文章编号：1005-8451（2005）09-0004-04

基于MATLAB的混合型蚁群算法求解旅行商问题

尹晓峰，刘春煌
（铁道科学研究院 电子计算技术研究所，北京 100081）

摘 要：蚁群算法是受自然界中蚁群搜索食物行为启发而提出的一种智能优化算法。通过介绍蚁群觅食过程中基于信息素的最短路径的搜索策略，给出基于MATLAB的蚁群算法在旅行商问题中的应用，针对蚁群算法存在的过早收敛问题，加入2-opt方法对问题求解进行局部优化。计算机仿真结果表明，这种混合型蚁群算法对求解旅行商问题有较好的改进效果。

关键词：旅行商问题；组合优化；蚁群算法；改进
中图分类号：U29-39 文献标识码：A

Hybrid approach based on Ant Colony System for solving traveling salesman problem

YIN Xiao-feng, LIU Chun-huang
(Institute of Computing Technology, China Academy of Railways Sciences, Beijing 100081, China)

Abstract: Ant algorithms have been inspired by the behavior of real ant colonies, in particular, by their foraging behavior.It was introduced the main idea of this distributed algorithm which was the indirect communication of ants based on pheromone trails, proposed codes written in MATLAB. Computer simulation showd that apply a hybrid approach of ant algorithm with 2-opt could efficiently find better minimum beyond premature convergence for traveling salesman problem.

Key words: traveling salesman problem; combinatorial optimization; ant colony algorithm; improvement

蚁群算法（ant colony algorithm）首先由意大利科学家M. Dorigo等人提出，称为蚁群系统（ant system, AS）。在求解二次分配、图着色问题、车辆调度、集成电路设计以及通信网络负载问题的处理中都取得了较好的结果。

旅行商问题（TSP, Traveling Salesman Problem）

收稿日期：2005-01-06
作者简介：尹晓峰，在读博士研究生；刘春煌，研究员。

达或出发的，按照客车、无调中转、有调中转、到达解体和自编始发的顺序进行接发车作业；对于解体作业，作为出发列车车流来源的车组所在车列按车组被编入出发列车的出发时刻排序，其余按到达时刻排序被编入列车出发时间最早的车列解体优先级最高。编组作业的优先级按出发列车的出发时刻依次递减；若同时出发，则最先集结完毕的优先级较高。

3 结束语

本文提出了一种利用计算机仿真技术自动编制车站调度作业计划的方法。该方法具有以下特点：

（1）车站作业过程的模拟可以计划时段内任一时刻为起始点，反复进行模拟，且模拟时间可伸缩，滚动编制作业计划。计划的编制不再拘泥于实际作业中分班计划、阶段计划和调车作业计划3步编制的形式，编制更为灵活。

（2）在作业计划的编制过程中，当需要对计划进行调整时，结束当前的模拟进程，以调整时刻为模拟起点，重新进行模拟，制定作业计划，实现计划的调整，编调合一。这样做加大了计算机的工作量，但减少了计划调整时的人工工作量，而且调整是对后续计划的重新编制，从而加大了计划的可调范围。

参考文献：

[1] 杨 浩，何世伟. 铁路运输组织学[M]. 北京：中国铁道出版社，2001.

[2] 杨肇夏.计算机模拟及其应用[M]. 北京：中国铁道出版社，1999.

[3] 齐 欢，王小平.系统建模与仿真[M]. 北京：清华大学出版社，2004.

