基于局部优化策略求解 TSP 的蚁群算法*

龚本灿[1,2]，李腊元[2]，蒋廷耀[1]，汪祥莉[2]
(1.三峡大学 电气信息学院，湖北 宜昌 443002；2. 武汉理工大学 计算机学院，武汉 430063)

摘 要：为了克服基本蚁群算法收敛速度慢、易于停滞的缺陷，提出了一种基于局部优化策略的蚁群算法(LOACA)。该算法根据 TSP 的特点，采用了三种局部优化算子来交换搜索路径中城市的位置，以改进解的质量。以 TSP 为例进行的实验结果表明，该算法优于 ACA 和 ACAGA。

关键词：蚁群算法；局部优化；旅行商问题
中图分类号：TP301.6 **文献标志码**：A **文章编号**：1001-3695(2008)07-1974-03

Ant colony algorithm based on local optimization for TSP

GONG Ben-can[1,2], LI La-yuan[2], JIANG Ting-yao[1], WANG Xiang-li[2]
(1. *College of Electrical Engineering & Information Technology, China Three Gorges University, Yichang Hubei* 443002, *China*; 2. *College of Computer Science & Technology, Wuhan University of Technology, Wuhan* 430063, *China*)

Abstract: This paper proposed an ant colony algorithm based on local optimization (LOACA) to avoid the default of slow convergence speed and early stagnation in the basic ant colony algorithm (ACA). According to the features of TSP, it used three local optimization operations to exchange the position of cities in the search paths to gain the better solutions. Experimental results for solving TSP show that the proposed algorithm performs better than ACA and ACAGA.
Key words: ant colony algorithm; local optimization; TSP (traveling salesman problem)

旅行商问题是一个经典的组合优化问题，在给定 n 个城市及其坐标位置的情况下，它要求一条经过每个城市一次且仅一次的最短 Hamilton 回路。设 $G=(V, E, d)$ 表示一个加权图。其中：V 是城市 v_i 的集合；E 是连接两个城市的边集；d 是边的权值，即两个城市之间的距离。TSP 的数学描述为 $T=\min(\{\sum_{i=1}^{n} d(v_i, v_{i+1}) + d(v_n + v_1)\})$。TSP 在许多工程领域具有广泛的应用价值，如电路板布线、VLSI 芯片设计、机器人控制、交通路由等。但 TSP 的求解是 NP-hard 问题。随着城市数目的增多，问题空间将呈指数级增长，无法在多项式时间内找到精确解。最近许多研究者用各种启发式算法来求解该问题，比较流行的算法有蚁群算法(ACA)[1]、遗传算法(GA)[2]、模拟退火(SA)[3]、禁忌搜索(TS)[4]、神经网络(NN)[5]、粒子群优化算法(PSO)[6]、免疫算法(IA)[7] 等。在这些算法中蚁群算法取得了较好的效果。蚁群算法是 20 世纪 90 年代由意大利学者 Macro Dorigo 等人提出的一种仿生算法。它模拟了自然界中蚂蚁的觅食行为，蚂蚁在觅食过程中会在经过的路径上留下一种称为信息素的物质，蚂蚁选择路径的概率与路径上信息素的强度成正比；经过某条路径的蚂蚁越多，在该路径上留下的信息素也越多，后面的蚂蚁选择这条路径的概率就越大。通过正反馈，蚂蚁很快能够找到从蚁巢到食物源的最短路径。蚁群算法具有很多优点，如很强的发现较好解的能力、鲁棒性强、分布式计算、简单、易于与其他算法结合等；但也有一些缺陷，如收敛速度慢、易陷入局部最优解。局部优化能在一定程度上缓解蚂蚁算法存在的不足，比较常见的局部优化方法有 2-OPT、3-OPT、LK 等。本文在这些方法的基础上，提出了三种局部优化算子，有效地改进了蚁群算法的性能。

1 基本蚁群算法

下面以 TSP 为例说明基本蚁群算法模型。首先将 m 只蚂蚁随机放置在 n 个城市，位于城市 i 的第 k 只蚂蚁选择下一个城市 j 的概率为

$$P^k(i, j)=\begin{cases}\dfrac{[\tau(i, j)]^{\alpha}\times[\eta(i, j)]^{\beta}}{\sum_{s\mid \text{tabu}_k}[\tau(i, s)]^{\alpha}\times[\eta(i, s)]^{\beta}} & \text{if } j\mid \text{tabu}_k\\ 0 & \text{otherwise}\end{cases} \quad (1)$$

