第31卷第11期
2003年11月

同济大学学报
JOURNAL OF TONGJI UNIVERSITY

Vol.31 No.11
Nov. 2003

基于模式求解旅行商问题的蚁群算法

李炳宇，萧蕴诗

（同济大学 信息与控制工程系，上海 200092）

摘要：群体智能已经被广泛应用于分布式控制、调度、优化等领域.其中蚁群算法已经成为该领域的一个研究热点.在蚁群算法的基础上针对旅行商问题(TSP)，首先提出了小窗口蚁群算法，提高初始解的质量，然后与基于模式的蚁群算法相结合，通过提取模式，改变计算粒度，缩短计算时间，提高计算精度.实验结果表明该算法有较好的效果.

关键词：蚁群算法；小窗口；模式；旅行商问题
中图分类号：TP 301.6　　**文献标识码**：A　　**文章编号**：0253－374X(2003)11－1348－05

Ant Colony Algorithm Based on Model Algorithm for Traveling Salesman Problem

LI Bing-yu, XIAO Yun-shi

(Department of Information and Control Engineering, Tongji University, Shanghai 200092, China)

Abstract: Swarm intelligence has been applied in domains of distributed control, job-shop schedule, and optimization. Ant colony algorithm (ACO), one of swarm intelligence, has become a hot research field. This paper proposes an ant colony algorithm based on little window and obtains models from typical ant algorithm. The algorithm reduces computing time and improves computing accuracy by limiting the size of solution space, extracting models and changing computing granularity. Simulations demonstrate that the improved algorithm can achieve better performance than typical algorithm and some other improved algorithms.

Key words: ant colony algorithm; little window; model; traveling salesman problem(TSP)

在管理科学、计算机科学和工程技术等许多领域中，存在着大量组合优化问题.其中许多问题是NP完全问题.人们先后使用了禁忌搜索(tabu search)算法、模拟退火算法、遗传算法、人工神经网络等启发式搜索算法来求解此类组合优化问题.20世纪90年代初，意大利学者Dorigo等人受到蚂蚁在觅食过程中可以找出巢穴到食物源的最短路径的启发，提出了蚁群算法[1](ant colony algorithm)，将其应用于求解TSP问题[2]、Job-Shop调度[3]、网络路由[4]等问题，并取得了较好的结果.

虽然蚁群算法已经应用于许多组合优化问题，但对蚁群算法本身运行机制的研究却很少.本文通过对蚁群算法的研究发现，蚁群这个自适应系统与许多复杂自适应系统一样存在模式，蚁群就是通过信息素来记忆、更换和组合这些模式，寻找最短路径的.本文在模式的基础上，针对蚁群算法在较大规模TSP问题中计算时间长、运算精度低的问题，通过两方面对算法进行了改进.首先通过限定蚂蚁每步移动的城市数目，大大降低解空间的规模，提高解的质量；在此基础上提取模式，将这些模式看作一个个整体、一个个“积木块”.再在这些“积木块”中寻找最优解.该改进算法的第一步在计算时间和解的质量上都有了很大的提高；在第二步中，由于是在较粗的粒度上计算，将一个高维空间转化为一个低维空间，充分利用了蚁群算法

收稿日期：2003－03－14
作者简介：李炳宇(1977－)，男，山东成武人，博士生.E－mail:yv.tou@163.com

在低维空间有较好搜索性能的特点.实验表明,该算法在 TSP 问题上获得较好的效果.

文章的组织如下:首先介绍基本蚁群算法和目前较常用的最大最小蚁群算法(MMAS)[5];其次分析蚁群算法中模式和信息素的关系;然后详细说明本文的改进算法;最后给出实验结果和结论.

1　基本蚁群算法

蚁群算法源于蚂蚁的觅食行为.蚂蚁在觅食的过程中,在走过的路径下留下一种称之为信息素(pheromone)的物质.其它蚂蚁就是靠这种信息素的指引往返于食物源与巢穴之间.各条路径上的信息素都会随着时间的迁移而不断蒸发减少.但某条路径上走过的蚂蚁越多,则路径上残留的信息素强度就越高.蚂蚁在运动的过程中总是倾向朝信息素强度高的方向运动.换句话说,蚂蚁选择信息素强度高的路径比强度低的路径的概率要高些.设想当几只蚂蚁分别沿着不同的路径回巢,长度越短的路径上信息素的强度越高,其它蚂蚁选择这样路径的概率也越大,选择的蚂蚁越多,路径上的信息素强度也会更高,这样形成正反馈,使更多的蚂蚁集中到最短的路径上来.蚁群算法就是模拟上述蚂蚁觅食行为,设计虚拟的蚂蚁,使其随机搜索不同的路径,并留下会随时间变化而蒸发的"信息素",根据"信息素"强度来寻找最短路径.在蚁群算法中,将自然蚂蚁群体抽象为"人工蚂蚁系统",人工蚂蚁系统具有以下特点:

(1) 每只蚂蚁都有一个记录自己已经走过城市的禁忌表($\mathrm{tabu}_k(s)$),以控制蚂蚁寻找路径的合法性,从起点出发遍历其它所有城市 1 次且仅有 1 次,并最终回到起点.

(2) 每只蚂蚁根据城市 i 到城市 j 的移动概率和禁忌表来决定下一个要到达的城市,在 t 时刻蚂蚁的移动概率表示为

$$p_{ij}^{k}=\begin{cases}[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}\Big/\Big[\sum\limits_{j\in\text{允许的}}[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}\Big], & j\in\text{允许的}\\ 0, & \text{其它的}\end{cases}\tag{1}$$

其中:$\tau(t)$表示在循环 t,边 E_{ij}上的信息素强度;η_{ij}表示 E_{ij}上的路径信息;α,β 是权重参数;$j\in$允许的,表示禁忌表中还未经过的城市.

(3) 周游完成后,蚂蚁在它每一条访问的边上留下信息素.信息素根据下式调整:

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

$$\Delta\tau(t)=\sum_{k=1}^{m}\Delta\tau_{ij}^{k}(t)\tag{3}$$

式(2)中 $\rho\in(0,1)$,$1-\rho$ 表示信息素的蒸发速度,初始时刻 $\tau_{ij}(0)$是 1 个常数;式(3)表示本次循环中路径 E_{ij}上信息素的增量(即为 m 只蚂蚁本次循环留在路径上的信息素之和).$\Delta\tau_{ij}^{k}$是在循环 t 第 k 只蚂蚁留在边 E_{ij}上的信息素的增量.

$$\Delta\tau_{ij}^{k}(t)=\begin{cases}Q/L_k, & \text{在循环 } t \text{ 第 } k \text{ 只蚂蚁经过边 } E_{ij}\\ 0, & \text{其它}\end{cases}\tag{4}$$

其中:信息素强度 Q 为常数;L_k 为蚂蚁 k 所走过的路径总长度.

此外,本文的改进算法就是建立在最大最小蚁群算法(MMAS)和 2 变换邻域局部搜索(MMAS + 2opt)的基础之上的.

2　蚁群算法中的模式分析

在 n 个城市的 TSP 问题中,TSP 的一个解可以表述为一个循环排列 $V=(v_1,v_2,\cdots,v_n,v_1)$,表示 $v_1\rightarrow v_2\rightarrow\cdots\rightarrow v_n\rightarrow v_1$ 的一条路径,v_i 表示该路径中第 i 个经过的城市.设 Λ 为该 TSP 问题的一个解集,则有 $V\in\Lambda$,求解 TSP 问题就是在 Λ 中寻找一个解 V^*,V^* 表示的路径长度最短.

本文将解集 Λ 中 2 个任意解 V_i,V_j 的交集定义为:$H_{ij}=V_i\cap V_j$.H_{ij}表示 V_i,V_j 中相同排列的子集.说明如下,不失一般性,设解 V_i,V_j 分别表示为:$V_i=(\cdots v_k,*\,*\,*,v_m,\cdots,v_w,*\,*\,*\,*\,*,v_u,\cdots)$,$V_j=(\cdots v_w,*\,*\,*\,*\,*,v_u,\cdots,v_k,*\,*\,*,v_m,\cdots)$.$V_i$,$V_j$ 中$(v_w,*\,*\,*\,*\,*,v_u)$,$(v_k,*\,*\,*,$

v_m)表示城市数目和排列顺序相同的子集(子集中包含的城市数目大于等于 2,在不同解中的相对位置可能不同).令 $h_{ij1}=(v_w,*\ *\ *\ *\ *,v_u)$,$h_{ij2}=(v_k,*\ *\ *,v_m)$,则 $H_{ij}=(h_{ij1},h_{ij2})$.如果 V_i,V_j 中有多个相同的子集,则 $H_{ij}=(h_{ij1},h_{ij2},\cdots,h_{ijl})^{\mathrm{T}}$,$1\leqslant l\leqslant n/2$.若解 V_i,V_j 无相同的子集,则 $H_{ij}=\Phi$.

