基于MATLAB的混合型蚁群算法求解车辆路径问题

尹晓峰　刘春煌　张惟皎
(铁道部科学研究院电子所,北京 100081)
E-mail:yinxiaofeng@rails.com.cn

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,给出了基于MATLAB的蚁群算法在车辆路径问题中的应用,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了局部优化,计算机仿真结果表明,这种混合型蚁群算法对求解车辆路径问题有较好的改进效果。

关键词　蚁群算法　组合优化　车辆路径问题

文章编号 1002-8331-(2005)35-0207-03　**文献标识码** A　**中图分类号** TP391

Hybrid Approach Based on Ant Colony System for Solving Vehicle Routing Problem

Yin Xiaofeng　Liu Chunhuang　Zhang Weijiao
(The Institute of Computing Technologies, China Academy of Railway Sciences, Beijing 100081)

Abstract: Ant Colony System algorithm has been inspired by the behavior of real ant colonies, in particular, by their foraging behavior. In the paper the authors introduce the main idea of this distributed algorithm which is the indirect communication of ants based on pheromone trails, codes written in MATLAB is proposed, computer simulation shows that applying a hybrid approach of ant algorithm with 2-opt can efficiently find better minimum beyond premature convergence for vehicle routing problem.

Keywords: Ant Colony System, combinatorial optimization, Vehicle Routing Problem

蚁群算法(Ant Colony System algorithm)[1]首先由意大利科学家M.Dorigo等人提出,由Ant-Q简化改进而来。在求解二次分配、图着色问题、旅行商问题、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。

车辆路径问题(Vehicle Routing Problem,VRP)是物流系统研究中的一项重要内容,许多实际的运输物流和配送问题都可以表示为VRP,选取合适的运输路线,可以加快对客户需求的响应速度,提高服务质量,增强客户对物流系统的满意度,降低服务商的运营成本。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)等等。而应用蚁群算法求解车辆路径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

VRP最早是由G.Dantzig提出的,它是一类最常见、研究最多的NP难问题,图$G=(V,E)$经常用来描述该问题。在图$G=(V,E)$中,$V=\{0,1,2,\cdots,n\}$,$E=\{(i,j),i\neq j,i,j\in V\}$,节点1表示仓库(depot),其它节点为客户。每个客户的需求为q_i,边(i,j)对应的距离或运输时间或成本为C_{ij}。所有车的运输能力为Q。车辆从仓库(depot)出发,完成运输任务后回到仓库,每个顾客唯一接受一辆车的服务一次,问题的目标函数通常有车辆数和运输成本,首先要最小化车辆数,然后最小化总运输成本。

求解VRP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VRP问题的过程如下:

作者简介:尹晓峰(1977-),男,山西万荣人,博士研究生,交通规划与管理专业,主要研究方向为交通规划与政策,交通仿真技术。刘春煌(1945-),男,北京人,研究员,博士生导师,主要研究方向:交通规划与政策,交通仿真技术。张惟皎(1972-),女,湖北人,博士生,主要研究方向:数据挖掘。

(1)首先初始化,设迭代次数为 NC。初始化 $NC=0$,各 $\tau(i,j)$ 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 为客户以及仓库数
tau=ones(n,n)*tau0;
ant_path=zeros(m,n+1);
```

(2)将 m 个蚂蚁置于仓库。

```
ant_path(:,1)=randint(m,1,[1,1]);
```

(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。蚂蚁任务是在约束条件下,访问客户后回到仓库,生成一条回路。设蚂蚁 k 当前所在的顶点为 i,则蚂蚁 k 由点 i 向点 j 移动要遵循

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

的状态变化规则而不断迁移,按不同概率来选择下一点。其中 $allowed_k=\{0,1,\cdots,n-1\}-tabu_k$ 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。(1)式中 τ_{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。α 取值为 1;参数 $\beta>0$ 描述启发函数的重要性;参数 q_0 $(0\leqslant q_0\leqslant 1)$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 $q\leqslant q_0$ 时,按(1)式选最好的路,否则按下式的概率进行选路:

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

相应的 MATLAB 程序如下:

