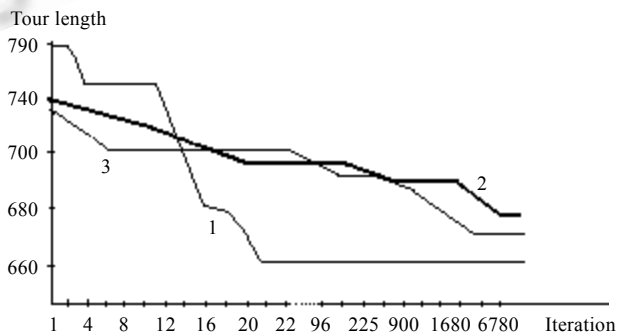

Fig.2 Comparison of convergent features of the optimized St70

图 2 优化 St70 收敛特性比较

3.2 优化其他TSP问题的结果比较

表 1 给出的是优化部分其他 TSP 问题的结果比较.

Table 1 Comparison between the different improved algorithms

表 1 不同改进算法比较

	Optimal results of TSPLIB	Optimal results of the proposed algorithm	Optimal results of the improved ACS	Convergence number of the proposed algorithm	Convergence number of the improved ACS
Berlin52	7 542	7 542	7 558	11	2 800
Pr107	44 303	44 283	44 385	330	6 700
Dl198	15 780	15 796	16 197	800	10 000

在表 1 中,改进的 ACS 优化结果是指文献[9~11]中的算法取最好结果.收敛代数是指当达到表 1 的结果时,蚂蚁的周游代数.从实验对比可见,本算法蚂蚁周游代数减少数十倍至数百倍,多数结果达到了最优解.

本文的算法不仅大幅度减少了周游代数,而且,蚂蚁完成一次周游的时间也大幅度降低.由于本文采用了最近邻居选择原则,每次周游仅需在 n/w 个城市中选择和计算,相当于把城市规模缩小了近 w 倍,城市规模越大,w 也可取得越大,压缩效果也就越显著.表 2 给出了部分实验结果,后 3 例的结果为:周游代数相同、蚂蚁数相同时,不同算法的时间比.在表 2 中,T,t_{ACO},t_{12},t_{13} 分别为本文、ACS、文献[10,11]的算法时间.

Table 2 The comparison of convergent time between the different improved algorithms

表 2 各种改进算法收敛时间比较

	W	T/t_{Acs}	T/t_{12}	T/t_{13}
Eil51	10	6	6	6
St70	10	5.5	5.5	5.5
Pr107	10	5	5	5
Dl198	15	8	8	8

4 结 语

本文的算法将 m 只蚂蚁均匀放置于 m 个边部城市,采用最近邻居节点选择原则,在此基础上进行动态局部信息素更新,并用变异算法加速局部寻优,使收敛速度提高了数百倍以上,且能得到最优值.

除了本文所列入的实验以外,我们还对近几年有代表性的其他文献中描述的算法进行了实验对比,结果表明,本文的算法其收敛速度远远高于其他算法,特别是在大规模 TSP 类优化应用时,本文的算法有着非常明显的优势,使蚁群优化算法用于大规模优化成为可能.本文虽然以 TSP 问题为例,但其思想方法可用于其他优化问题.

然而,蚁群算法是一种新的模拟进化算法,还没有像 GA 等算法那样形成系统分析的方法和坚实的数学基础,各种实验参数的确定也没有理论上的指导,目前国际上诸多研究成果还都基于实验分析,还有许多理论问题有待进一步研究.但可以推断的是,随着研究的深入,蚁群优化算法也将与其他模拟进化算法一样,能够获得越来越多的应用.

References:

[1] Colorni A, Dorigo M, Maniezzo V. Distributed optimization by ant colonies. In: Varela F, Bourgine P, eds. Proc. of the ECAL'91 European Conf. of Artificial Life. Paris: Elsevier, 1991. 134~144.

[2] Dortgo M, Maniezzo V, Colorni A. Ant system: Optimization by a colony cooperating Agents. IEEE Trans. on Systems, Man, and Cybernetics—Part B: Cybernetics, 1996,26(1):29~41.

[3] Dortgo M, Gambardella LM. Ant colony system: A cooperative learning approach to the traveling salesman problem. IEEE Trans. on Evolutionary Computation, 1997,1(l):53~66.

[4] Hoshyar R, Jamali SH, Locus C. Ant colony algorithm for finding good interleaving pattern in turbo codes. IEE Proceedings—Communications, 2000,147(5):257~262.