其中：$\tau(i, j)$ 表示边 (i, j) 上的信息素浓度；$\eta(i, j)=1/d(i, j)$ 是启发信息，$d(i, j)$ 是城市 i 和 j 之间的距离；α 和 β 反映了信息素与启发信息的相对重要性；tabu_k 表示蚂蚁 k 已经访问过的城市列表。

当所有蚂蚁完成周游后，按式(2)(3)进行信息素更新。

$$\tau(i, j)=\rho\times\tau(i, j)+\Delta\tau(i, j)$$

$$\Delta\tau(i, j)=\sum_{k=1}^{m}\Delta\tau^k(i, j) \quad (2)$$

$$\Delta\tau^k(i, j)=\begin{cases}Q/L_k & (i, j)\in R_k\\ 0 & \text{otherwise}\end{cases} \quad (3)$$

其中：$0<\rho<1$ 是信息素挥发因子；$\Delta\tau^k(i, j)$ 表示第 k 只蚂蚁在边 (i, j) 上留下的信息量；Q 为常数；R_k 表示第 k 只蚂蚁走

收稿日期：2007-07-23；修回日期：2007-10-31 基金项目：国家自然科学基金资助项目(60672137)；教育部博士点基金资助项目(20060497015)

作者简介：龚本灿(1970-)，男，湖北监利人，副教授，博士研究生，主要研究方向为无线传感器网络与路由算法(gonbc@sina.com)；李腊元(1946-)，男，教授，博导，主要研究方向为高性能网络技术及通信协议；蒋廷耀(1969-)，男，副教授，博士，主要研究方向为计算机网络及信息安全；汪祥莉(1978-)，女，讲师，博士研究生，主要研究方向为无线传感器网络与路由算法.

过的路径; L_k 为路径长度。

2　局部优化算子

2.1　局部优化算子 1

设路径为: ～, k, l, l', ～, p', p, q, q', ～, $r', r, s,$ ～, 从当前节点 k 开始, 依次检测随后的节点。如果某节点 q 满足条件 $d(k,l) > d(k,q)$, 即节点 k 和 l 之间的距离大于节点 k 和 q 之间的距离, 则从节点 q 继续向后检测; 如果有节点 r 满足条件 $d(k,l) + d(p,q) + d(r,s) > d(k,q) + (r,l) + d(p,s)$, 则进行变异, 优化后的路径为: ～, k, q, q', ～, r', l, l, l', ～, $p', p, s,$ ～。根据 TSP 的特点, 变异时应保持子路径$(l'$, ～, $p)$ 和$(q'$, ～, $r)$ 中各节点间的邻接关系不变。

为了便于处理, 将路径用单链表表示, 单链表中节点的结构为 | c | n |。其中: c 表示路径中城市的编号; n 是指向下一个节点的指针。程序伪代码如下:

```
//定义指向节点 k, l, p, q, r, s 的指针
LNode * L1,* L2,* L3,* L4,* L5,* L6;
L1 = H; //H 为单链表的头指针
L2 = L1→n; L3 = L2→n; L4 = L3→n;
for ( i = 1; i < CityCount - 5; i + + )
    while( L4→n→n! = NULL)
    flag = 0; //优化标志
    if( d[ L1→c] [ L2→c] > d[ L1→c] [ L4→c] )
      L5 = L4→n; L6 = L5→n;
      while( L6! = NULL)
        d1 = d[ L1→c] [ L2→c] + d[ L3→c] [ L4→c] + d[ L5→c]
[ L6→c] ;
        d2 = d[ L1→c] [ L4→c] + d[ L5→c] [ L2→c] + d[ L3→c]
[ L6→c] ;
        if( d1 > d2)
          L1→n = L4; L5→n = L2; L3→n = L6; //优化
          flag = 1; break;
        else
          L5 = L5→n; L6 = L6→n;
        End if
    End while
  End if
  if( flag = = 1) //进行优化
    L2 = L1→n; L3 = L2→n; L4 = L3→n; //指针置位
  else
    L3 = L3→n; L4 = L4→n;
    End if
  End while
  L1 = L1→n; L2 = L1→n; L3 = L2→n; L4 = L3→n;
  End for
```