```
current_node=ant_path(k,s-1);  %k 为蚂蚁数目,取值 1..m,s 为问题规模,取 2..n
visited=ant_path(k,:);
to_visit=setdiff([1:n],visited);
c_temp=length(to_visit);
if c_temp~=0
 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(τ_ij)^α(η_ij)^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
  if c_temp==1  %处理最后一个客户
   select=to_visit(c_temp);
  end
  ordinal_of_vehicle=find(ant_path(k,:)==1);
  last_vehicle=ordinal_of_vehicle(length(ordinal_of_vehicle));
     for l=last_vehicle:n+20
        if (ant_path(k,l)~=1) & (ant_path(k,l)~=0)
           total_load=total_load+load(ant_path(k,l));
        end
     end
  if (total_load+load(select))>capacity_limit %不满足约束条件则回到仓库
     select=1;
  end
  total_load=0;
  city_to_visit=select;
  ant_path(k,s)=city_to_visit;
end
```

(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

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

局部信息素更新规则来改变弧上关联的信息素值,其中 $0<\rho<1$ 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:

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

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

其中 $0<\rho<1$ 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 方法局部优化用蚂蚁算法构造的 TSP 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,其中最实用有效的是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。

```
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
```

其中的current_tour是某辆车从仓库出发送货后又回到仓库的路线。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 进一步改进

VRP与TSP最大的不同之处在于对仓库的考虑,在VRP中不仅客户之间的距离重要,而且客户与仓库之间的距离也可以用来进一步提高算法的性能。定义节省量Saving为$u_{ij}=d(c_i,c_{depot})+d(c_{depot},c_j)-d(c_i,c_j)$,$u_{ij}$的值高说明应当在访问客户$i$后直接访问客户$j$,启发式函数$u_{ij}^{\gamma}$可用于状态转换规则中提高解的质量,参数$\gamma$体现了节省量对蚂蚁决策的影响。

定义启发式信息$w_{ij}=(Q_i+q_j)/Q$来提高车辆载重利用率。启发式函数w_{ij}^{γ}也可用于状态转换规则中提高解的质量。参数λ体现了车辆载重利用率对蚂蚁决策的影响。相应的公式(1)以及(2)改变为:

$$v=\begin{cases}\arg\max\limits_{k\in allowed_k}[(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}(u_{ij})^{\gamma}(w_{ij})^{\lambda}] & q\leqslant q_0\ \text{(Exploitation)}\\ V & q>q_0\ \text{(Exploration)}\end{cases}\tag{5}$$

$$p_{ij}^{k}(t)=\begin{cases}\dfrac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}[u_{ij}]^{\gamma}[w_{ij}]^{\lambda}}{\sum\limits_{k\in allowed_k}[\tau_{ik}(t)]^{\alpha}[\eta_{ik}]^{\beta}[u_{ik}]^{\gamma}[w_{ik}]^{\lambda}} & \text{if } j\in allowed_k\\ 0 & \text{otherwise}\end{cases}\tag{6}$$

5 仿真实验

本文选用VRPLIB中列出的Christofides实例[2]。对问题C1分别用各种算法对问题进行求解,每个程序各运行30次,m以及n取客户和仓库数目之和,计算结果如表1所示。可以看出基本蚁群算法和加入2-opt的混合算法相比,后者的解得到明显改善。而考虑了节省量的Saving混合算法($\gamma=5,\lambda=0$)以及考虑车辆载重利用率的Capacity混合算法($\gamma=0,\lambda=5$)都提高了蚁群算法的性能,同时考虑了两者的SC混合算法($\gamma=5,\lambda=5$)的平均结果偏差只有2.66%,并且可以得出最优解。

表1 各种算法的计算结果比较

$\alpha=1,\beta=5,rho=0.75,NC_{max}=30$

算法	平均值	偏差	最好结果	偏差
基本蚁群算法	614.67	17.16%	588.69	12.21%
2-opt混合算法	585.23	11.55%	558.21	6.41%
Saving混合算法	556.45	6.06%	541.57	3.24%
Capacity混合算法	561.28	6.98%	542.23	3.36%
SC混合算法	538.56	2.66%	524.61	0.00%

综上所述,混合算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。

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创造了条件。

(收稿日期:2005年2月)

参考文献

1.V Maniezzo,A Carbonaro.Ant Colony Optimization:an overview[C].In:C Ribeiro.Essays and Surveys in Metaheuristics,Kluwer,2001:21~44

2.N Christofides,A Mingozzi,P Toth et al.Combinatorial optimization[M].John Wiley,Chichester,1979

3.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,2001;(3):14~19

4.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