被认为是一个基本问题，是在1859年由威廉·汉密尔顿爵士首次提出的。所谓TSP问题是指：有N个城市，要求旅行商到达每个城市各一次，且仅一次，并回到起点，且要求旅行路线最短。这是一个典型的优化问题，对一个具有中等顶点规模的图来说，精确求解也是很复杂的，计算量随着城市个数的增加而呈指数级增长，即属于所谓的NP问题。TSP在工程领域有着广泛的应用，并常作为比较算法性能的标志。如网络通讯、货物运输、电气布线、管道铺设、加工调度、专家系统、柔性制造系统等方面，都是TSP广泛应用的领域。求解算法包括贪婪法（GM）、极小代数法（MA）、模拟退火法（SA）和遗传算法（GA）等。而应用蚁群算法求解旅行商问题是近年来研究的新方向，由于其并行性与分布性，特别适用于大规模启发式搜索，实验结果证明了其可行性和有效性。

1　蚁群系统基本原理

在蚂蚁群找到食物时，它们总能找到一条从食物到巢穴之间的最优路径。这是因为蚂蚁在寻找路径时会在路径上释放出一种特殊的信息素（pheromone）。当它们碰到一个还没有走过的路口时，就随机地挑选一条路径前行。与此同时释放出与路径长度有关的信息素。路径越长，释放的激素浓度越低。当后来的蚂蚁再次碰到这个路口的时候，选择激素浓度较高路径概率就会相对较大。这样形成了一个正反馈。最优路径上的激素浓度越来越大，而其它的路径上激素浓度却会随着时间的流逝而消减。最终整个蚁群会找出最优路径。在整个寻径过程中，虽然单个蚂蚁的选择能力有限，但是通过激素的作用，整个蚁群之间交换着路径信息，最终找出最优路径。

2　基于MATLAB的蚁群算法求解旅行商问题

TSP问题描述如下：

设有n个城市集C=(1,2,…,n)，任意两个城市i,j之间的距离为d_{ij}，求一条经过每个城市仅一次的路径$\pi=(\pi(1),\pi(2),\cdots\pi(n))$，使得

$$\sum_{i=1}^{n} d_{\pi(i)\pi(i+1)} + d_{\pi(n)\pi(1)} \text{ 最小}$$

求解TSP问题的蚂蚁算法中，每只蚂蚁是一个独立的用于构造路线的过程，若干蚂蚁过程之间通过自适应的信息素值来交换信息，合作求解，并不断优化。这里的信息素值分布式存储在图中，与各弧相关联。蚂蚁算法求解TSP问题的过程如下：

（1）首先初始化，设迭代次数为NC。初始化NC=0，各τ(i,j)初始化；$\tau_0 = nL_{nn}^{-1}$。相应的MATLAB程序如下：

[L_nn, P_nn] = NearestNeighborTSP(d)；%L_{nn}是最近邻域启发算法产生的路线长度；

```
L_best = inf;
T_best = 0;
tau0 = 1 / (n * L_nn); % n为城市数；
tau = ones(n,n) * tau0;
ant_tours = zeros(m, n+1);
ant_tours(:,1) = randint(m,1,[1,n]);
```

（2）将m个蚂蚁置于n个顶点上；

（3）构造解。每个蚂蚁按照状态变化规则逐步地构造一个解，即生成一条线路。蚂蚁任务是访问所有的城市后回到起点，生成一条回路。设蚂蚁k当前所在的顶点为i，那么，蚂蚁k由点i向点j移动要遵循（1）式的状态变化规则而不断迁移，按不同概率来选择下一点。

$$j=\begin{cases}\arg\max\limits_{k\in allowed_k}[(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}] & q\le q_0 \quad (Exploitation)\\ J & q>q_0 \quad (Exploration)\end{cases} \tag{1}$$

（1）式中，$allowed_k=\{0,1,\cdots n-1\}-tabu_k$表示蚂蚁$k$当前能选择的城市集合，$tabu_k$为禁忌表，它记录蚂蚁$k$已路过的城市，用来说明人工蚂蚁的记忆性；式中$\tau_{ij}$相当于真实蚂蚁沿途散播的信息素，是一个正实数，与图G中弧（i，j）关联，其值在运行时不断改变，用于表示蚂蚁从点i向点j移动的动力；η_{ij}用于评价蚂蚁从点i向点j移动的启发函数，其值通常用距离的倒数来求得，即$\eta_{ij}=d(c_i,c_j)^{-1}$。$\alpha,\beta$体现了信息素和启发信息对蚂蚁决策的影响。$\alpha$取值为1；参数$\beta$>0描述启发函数的重要性；参数q0（0≤q0≤1）决定利用和开发的相对重要性，利用（Exploitation）是指走最好的路，开发（Exploration）是指按浓度高概率高的原则选路V，这样可以保证选择路径的多样性；q是在[0，1]上任取的随机数，当q≤q0时，按（1）式选最好的路，否则按（2）