2.2　局部优化算子 2

设路径为: ～, $k, l,$ ～, $p, q,$ ～。从当前节点 k 开始, 依次检测随后的节点。如果有一个节点 p 满足条件 $d(k,l) + d(p,q) > d(k,p) + (l,q)$, 则进行变异, 优化后的路径为: ～, $k, p,$ ～, $l, q,$ ～。变异时应保持子路径$(l,$ ～, $q)$ 中各节点间的邻接关系不变, 但节点顺序相反, 变为$(p,$ ～, $l)$。程序伪代码如下:

```
for ( i = 0; i < CityCount - 3; i + + ) //对每个城市
    j = i + 2;
    while( j < CityCount - 1)
      d1 = d[ tabu[ i] ] [ tabu[ i + 1] ] + d[ tabu[ j] ] [ tabu[ j + 1] ] ;
      d2 = d[ tabu[ i] ] [ tabu[ j] ] + d[ tabu[ i + 1] ] [ tabu[ j + 1] ] ;
      if( d1 > d2) //距离变短, 应变异
        switch_num = ( j - i) /2; //交换次数
        for( k = 1; k < = switch_num; k + + )
          temp = tabu[ i + k] ; //城市号
          tabu[ i + k] = tabu[ j + 1 - k] ;
          tabu[ j + 1 - k] = temp;
        end for
        j = i + 2;
      else
        j + + ;
      end if
    end while
end for
```

2.3　局部优化算子 3

TSP 具有邻域特征, 蚂蚁在选择下一个城市时, 最优路径只可能出现在附近的几个城市, 这一搜索范围称为候选窗口。候选窗口的大小应根据 TSP 的规模而定, 取值太小, 会降低解的质量; 取值过大, 则影响搜索速度。蚂蚁总是优先选择候选窗口中的城市, 只有当候选窗口中的城市都走过时, 才选择窗口外的城市。搜索结束后, 根据候选窗口对路径进行优化, 如果将候选窗口内的节点交换到当前节点附近后距离更短, 则进行变异。

设路径为: ～, k, l, l', ～, p', p, t, q, q', ～, $r', r, s,$ ～, 对任一节点 t, r 和 l 是 t 候选窗口中的城市。如果 $d(t,q) + d(r,s) > d(t,r) + (q,s)$, 则进行变异, 优化后的路径为: ～, k, l, l', ～, p', p, t, r, r', ～, $q', q, s,$ ～; 如果 $d(p,t) + d(k,l) > d(l,t) + (k,p)$ 则进行变异, 优化后的路径为: ～, k, p, p', ～, l', l, t, q, q', ～, $r', r, s,$ ～。

另外, 本文采用了最优路径更新策略, 每轮搜索结束后, 只对最短路径进行信息素更新。

对 Kroa100, 各算子优化前后的路径如图 1 所示。路径长度分别为: 优化前(28 596), 算子 1 优化后(26 439), 算子 2 优化后(23 012), 算子 3 优化后(21 282)。

(a) 优化前　(b) 算法 1 优化后　(c) 算法 2 优化后　(d) 算法 3 优化后

图 1　各算子优化前后的路径

3　实验结果

本文选用了国际上通用的 TSP 测试库 TSPLIB 中的多个实例, 用 VC + + 编程进行测试, 并将 LOACA 分别与结合遗传算法的蚁群算法 ACAGA[8] 和基本蚁群算法 ACA 进行比较。

实验参数设置为: 蚂蚁数 = 50; $\alpha = 1$; $\beta = 3$; $\rho = 0.9$; 信息素初始值 = 0.1; 迭代次数 = 1 000; 对 Eil51、Eil76 和 Kroa100, 候选窗口大小 $w = 10$, 对 Lin318, $w = 30$; 对 Eil51 和 Eil76, $Q = 20$, 对 Kroa100, $Q = 1\ 000$, 对 Lin318, $Q = 2\ 000$。