在此引入模式的概念,将 H_{ij} 中的 $h_{ij1},h_{ij2},\cdots,h_{ijl}$ 称为模式,将 $h_{ij1},h_{ij2},\cdots,h_{ijl}$ 看成一个个"积木块",用模式的概念来解释蚁群算法.蚂蚁周游完所有城市后,对最短的路径添加信息素,使得该路径上"好的模式"有更高的概率在下一次周游中被选中.而在第二次周游中,选中"好的模式"的路径也更有可能发现更短的路径.这样周而复始,通过信息素的正反馈,使"好的模式"不断被强化,而通过信息素的蒸发丢弃那些"不好的模式",这样不断缩小搜索空间,积累"好的模式",通过"好的模式"的拼搭结合,从而找到更优的解.因此可以看出,如何有效地发现和利用"好的模式"对蚁群算法解的质量有很大的影响.

3 基于模式的蚁群算法

TSP 是一个强 NP 完全问题,即使在每秒能计算 100 万个解的计算机上遍历 50 个城市 TSP 的解空间,也需要将近 47 个世纪,而 100 个城市的问题则需要 140 个世纪.以往的蚁群算法基本上都是在整个解空间中搜索,通过信息素的启发来寻找最优解.随着城市数目的增大,其解空间急剧放大,使得在有限时间内很难找到满意的结果.而通过调整信息素和加入变异的改进蚁群算法,仍然有计算时间长的缺点.本文通过两步来提取和利用模式以缩短计算时间和提高计算精度.首先,通过小窗口算法将解空间缩小几十个数量级,以更高的概率得到"好的模式";在此基础上,将这些模式看作一个个整体,将一个高维空间转化为一个低维空间,在较粗的粒度上计算以提高计算速度,并利用 2opt 局部搜索方法提高搜索效率.

3.1 小窗口算法

本文作者在对 TSP 问题研究的过程中发现,TSP 问题解空间中绝大部分是劣质解,而且干扰对优质解的寻找.如果能剔除这些劣质解,不但可以缩小解空间的搜索范围,而且可以减少劣质解的干扰,迅速发现优质解.下面用 Oiver30 TSP[6]来说明小窗口算法的原理.图 1 显示了 Oiver30 的最优路线.

在图 1 中可以发现,与每个城市相连的另外 2 个城市是与该城市最近的几个城市中的 2 个.如城市 15,与之相连的城市 14,16 就是距离城市 15 最近的城市中的 2 个.其它的城市也可以发现类似的特点.而绝不会出现如图 2 所示的最优解,或者说如果出现了城市 15～0,16～1 这样的路径,那么存在这样路径的解集就一定不是最优解.更一般地说,在一个 N 城市的 TSP 问题中,任何 1 个城市有 $N-1$ 个从该城市出发的路径,而在这 $N-1$ 条路径中,只有最短的几条路径中的 1 条才可能是组成最优解的路径之一,其它的较长的路径不可能是最优解中的 1 条路径,因此,通过限定蚂蚁每次移动范围,或者说限定蚂蚁在每个城市所能"看到"的下一个城市的数目,来达到剔除劣质解,缩小搜索范围的目的.在算法中为每个城市建一个数组 cityWin[window],保存 window 个距离最近的城市.其中 window 表示窗口的大小,对窗口大小的选择是依据 TSP 问题的维数而定的,维数越高,窗口越大.蚂蚁每次移动就从 cityWin[window]和 tabu[k]的交集中选择移动的城市,根据公式(1),移动到下一个城市.若交集为空,则只在 tabu[k]中选择.小窗口算法(LWMMAS):

Step1.初始化参数:Q,α,β,ρ,taoMax,taoMin,window,N_{cmax},tabu[k],cityWin[m];

Step2.迭代次数:n_c++;

Step3.蚂蚁从 cityWin[window]和 tabu[k]中选择允许到达的城市,根据公式(1)移动;

Step4.若 tabu[k]未满,转致 Step4 继续;

Step5.2opt 局部优化路径;

Step6.记录本次最短路径,置空 tabu[k];

Step7.由公式(2),(4)更新最短路径信息素;

Step8.若 n_c 小于 N_{cmax} 转至 Step2;

Step9.输出最优结果.