[5] Merkle D, Mdiddendorf M, Schmeck H. Ant colony optimization for resource-constrained project scheduling. IEEE Trans. on Evolutionary Computation, 2002,6(4):333~339.

[6] Parpinelli RS, Lopes HS, Freitas AA. Data mining with an ant colony optimization algorithm. IEEE Trans. on Evolutionary Computation, 2002,6(4):321~328.

[7] Dorigo M, Caro GD. Ant colony optimization: A new meta-heuristic. In: Proc. of the 1999 Congress on Evolutionary Computation, Vol 2. Washington: IEEE Press, 1999. 1470~1477.

[8] Dorigo M. Special section on ant colony optimization. IEEE Trans. on Evolutionary Computation, 2002,6(4):317~319.

[9] Stutzle T, Hoos HH. MAX-MIN ant xystem and local search for the traveling salesman problem. In: IEEE Int'l Conf. on Evolutionary Computation. Indianapolis: IEEE Press, 1997. 309~314.

[10] Lee SG, Jung TU, Chung TC. An effective dynamic weighted rule for ant colony system optimization. In: Proc. of the 2001 Congress on Evolutionary Computation, Vol 2. IEEE Press, 2001.

[11] Tsai CF, Tsai CW. A new approach for solving large traveling salesman problem using evolution ant rules. In: Neural Networks, IJCNN 2002, Proc. of the 2002 Int'l Joint Conf. on, Vol 2. Honolulu: IEEE Press, 2002. 1540~1545.

[12] Wu B, Shi ZZ, An ant colony algorithm based partition algorithm for TSP. Chinese Journal of Computers, 2001,24(12):1328~1333 (in Chinese with English abstract).

[13] Wu QH, Zhang JH, Xu XH. An ant colony algorithm with mutation features. Journal of Computer Research and Development, 1999,36(10):1240~1245 (in Chinese with English abstract).

附中文参考文献:

[12] 吴斌,史忠植.一种基于蚁群算法的 TSP 问题分段求解算法.计算机学报,2001,24(12):1328~1333.

[13] 吴庆洪,张纪会,徐心和.具有变异特征的蚁群算法.计算机研究与发展,1999,36(10):1240~1245.

敬 告 作 者

《软件学报》创刊以来,蒙国内外学术界厚爱,收到许多高质量的稿件,其中不少在发表后读者反映良好,认为本刊保持了较高的学术水平.但也有一些稿件因不符合本刊的要求而未能通过审稿.为了帮助广大作者尽快地把他们的优秀研究成果发表在我刊上,特此列举一些审稿过程中经常遇到的问题,请作者投稿时尽量予以避免,以利大作的发表.

1. 读书偶有所得,即匆忙成文,未曾注意该领域或该研究课题国内外近年来的发展情况,不引用和不比较最近文献中的同类结果,有的甚至完全不列参考文献.

2. 做了一个软件系统,详尽描述该系统的各个方面,如像工作报告,但采用的基本上是成熟技术,未与国内外同类系统比较,没有指出该系统在技术上哪几点比别人先进,为什么先进.一般来说,技术上没有创新的软件系统是没有发表价值的.

3. 提出一个新的算法,认为该算法优越,但既未从数学上证明比现有的其他算法好(例如降低复杂性),也没有用实验数据来进行对比,难以令人信服.

4. 提出一个大型软件系统的总体设想,但很粗糙,而且还没有(哪怕是部分的)实现,很难证明该设想是现实的、可行的、先进的.

5. 介绍一个现有的软件开发方法,或一个现有软件产品的结构(非作者本人开发,往往是引进的,或公司产品),甚至某一软件的使用方法.本刊不登载高级科普文章,不支持在论文中引进广告色彩.

6. 提出对软件开发或软件产业的某种观点,泛泛而论,技术含量少.本刊目前暂不开办软件论坛,只发表学术文章,但也欢迎材料丰富,反映现代软件理论或技术发展,并含有作者精辟见解的某一领域的综述文章.

7. 介绍作者做的把软件技术应用于某个领域的工作,但其中软件技术含量太少,甚至微不足道,大部分内容是其他专业领域的技术细节,这类文章宜改投其他专业刊物.

8. 其主要内容已经在其他正式学术刊物上或在正式出版物中发表过的文章,一稿多投的文章,经退稿后未作本质修改换名重投的文章.

本刊热情欢迎国内外科技界对《软件学报》踊跃投稿.为了和大家一起办好本刊,特提出以上各点敬告作者.并且欢迎广大作者和读者对本刊的各个方面,尤其是对论文的质量多多提出批评建议.