对每个 TSP 共进行 10 次实验, 取平均值, 实验结果如表 1 所示。表中* 标注的数据选自文献[8]。

表 1　不同算法的对比实验结果

实例名称	Eil51	Eil76	Kroa100	Lin318
已知最优值	426*	538*	21 282*	42 029*
ACAGA 最好值	426*	540*	21 292*	45 733*
ACA 最好值	427*	543*	21 389*	48 416*
LOACA 最好值	426	538	21 282	42 091
LOACA 平均值	426.1	538	21 282	42 106.2

从表 1 可以看出, 提出的算法在解的质量上明显优于 ACAGA 和 ACA, 对 Eil51、Eil76 和 Kroa100, 分别有 9、10 和 10 次找到了已知最优解。各 TSP 实例所求得的最好路径如图 2 所示。

(a) Ei151 (b) Ei176 (c) Kroa100 (d) Lin318

图 2 各 TSP 实例的最好路径

对 Kroa100、LOACA 和 ACA 算法的收敛特性比较如图 3 所示。从图中可以看出,对于基本蚁群算法,路径长度变化大(22 288 ~38 218,前五次迭代在图中未列出),收敛速度慢;而优化算法路径长度变化小(21 282 ~21 610),收敛速度快,仅用了 25 轮即取得已知最优解 21 282。

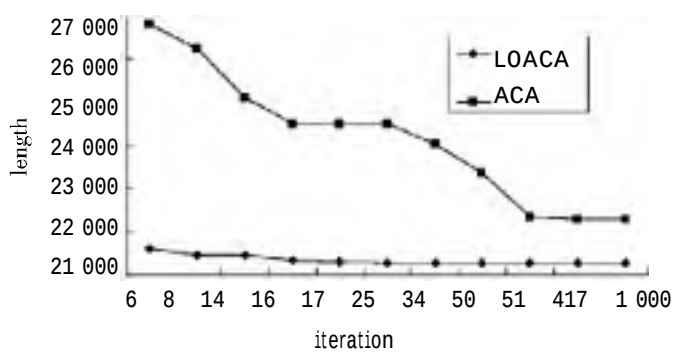

图 3 LOACA 和 ACA 的收敛特性对比

4 结束语

本文根据 TSP 的特点,设计了三种局部优化算子,每一轮搜索结束后,采用该算子对结果路径进行变异,以寻求更优解。局部优化加快了蚂蚁算法的收敛速度,避免了早熟和停滞现象的发生,增强了寻优能力。经过多个 TSP 实例测试,实验结果表明:对中小规模的 TSP,该算法基本上能找到最优解;对大规模的 TSP,也能明显地改善解的质量。

参考文献:

[1] ORIGO M, GAMBARDELLA L M. Ant colony system: a cooperative learning approach to the traveling salesman problem[J]. IEEE Trans on Evolutionary Computation, 1997, 1(1): 53-66.

[2] TALBI H, DRAA A, BATOUCHE M. A new quantum-inspired genetic algorithm for solving the traveling salesman problem[C]//Proc of IEEE International Conference on Industrial Technology. 2004: 1192-1197.

[3] SONG Chi-hua, LEE K, LEE W D. Extended simulated annealing for augmented TSP and multi-salesmen TSP[C]//Proc of International Joint Conference on Neural Networks. 2003: 2340-2343.

[4] MICHEL G, GILBERT L, FREDERIC S. A tabu search heuristic for the undirected selective traveling salesman problem[J]. European J of Operational, 1998, 106(1): 539-545.

[5] YANG Hai-qing, YANG Hai-hong. An self-organizing neural network with convex-hull expanding property for TSP[C]//Proc of International Conference on Neural Networks and Brain. 2005: 379-383.

[6] 王文峰,刘光远,温万惠. 求解 TSP 问题的混合离散粒子群算法[J]. 西南大学学报:自然科学版, 2007, 29(1): 85-88.

[7] 黄雪梅,李涛,徐春林,等. 一种基于免疫遗传的 TSP 求解方法[J]. 四川大学学报:工程科学版, 2006, 38(1): 86-91.

[8] 孙力娟,王良俊,王汝传. 改进的蚁群算法及其在 TSP 中的应用研究[J]. 通信学报, 2004, 25(10): 111-116.