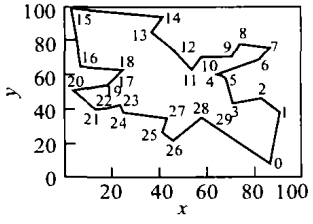

图 1 Oliver 30 的最优路径

Fig.1 Best route of Oliver 30

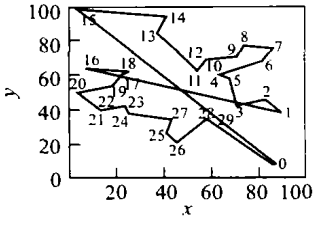

图 2 Oliver 30 的可能路径

Fig.2 Feasible route of Oliver 30

以 50 个城市为例，窗口大小选为 5，则限定后的解空间比原解空间缩小了 30 个数量级，而算法的复杂为 $O(n_c n^2 m)$，与算法 MMAS 相比并没有变化．实验结果表明，小窗口算法较算法 MMAS 有了一定程度的提高，能够得到较好的解．

3.2 基于模式的算法

城市数目对蚁群算法和其它求解 TSP 问题的算法都有较大的影响．基于模式的算法是建立在小窗口算法基础之上的，在小窗口算法得到较好解的基础上，通过提取解中的模式，将一个个模式看作整体，降低实际计算中城市的数目，利用蚁群算法在低维空间有较好搜索性能的特点，提高算法的运行效率．首先，独立运行 2 次小窗口算法，得到 2 个较好的解，比较 2 个解 V_i，V_j，将 2 个解的交集 $H_{ij}=(h_{ij1},h_{ij2},\cdots,h_{ijl})^{\mathrm{T}}$ 保存的一个二维数组 model[][]中，将 $h_{ij1},h_{ij2},\cdots,h_{ijl}$ 看成 1 个个整体，每个模式看作 1 个点（城市），l 个模式看作 l 个点，减少 TSP 中城市的数目．在完成一次运算后，将得到的最优解和 V_i，V_j 比较，再提取其中较好 2 个解的模式重新计算，直到无更好的解或超过最大循环次数停止．基于模式的算法为：

Step1．LWMMAS()，保存解 V_i，LWMMAS()，保存解 V_j；

Step2．getModel(V_i，V_j)，将交集保存到数组 model[][]；

Step3．changStructure(model)，改变城市数目；

Step4．LWMMAS()，保存解 V_k；

Step5．将目前最好的 2 个解保存到 V_i，V_j；

Step6．若未超过最大循环次数且未出现停滞，则转至 Step3；

Step7．输出最优结果．

该算法的复杂度为 $O(N_C(n_c x^2 m+n^2))$，其中 x 是减少城市个数后运算中实际的城市数目，N_C 为提取模式的次数，提取模式的复杂度是 n^2．由于 x 和 N_C 都很小，实际的复杂度大大降低了．

4 实验结果与分析

本文使用的应用实例是采用来自 www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/中的 TSPLIB 实例，Eil51，St70 和 Lin105．

4.1 调整信息素算法（见表 1）与小窗口算法（见表 2）的比较

在此通过与在 MMAS 基础上改进的动态调整信息素的算法的比较，证明小窗口算法的有效性．

各种参数的设定如下，$\alpha=4$，$\beta=1$，$Q=\mathrm{taoMax}=1$，$\mathrm{taoMin}=0.000\ 24$，$\mathrm{window}=5$，$\rho=0.8$，蚂蚁数等于城市数，即 $m=n$．Eil51 问题的已知最优解是 426，由表 1 和表 2 得到的平均结果、偏差（$\sqrt{\sum_n (X_i-X_{\mathrm{Best}})^2/n}$）、最优解、最差解等统计结果见表 3．

表 1　动态调整信息素算法 Eil51 计算结果[7]

Tab.1　Simulation results of Eil51 based on adaptively adjusting pheromone algorithm

循环次数 $N_{cmax}=250$，平均时间(avgtime)=100.54 s				
426	428	428	427	429
429	430	426	426	428
429	427	428	428	427
426	431	430	427	428
430	426	427	430	428

表 2　小窗口算法 Eil51 计算结果

Tab.2　Simulation results of Eil51 based on little window algorithm

循环次数 $N_{cmax}=50$，平均时间(avgtime)=28.7 s				
426	428	428	427	429
427	429	430	429	428
426	427	427	428	427
427	426	429	428	428
427	427	426	427	428

表 3　两种算法的统计结果

Tab.3　Statistical results of the two algorithms

实例	平均结果	偏差	最优解	最差解	平均运行时间/s	循环次数
Eil51(小窗口算法)	427.56	1.887	426	430	28.70	50
Eil51(动态调整信息素算法)	427.96	2.441	426	431	100.54	250

由表 3 可见,本文算法各项均大大优于改进后的动态调整信息素算法,尤其是循环次数远远小于该算法,而在实际中运算中大多没有循环 N_{cmax}次就已经找到最优解了.这是因为小窗口算法比动态调整信息素算法的解空间要小 30 个数量级,前者的运算效率要好于后者.而且本文还对主要参数 Q,α,β,ρ 做了测试,发现对算法无明显影响,因此本算法无需像以往许多蚁群算法需要大量实验来寻找合适的参数.

