第19卷 第8期 控 制 与 决 策 2004年8月
Vol. 19 No. 8 *Control and Decision* Aug. 2004

文章编号：1001-0920(2004)08-0885-04

求解TSP问题的模式学习并行蚁群算法

萧蕴诗，李炳宇，吴启迪
（同济大学 电子与信息工程学院，上海 200092）

摘 要：针对大规模旅行商问题(TSP)会遇到计算时间过长以及计算效率降低的问题，将并行计算和模式学习引入蚁群算法，通过各个节点机提取模式，在各节点间筛选和交流优良模式，以改变计算粒度，达到缩短计算时间、提高计算效率的目的.实验结果表明该算法取得了较好的效果.

关键词：旅行商问题；蚁群算法；模式学习；并行策略
中图分类号：TP301.6 **文献标识码**：A

Parallel model-learning ant colony optimization algorithm for TSP

XIAO Yun-shi, LI Bing-yu, WU Qi-di
(School of Electronic and Information Engineering, Tongji University, Shanghai 200092, China. Correspondent: XIAO Yun-shi, E-mail: yv.tou@163.com)

Abstract: For large-scale traveling salesman problems (TSP), a new ant colony optimization (ACO) is proposed by integrating parallel computation and model-learning into ACO. The new ACO can reduce the computing time and get better results by screening out, exchanging good model and changing computing granularity. Simulation result demonstrates that the improved algorithm can achieve good performance.

Key words: TSP; ant colony optimization; model-learning; parallelization strategy

1 引 言

在网络和车辆路由以及VLSI芯片设计和电路版布局等领域中存在的问题可抽象为旅行商问题(TSP).TSP问题是典型的NP-hard组合优化问题，其特点是迄今还没有一种算法能在多项式时间里找到最优解.能保证此类问题找到最优解的确定性算法，其运行时间呈指数复杂度，因此对于大规模问题，要在多项式时间内结束，只能用近似算法得到可以接受的解.人们曾先后使用禁忌搜索(Tabu Search)算法、模拟退火算法、遗传算法以及人工神经网络等启发式搜索算法来求解此类NP-hard问题.20世纪90年代初，意大利学者Dorigo等受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 optimization)，并将其成功地应用于求解TSP问题[2].但随着TSP问题规模的增加，蚁群算法会出现运算时间过长、运算效率降低等问题.

文献[3]将并行策略引入蚁群算法，设计、比较了同步并行计算和异步并行计算对蚁群算法的影响，但未对蚁群算法本身的机制进行深入分析.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

收稿日期：2003-08-04；**修回日期**：2003-09-26.
基金项目：国家自然科学基金资助项目(70271035，60104004)；国家973子项目(2002CB312202)；上海市启明星计划资助项目(03QG14053).
作者简介：萧蕴诗(1946—)，男，湖北武汉人，教授，博士生导师，从事智能自动化、复杂系统理论与方法等研究；李炳宇(1977—)，男，山东成武人，博士生，从事群体智能、复杂系统理论与方法等研究.

过信息素来记忆、更换和组合这些模式，寻找最短路径. 因此，本文将异步并行计算、优良模式学习与蚁群算法相结合，提出一种优良模式学习并行蚁群算法. 各个节点机各自独立运算、分别学习，通过筛选交流学习优良模式，达到降低计算复杂度，提高运算效率，得到更优解的目的. 实验表明，该算法在 TSP 问题上获得了较好的效果.

2 基本蚁群算法

蚁群算法是一种仿生算法，源于蚂蚁的觅食行为. 蚂蚁在觅食过程中通过一种称为信息素(pheromone)的物质交流信息. 每次觅食，蚂蚁离巢分头寻找食物，发现食物后蚂蚁会返回巢穴通知同伴，在走过的路径上留下信息素. 其他蚂蚁会依据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会随时间的迁移而不断蒸发减少，同时某条路径上的信息素强度会随着走过的蚂蚁数量的增多而增强. 蚂蚁在运动过程中总是倾向于朝信息素强度高的方向运动. 换言之，蚂蚁选择信息素强度高的路径比选择强度低的路径的概率高. 设想当多只蚂蚁分别沿着不同的路径回巢时，长度越短的路径上的信息素强度越高，其他蚂蚁选择这样路径的概率越大；选择的蚂蚁越多，路径上的信息素强度越高. 这样便形成正反馈，使更多的蚂蚁集中到最短的路径上. 蚁群算法的数学模型可用 3 条规则和 4 个表达式描述. 蚂蚁必须遵循的规则是：