(上接第 1963 页)

[6] EI J, HAN J. PrefixSpan: mining sequential patterns efficiently by prefix-projected pattern growth[C]//Proc of the 7th International Conference on Data Engineering. Washington DC: IEEE Computer Society, 2001: 215-224.

[7] HAN J, PEI J, MORTAZVI-ASL B, *et al*. FreeSpan: frequent pattern-projected sequential pattern mining[C]//Proc of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York: ACM Press, 2000: 355-359.

[8] HAN J, PEI J, YAN X. From sequential pattern mining to structured pattern mining: a pattern-growth approach[J]. Journal of Computer Science and Technology, 2004, 19(3): 257-279.

[9] 张坤,朱杨勇. 无重复投影数据库扫描的序列模式挖掘算法[J]. 计算机研究与发展, 2007, 44(1): 126-132.

[10] LIN Ming-yen, LEE S Y. Fast discovery of sequential patterns by memory indexing[C]//Proc of the 4th International Conference on Data Warehousing and Knowledge Discovery. London, UK: Springer-Verlag, 2002: 150-160.

[11] GAROFALAKIS M N, RASTOGI R, SHIM K. Spirit: sequential pattern mining with regular expression constraints[C]//Proc of the 25th International Conference on Very Large Databases. San Francisco, CA: Morgan Kaufmann Publishers Inc, 1999: 223-234.

[12] PINTO H, HAN J, PEI J, *et al*. Multi-dimensional sequential pattern mining[C]//Proc of the 10th International Conference on Information and Knowledge Management. Atlanta, New York: ACM Press, 2001: 81-88.

[13] ZHANG Ming-hua, KAO B, CHEUNG D W, *et al*. Efficient algorithms for incremental update of frequent sequences[C]//Proc of the Pacific-Asia Conference on Knowledge Discovery and Data Mining. London, UK: Springer-Verlag, 2002: 186-197.

[14] PARTHASARATHY S, ZAKI M J, OGIHARA M, *et al*. Incremental and interactive sequence mining[C]//Proc of the 8th International Conference on Information and Knowledge Management. Kansas City, New York: ACM Press, 1999: 251-258.

[15] MASSEGLIA F, PONCELET P, TEISSEIRE M. Incremental mining of sequential patterns in large databases[J]. Data and Knowledge Engineering, 2003, 46(1): 97-121.

[16] ZHENG Qing-guo, XU Ke, MA Shi-ling, *et al*. The algorithms of updating sequential patterns[C]//Proc of the 5th International Workshop on High Perfor-mance Data Mining. Washington DC: [s. n.], 2002.

[17] CHENG Hong, YAN X, HAN J. IncSpan: incremental mining of sequential patterns in large database[C]//Proc of the 10th International Conference on Knowledge Discovery and Data Mining. New York: ACM Press, 2004: 527-532.

[18] 牛兴雯,杨冬青,唐世渭,等. OSAF2tree ——可迭代的移动序列模式挖掘及增量更新方法[J]. 计算机研究与发展, 2004, 41(10): 1761-1767.

[19] 邹翔,张巍,刘洋,等. 分布式序列模式发现算法的研究[J]. 软件学报, 2005, 16(7): 1262-1269.

[20] 吕静,王晓峰. 序列模式图及其构造算法[J]. 计算机学报, 2004, 27(6): 782-787.

[21] HAN J, DONG G, YIN Y. Efficient mining of partial periodic patterns in time series database[C]//Proc of the 15th International Conference on Data Engineering. Washington DC: IEEE Computer Society, 1999.

[22] YANG J, WANG Wei, YU P S. Mining asynchronous periodic patterns in time series data[C]//Proc of the 6th International Conference on Knowledge Discovery and Data Mining. New York: ACM Press, 2000: 275-279.

[23] ELFEKY M G. Incremental mining of partial periodic patterns in time-series databases[EB/OL]. (2000). http://citeseer.ist.psu.edu/421296.html.

[24] BETTINI C, WANG X S, JAJODIA S. Mining temporal relationships with multiple granularities in time sequences[J]. Data Engineering Bulletin, 1998, 21: 32-38.

[25] HARMS S K, DEOGUN J S. Sequential association rule mining with time lags[J]. Journal of Intelligent Information Systems, 2004, 22(1): 7-22.