4.2 证明基于模式算法的有效性

在基于模式的算法中,设定小窗口算法中的最大循环次数 $N_{cmax}=10$,模式比较的次数 $N_C=3$.由表 4 可得 Eil51 基于模式算法的统计结果,平均结果为 427.04,偏差 1.327,最优解 426,最差解 429,平均运行时间 9.8 s.可以看出,各项计算结果都明显好于小窗口算法,尤其是计算时间大大缩短了.

在表 5 中列出了 St70(70 个城市,已知最优解是 675)20 次的计算结果,最优解 675,最差解 680,平均值 676.1,偏差 1.95,平均计算时间 29.6 s.从计算结果上可以看出,虽然城市数目增大了,但与小窗口算法计算 Eil51 的时间相差不到 1 s.因此可以证明模式算法的有效性.应用模式算法计算 Lin105(105 个城市,已知最优解是 14 379),最优解 14 379,最差解 14 448,平均值 14 397.65,循环次数 10 次,平均计算时间 187.7 s,也取得了很好的结果.上面的数据说明基于模式的蚁群算法能够取得较好的实验结果,同时也说明该算法在解决组合优化问题方面较同类算法有一定的优势,还需要说明的是,在这些实验中算法结束条件都是由一个固定的循环次数 N_{cmax}决定,而算法获得最优解时要小于有时远小于这个固定循环数,也就是说算法性能较列表说明的还要好一些.

表 4 Eil51 基于模式算法的计算结果

Tab.4 Simulation results of Eil51 based on model algorithm

循环次数 $N_{cmax}=10$,平均时间(avgtime)=9.8 s,$N_C=3$								
426	428	427	427	426	427	428	428	428
427	427	429	426	426	426	427	427	427
428	427	427	426	427	428	426		

表 5 St70 基于模式算法的计算结果

Tab.5 Simulation results of St70 based on model algorithm

循环次数 $N_{cmax}=10$,平均时间(avgtime)=29.6 s,$N_C=3$									
675	676	675	675	680	676	680	676	675	676
675	675	676	679	675	676	677	675	675	675

5 结论

实验说明小窗口算法通过限定计算空间在一定程度上改善了解的质量,为基于模式的蚁群算法提供了较好的初始解,保证了有较好的模式,而基于模式的蚁群算法在提取模式的基础上改变计算粒度,提高运算速度,克服了以往蚁群算法的计算时间长、精度低的缺点,使得蚁群算法有了显著的提高.由于搜索速度的提高,本文算法可以满足更多实际问题要求,有利于蚁群算法的应用和推广.

参考文献:

[1] Dorigo M, Maniezzo V, Colorni A. The ant system: Optimization by a colony of cooperating agents[J]. IEEE Transactions on Systems, Man, and Cybernetics-Part B, 1996, 26(1): 29 - 41.

[2] Dorigo M, Gambardella L. Ant colony system: A cooperative learning approach to the traveling salesman problem[J]. IEEE Transactions on Evolutionary Computation, 1997, 1(1): 53 - 66.

[3] Colorni A, Dorigo M, Maniezzo V, et al. Ant system for job-shop scheduling[J]. Belgian Journal of Operations Research, Statistics and Combuter Science, 1994, 34(1): 39 - 53.

[4] Gianni Di Caro, Marco Dorigo. AntNet: Distributed stigmergetic control for communications networks[J]. Journal of Artificial Intelligence Research, 1998, (9): 317 - 355.

[5] Thomas Stützle, Holger Hoos. MAX-MIN ant system and local search for the traveling salesman problem[A]. Proc IEEE International Conference on Evolutionary Computation(ICEC'97)[C]. Indianapolis: [s. n.], 1997. 309 - 314.

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

[7] 覃刚力,杨家本.自适应调整信息素的蚁群算法[J].信息与控制,2002,31(3):198 - 201.