式的概率进行选路：

$$p_{ij}^{k}(t)=\begin{cases}\dfrac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{k\in allowed_k}[\tau_{ik}(t)]^{\alpha}[\eta_{ik}]^{\beta}} & \text{if } j\in \text{allowed}_k\\ 0 & \text{otherwise}\end{cases} \quad (2)$$

相应的MATLAB程序如下：

```
current_node = ant_tours(k,s-1)；  %k为蚂蚁数目，取值1,…,m，s为问题规模，取2,…,n
visited = ant_tours(k,:);
to_visit = setdiff([1:n],visited);
c_temp = length(to_visit);
p = zeros(1,c_temp);
for i = 1 : c_temp
p(i) = (tau(current_node,to_visit(i)))^alpha * (1/d(current_node,to_visit(i)))^beta; % 计算(τij)^α(ηij)^β
end
sum_p = sum(p);
q0 = rand;
select = to_visit(c_temp);
 if (q0 <=0.9 )
  [y i] = max(p(i));
   select = to_visit(i);
 else  p = p / sum_p;
   [y i] = max(p(i));
   select = to_visit(i);
 end
 city_to_visit = select;
 ant_tours(k,s) = city_to_visit;
```

（4）局部更新信息素值。应用局部信息素更新规则来改变信息素值。在构造解时，蚂蚁k对其走过的每条弧用：

$$\tau_{ij}^{new}=(1-\rho)\tau_{ij}^{old}+\rho\tau_0 \quad (3)$$

局部信息素更新规则来改变弧上关联的信息素值，其中0< ρ <1是信息素挥发参数。相应的MATLAB程序如下：

```
tau(current_node,city_to_visit) = (1 - rho) * tau(current_node,city_to_visit) + tau0;
```

（5）若所有的m个蚂蚁都构造完解，则转（6），否则转（3）；

（6）全局更新信息素值。应用全局信息素更新规则来改变信息素值。当所有m个蚂蚁生成了m个解，其中有一条最短路径是本代最优解，将属于这条路线上的所有弧相关联的信息素值按下式更新：

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

$$\Delta\tau_{ij}^{gb}(t)=\begin{cases}1/L^{gb}(t) & \text{if arc(i, j)}\in T^{gb}\\ 0 & \text{otherwise}\end{cases}$$

其中0< ρ <1是挥发参数；$L^{gb}(t)$是目前得到的全局最优解的路线长度。相应的MATLAB程序如下：

```
tau(Tour_min(i),Tour_min(i+1)) = (1 - rho) * tau(Tour_min(i),Tour_min(i+1)) + rho / L_gb; %
```

全局信息素更新的目的是在最短路线上注入额外的信息素，即只有属于最短路线的弧上的信息素才能得到加强，这是一个正反馈的过程，也是一个强化学习的过程。在图中各弧上，伴随着信息素的挥发，全局最短路线上各弧的信息素值得到增加；

（7）终止。若终止条件满足，则结束；否则NC=NC+1，转（2）进行下一代进化。终止条件可指定进化的代数，也可限定运行时间，或设定最短路长的下限。

3 用2-opt方法局部优化蚂蚁算法构造的解

不同的智能算法出现停滞现象的原因各不相同，但结果是相同的，即所求的解越来越相似，避免这种现象的方法也是一致的，那就是增加解的多样性。蚂蚁算法尽管能够分布式并行搜索，但在限定的时间或代数内找到最优解仍是困难的，可能找到的只是可行的近优解，这一点与遗传算法相似。用于启发式局部优化的方法很多，主要包括2-opt, 3-opt, 顶点重定位(relocate), 交换(exchange)和交叉（cross）等，其中实用有效的是2-opt和3-opt算法。