1) 每只蚂蚁必须依据以城市距离和连接边上信息素数量为变量的概率函数(如式(1) 所示)，决定选择下一个城市(设 $\tau_{ij}(t)$ 为 t 时刻边 $E(i,j)$ 上信息素的强度) 的概率.

$$p_{ij}^{k}=\begin{cases}\dfrac{[\tau_{ij}(t)]^{\alpha}\cdot[\eta_{ij}]^{\beta}}{\sum\limits_{j\in\text{allowed}}[\tau_{ij}(t)]^{\alpha}\cdot[\eta_{ij}]^{\beta}}, & j\in\text{allowed};\\ 0, & \text{otherwise}.\end{cases}\tag{1}$$

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 为权重参数；$j\in$ allowed 表示禁忌表中未经过的城市.

2) 每只蚂蚁都有一个记录自己已经走过城市的禁忌表($\text{tabu}_k(s)$)，蚂蚁必须根据禁忌表和概率函数寻找下一个城市，以保证该蚂蚁从起点出发遍历其他所有城市一次且仅一次，并最终回到起点.

3) 完成周游后，蚂蚁在其每一条访问过的边上留下信息素，信息素根据如下公式调整：

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

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

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

式(2) 中：$\rho\in(0,1)$，$1-\rho$ 表示信息素的蒸发速度，初始时刻 $\tau_{ij}(0)$ 是一个常数；式(3) 表示本次循环中路径 E_{ij} 上信息素的增量(即 m 只蚂蚁本次循环留在路径上的信息素之和)，$\Delta\tau_{ij}^{k}$ 为循环 t 第 k 只蚂蚁留在边 E_{ij} 上的信息素的增量；式(4) 中：信息素强度 Q 为常数，L_k 为蚂蚁 k 所走过的路径的总长度. 此外，蚁群算法与局部搜索方法相结合可提高搜索效率，本文提出的改进算法就是建立在最大最小蚁群算法[4] 和局部搜索(MMAS + 2opt) 的基础之上.

3 模式学习并行蚁群算法

3.1 算法的基本思想

TSP 是一个 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 问题的解空间，也需近 47 个世纪，而 100 个城市的问题则需 140 个世纪. 因此，解决 TSP 问题的关键是如何处理解空间. 对于这个问题可从两方面考虑：1) 随着迭代运算，计算能力不断提高；2) 随着迭代运算，解空间不断缩小. 但因前者受限于计算机硬件，所以应从后者入手，通过缩小解空间来解决计算爆炸问题.

TSP 问题的解是一组边的排列组合，缩小解空间就是减少边的数目. 在对大量 TSP 问题解的观察中发现，一些次优解与最优解的某些局部路径排列是相同的，在此简称为局部最优解(相对于完整路径而言是局部解). 假设最优解 V^{*}，次优解 V_i 和 V_j 表示为

$$V^{*}=(*\,*\,*,v_k,v_l,v_m,*\,*\,*\,*\,*),$$
$$V_i=(*\,*\,*\,*,v_k,v_l,v_m,*\,*\,*\,*),$$
$$V_j=(*\,*\,*\,*\,*,v_k,v_l,v_m,*\,*\,*).$$

其中局部路径 $V=(v_k,v_l,v_m)$，尽管在上述 3 个解中的位置不同(前后城市排列“*”是不同的)，但 3 个解中局部路径 $V=(v_k,v_l,v_m)$ 的排列却是一定的. 如果把 $V=(v_k,v_l,v_m)$ 看作一个整体，记为 v_{klm}，则 V^{*} 和 V_i,V_j 可重新表示为

$$V^{*'}=(*\,*\,*,v_{klm},*\,*\,*\,*\,*),$$
$$V_i'=(*\,*\,*\,*,v_{klm},*\,*\,*\,*),$$
$$V_j'=(*\,*\,*\,*\,*,v_{klm},*\,*\,*).$$

假设这个 TSP 问题的城市数目为 N，那么变换后的城市数目为 $N-2$. 虽然计算规模减小了，但不会影响次优解 V_i,V_j 到最优解 V^{*} 的变化，即不会丢失最优解. 因此，如果存在多个这样的局部最优解，则计算规模会降低很多，而且更易得到最优解.

本文的思路是:在计算过程中不断寻找这种局部最优解,通过对局部最优解的归并,达到降低计算规模,提高运算效率的目的.但如何选取局部最优解成为难点,因为无法确定在得到的多个次优解中,哪些局部排列一定会出现在最优解中.因此,本文提出一种模式学习并行蚁群算法.实际上,这些局部排列可看作一种"模式"、"积木块",本文算法的实质就是在迭代计算过程中,不断寻找、学习、积累"好的模式",并将这些"好的模式"象积木块一样重新排列组合,以得到最优解.随着"模式"的积累,"积木块"会越来越多、越来越大,所以计算的粒度会逐渐变粗,解空间会相对变小,像滚雪球一样计算速度也会越来越快,本文设定的目标也就达到了.

蚁群算法的本质也是一种模式的选择和积累.蚂蚁在 1 次周游完所有城市后,对最短的路径添加信息素,使得该路径上"好的模式"具有更高的概率,以便在下一次周游中被选中.而在第 2 次周游中,选中"好的模式"的路径更有利于发现更短的路径.这样周而复始,通过信息素的正反馈,使"好的模式"不断强化,"不好的模式"则通过信息素的蒸发而丢弃.这样不断缩小搜索空间,积累"好的模式",并通过"好的模式"的拼搭结合,找到更优的解.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.

蚁群算法在处理大规模 TSP 问题时,依然会出现计算时间长、计算效果差的问题,所以通过外在的模式学习与蚁群算法内在的模式处理相结合,得到一种较好的算法.

3.2　模式的获取

模式的获取是通过不同次优解之间的交集实现的.TSP 问题的解的表现形式是一组边的排列,将不同解中相同的局部排列片段提取出来作为待选模式,具体过程如图 1 ～ 图 3 所示.图 1 和图 2 是两个不同的解,但两个虚线圈内的局部排列是相同的.两个解的交集就是两个解中相同的部分,所以两个虚线圈内的部分是这两个解的交集.将这两个虚线圈内的路径看作一个整体,从原路径中删除,则原路径解 1 最终变成图 3,城市数目减少了.算法中增加的集合交集运算是线性的,因此不会对算法的计算时间产生较大的影响.

然而,这样得到的模式未必是最优解中的模式,但这些模式很可能出现在最优解中.因此,要在计算过程中对得到的模式不断进行调整改进,以期得到最优模式.

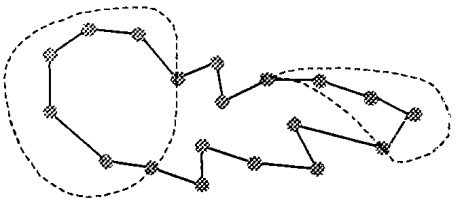


图 1　原路径解 1

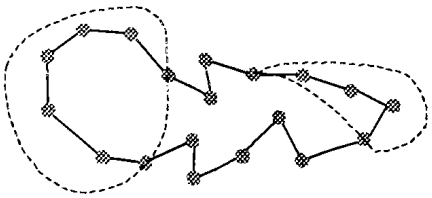


图 2　原路径解 2

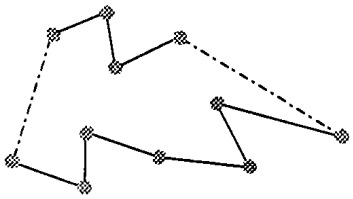


图 3　删除相同部分后的新解

3.3　模式学习并行蚁群算法描述

模式学习并行蚁群算法采用主从式.在各个节点上运行 MMAS + 2opt 算法,当其运行一段时间后,将得到的最好解传送给主程序.主程序从接收到的解中选出最好的几个解,并根据 3.2 节的方法得到多个模式;然后将几个不同的模式传送回各节点,MMAS + 2opt 根据得到的模式在一个较粗的粒度上运行;运行一段时间后,再将最好的解传送回主程序,用同样方法获取模式,传回各节点;如此往复,直到传回主程序的最好结果不再改进为止.当结果不再改进时,表明已学不出更好的模式,必须对模式进行处理.由于模式是不断积累的,随着计算的进行,模式不断由小"积木块"聚合成大"积木块".模式使计算粒度变粗,计算速度会逐渐加快,但使解改进的可能却越来越小.因此,要将大"积木块"重新打碎成小"积木块",让其再次重新组合成大"积木块",以期得到更好的解;将模式调整后再传回各节点进行计算,直到满足终止条件.

本文算法是在曙光天潮 TC3000 服务器上实现的.曙光天潮系列"曙光 3000"超级服务器是基于分布式存储和消息传递体系结构的通用的可扩展并行计算机系统,所有节点都是 4 路 SMP,节点 CPU 采用先进的 Power3- Ⅱ 微处理器,每个节点上运行完整的 IBM AIX 操作系统.本算法共使用了 3 个节

点，每个节点有 4 个 CPU 和 4G 内存，使用 MPI 编程.