因此，我们在蚂蚁算法中混入局部优化算法，对每代构造的解进行改进，从而进一步缩短解路线的长度，以加快蚂蚁算法的收敛速度。将2-opt方法混入蚂蚁算法求解过程中，对每代最优解进行改进，这样，在上述求解过程（5）与（6）之间加入以下2-opt方法对本代最优解进行改进。

```
current_tour：= create_random_initial_tour()
repeat
modified_tour：= apply_2opt_move (current_tour)
if length(modified_tour) < length(current_tour)
```

```
    then current_tour ; = modified_tour
    until no further improvement or a specified number
of iterations
    相应的MATLAB程序如下：
    N = [1 2];
    for r = 1 : n-1   %n为问题规模
      for s= r+1 : n
        N = [N ; [r s]];
      end
    end
    All_line=N;
    while (length(All_line(:,1)) > 0)
      r = All_line(1,1);
      s = All_line(1,2);
      All_line = setdiff(All_line, [r s], 'rows'); %排除
已经处理过的边
      ctemp = T(r+1:s-1);
      n_ctemp = length(ctemp);
      temp = zeros(1,n_ctemp);
      for i = 0 : n_ctemp-1
        temp(i+1) = ctemp(n_ctemp-i);
      end
      current_T = [T(1:r-1) T(s) temp T(r) T(s+1:n)]; %
进行边交换
      current_T = [current_T current_T(1)];% 形成
回路
      current_L = 0;
      for i = 1 : n
        current_L = current_L + d(current_T(i),
current_T(i+1));
      end
      if (current_L < L)
        T = current_T;
        L = current_L;
        All_line = N;
      end
    end   % 此时的T为经过2-opt后的最短路径。
```

4　仿真实验

本文选用TSPLIB中列出的两个TSP问题：Bavaria 29 – city 和 swiss 42 – city 问题。这两个问题目前已知的最优长度为2020和1273。分别用基本蚁群算法和加入2-opt的混合算法对两问题进行求解，每个程序各运行20次，计算结果如表1所示。求解得到的最优路径分别为14 17 18 15 4 10 20 2 3 29 26 5 9 12 6 28 1 21 13 16 24 8 27 23 7 25 19 11 22 14和6 27 19 13 12 26 11 9 42 24 10 41 25 22 40 23 39 31 30 29 28 3 4 5 7 2 1 33 35 34 21 36 37 32 18 8 38 16 17 15 20 14 6。

表1　二种算法的计算结果比较

$a=1, \beta=9, \rho=0.1, 29-city, m=29, 42-city, m=42$

算法	29-city 问题		42-city 问题	
	平均值	最好结果	平均值	最好结果
基本蚁群算法	2225.5	2163	1343.5	1322
混合算法	2047	2020	1306	1274

从表1可以发现，混合ACO方法改进了蚂蚁算法的局部搜索能力，因而提高了算法的求解能力，可得到更短的路线长度，明显优于基本蚁群算法，验证了改进措施的有效性。

5　结束语

实验证明，我们研究并提出的混合算法是有效的，同时为继续深入研究更复杂的带约束的组合优化问题-车辆路线问题(Vehicle Routing Problem,VRP)创造了条件。

参考文献：

[1] 赵学峰.一种求解TSP的混合型蚁群算法 [J].西北师范大学学报，2003，39（4）：31—34.

[2] 李　虹，孙志毅.基于MATLAB的改进型基本蚁群算法 [J].太原重型机械学院学报，2003（9）：201—204.

[3] SeungGwan Lee, TaeUng Jung, Taechoong Chung. An effective dynamical weighted rule for Ant Colony system algorithm[C]. Proceedings of IEEE conference on Evolutionary Computation. 2001（2）：1393—1396.

[4] 储理才.基于MATLAB的遗传算法设计及TSP问题求解 [J].集美大学学报，2000（3）：14—19.