每个节点运行 4 个进程，共 12 个进程，0 进程设为主进程，负责模式处理. 每个进程都运行 MMAS + 2opt，各个进程将计算结果发送到 0 进程(包括 0 进程自身的结果). 0 进程将 12 个结果排序，取出 4 个最好的结果. 按排序索引，分别将解 1 和 2，2 和 3，3 和 4 的交集取出，作为优良模式分别传到节点 0，1，2 上的进程，因此不同节点上的模式是不同的. 各节点分别计算，再将最好的结果发回到 0 进程，如此往复直到解的质量不再改进. 此时，将“积木块”中城市个数大于总城市数目 1/8 的大“积木块”截成小于等于总城市数目 1/8 的小“积木块”，将这些小“积木块”发回各节点，重新排列和寻找更好的模式. 在此，终止条件是上述模式的截取次数大于等于 5.

算法步骤如下：

Step1：初始化参数：Q，α，β，ρ，taoMax，taoMin，window，NcMax，tabu[k]，cityWin[m]；

Step2：同时开始 12 个进程，12 个进程分别运行 MMAS + 2opt；

Step3：各进程判断结果是否凝滞，如果是则转 Step5，否则转 Step3；

Step4：进程传送各自最好结果到主进程 0；

Step5：对结果进行处理，获取模式，判断全局最好解是否改进，如果是，则将不同的模式发回到各个节点，转 Step3，否则转 Step7；

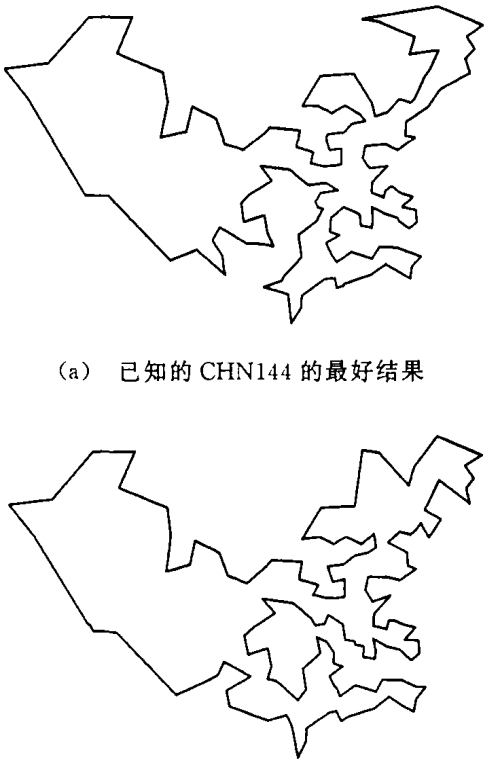
Step6：判断终止条件，若不满足，则将长的模式截短，传回各个节点，转 Step3，否则转 Step7；

Step7：输出最优结果.

4 实验结果

以 CHN144 为例[5]，各参数设定如下：$\alpha = 4$，$\beta = 1$，Q = taoMax = 1，taoMin = 0.000 024，window = 5，$\rho = 0.8$，蚂蚁数等于城市数，即 $m = n = 144$.

已知 CHN144 的最好结果是 30 380 km，如图 4(a) 所示. 采用本文算法得到的结果为 30 349 km，如图 4(b) 所示. 由图 4 可以看出，本文算法得到了一个更好的解，从而证明了该算法的有效性.



(a) 已知的 CHN144 的最好结果

(b) 采用本文算法得到的结果

图 4 实验结果

5 结 论

本文在对 TSP 问题分析的基础上，找出局部排列与全局最优解之间的关系，将并行计算、模式学习与蚁群算法相结合，通过提取模式来改变计算粒度，达到降低解空间、提高计算效率的目的. 该算法在曙光天潮 TC3000 服务器上计算的结果证明了该算法的有效性，从而为 TSP 问题的研究提供了一种有效的方法，同时为采用蚁群算法解决其他问题提供了新的思路.

参考文献(References)：

[1] Dorigo M, Maniezzo V, Colorni A. The ant system: Optimization by a colony of cooperating agents [J]. *IEEE Trans 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 Trans on Evolutionary Computation*, 1997, 1(1): 53-66.

[3] Bullnheimer B, Kotsis G, Strauss C. Parallelization strategies for the ant system[R]. Vienna: University of Vienna, 1997.

[4] Thomas Stützle, Holger Hoos. MAX-MIN ant system and local search for the traveling salesman problem [A]. *Proc IEEE Int Conf on Evolutionary Computation (ICEC'97)*[C]. Indianapolis, 1997. 309-314.

[5] 康立山，谢云，尤失勇，等. 非数值并行算法(一)[M]. 北京：科学出版社，1998.