一种并行 ACS-2-opt 算法处理 TSP 问题的方法

李 俊 童 钊 王 政
(湖南师范大学数学与计算机科学学院 长沙 410006)

摘 要 针对基本 ACS 算法模型求解 TSP 问题的缺陷,对 ACS 算法添加 2-opt 邻域搜索策略,增强算法对 TSP 问题解的构造能力,提高算法对 TSP 问题的求解精度。同时,根据 ACS 算法易于并行化的特点,使用并行化 ACS 算法与算法参数优化混合方案,提高 ACS 算法求解 TSP 问题的速度。最终实现了对中等规模 TSP 问题具有较好求解性能的并行 ACS-2-opt 算法。实验结果表明,2-opt 策略对于提升 ACS 算法的求解精度具有明显的效果;采用不同参数设定信息素启发因子时,求解时间具有较大差异;在采用节点距离倒数作为期望启发值时,ACS 算法模型呈现退化性;在并行条件下,ACS-2-opt 算法处理 TSP 问题时具有良好的并行性能。

关键词 2-opt 邻域搜索策略,ACS 算法,TSP 问题,并行计算
中图法分类号 TP301.6 **文献标识码** A

Approach to Solve TSP with Parallel ACS-2-opt

LI Jun TONG Zhao WANG Zheng
(College of Mathematics and Computer Science, Hunan Normal University, Changsha 410006, China)

Abstract According to defects in basic ACS solving TSP, this paper added 2-opt local search strategy to ACS model to improve the computing capacity and accuracy in the process of building the best tour. Moreover, since ACS algorithm is easy to be parallel processed, this paper used multithreaded concurrency and parameter optimization to enhance computing speed of basic ACS. Finally, this paper actualized parallel ACS-2-opt algorithm which has preferable performance in solving the medium TSP. According to the experimental results, 2-opt strategy has obvious effect on improving the accuracy of ACS solving TSP. The time cost of ACS solving TSP is distinct with different pheromone heuristic value settings. ACS becomes vestigial when using reciprocal of distance between two nodes as the corresponding heuristic value. Under the circumstance of parallel computation, ACS-2-opt has good parallel computing on solving TSP.

Keywords 2-opt local search strategy, ACS, TSP, Parallel computing

1 引言

旅行商问题(Traveling Salesman Problem, TSP)是一个易于描述但难以求解的路径搜索难题,其在现实生活中有诸多应用,如物流调度、路径导航、策略优化等[1-5]。TSP 问题可以简单描述为:在给定一个图 G 中,有 n 个顶点,且已知它们之间的距离关系,求从某一个顶点出发,遍历 n 个顶点的最短 Hamilton 环路的长度[6]。由于采用传统搜索算法,例如贪心算法求解 TSP 问题,具有高时间复杂度与收敛性差两大问题,现代计算机科学家先后提出了一系列具有 $O(n^2)$ 时间复杂度的启发式算法来处理 TSP 问题[7],如遗传算法[8](Genetic Algorithm, GA)、神经网络[9](Neural Network, NN)、粒子群算法[10](Particle Swarm Optimization, PSO)、模拟退火算法[11](Simulate Anneal, SA)等,均在处理 TSP 问题方面取得了良好的效果。

在诸多启发式算法中,Dorigo 于 1991 年提出的模仿蚂蚁觅食机制特性的蚁群优化算法[12](Ant Colony Optimization, ACO)对求解 TSP 问题具有较好的效果。因为该算法的设计原理与 TSP 问题的路径求解过程相似,近年来被广泛运用于求解 TSP 问题等路径规划问题中。ACO 算法从蚂蚁觅食的自然过程出发,参考蚂蚁释放生物信息素导航以确定食物与巢穴最短路径的机制,采用计算机模拟蚂蚁搜索路径时信息素散布与更新的方法,搜索最优路径,以求解路径规划问题。在 20 多年的发展历程中,Dorigo 等又先后提出精英蚂蚁[13](Elite-Ant, EA)、蚁群算法[14](Ants System, AS)、最大最小蚁群算法[15](Max-Min Ants System, MMAS)、蚁群系统算法[16](Ants Colony System, ACS)等一系列 ACO 算法的改进增强算法。

在这些 ACO 算法的改进模型中,ACS 算法具有局部信息素更新策略与更低运算时间两大新特性,对后续的 ACO 算法的设计产生了深远的影响。但是 ACS 算法同时也具有收敛性较差的特点,本文采用基础的 ACS 算法作为算法优化蓝本,针对基础 ACS 算法收敛性较差的问题,采用双元素交换优化的邻近搜索策略[17](2-opt 策略)对 ACS 求解周游序

本文受国家自然科学基金项目(61502165),湖南省教育厅一般项目(17C0959),湖南师范大学青年基金项目(11404),湖南师范大学大学生创新性实验项目(201501023)资助。

李 俊(1996-),男,主要研究方向为并行处理;童 钊(1985-),男,博士,讲师,CCF 会员,主要研究方向为云计算、资源优化,E-mail:tongzhao@hunnu.edu.cn(通信作者);王 政(1996-),男,主要研究方向为并行处理。

列进行路径交叉检测,去除蚂蚁求解最优路径中的路径交叉问题,从而提高算法的求解精度。针对 ACO 算法具有易并行化的特性,综合参数优化与算法并行化方案得到了并行 ACS-2-opt 算法,进一步提高了 ACS 算法的求解速度,通过对比实验得知,该算法在求解精度和求解速度上优于基础 ACS 算法。

2　TSP 问题的定义与 ACS 算法模型

2.1　TSP 问题的数学定义

在给定图 $G=(V,E)$中,V 为顶点集合,E 为顶点相互连接的边集合。设距离矩阵 $D=(d_{ij})$,d_{ij} 是顶点 i 和顶点 j 之间的距离。对于顶点集 $V=\{v_1,v_2,\cdots,v_n\}$的一个寻访序列 $T=\{t_1,t_2,\cdots,t_i,\cdots,t_n,t_{n+1}\}$,其中 $t_i\in V(i=1,2,3,\cdots,n+1)$,且 $t_{n+1}=t_1$,则 TSP 问题可描述为求

$$L_{\text{best}}=\min L=\sum_{i=1}^{n}d_{t_it_{i+1}} \tag{1}$$

的路径最优化问题。

TSP 问题大致可分为两类:1)对称 TSP 问题($d_{ij}=d_{ji}$,$\forall i,j=1,2,3,\cdots,n$);2)非对称 TSP 问题($d_{ij}\neq d_{ji}$,$\exists i,j=1,2,3,\cdots,n$)。由于非对称 TSP 问题在求解时较为困难,本文以下所讨论的 TSP 问题均指对称 TSP 问题。

2.2　ACS 算法模型

ACS 算法作为一种特殊的蚁群优化算法,具有以下 3 种区别于传统蚁群优化算法的特点:1)状态转移规则综合了新路径探索与先验知识应用;2)全局信息素更新规则仅应用于最优解中的边;3)局部信息素更新规则在路径构造中被采用。由于局部信息素更新策略的使用使得 ACS 算法中的蚂蚁对先前蚂蚁路径搜索过程产生的经验具有更强的学习能力,因而具有更好的路径构造性能。不同于其他 ACO 算法设置参数时,广泛推荐采用与所处理 TSP 问题节点数相等的蚂蚁数,ACS 算法中蚂蚁数的设定可采用固定数值,使得 ACS 算法相对于其他 ACO 算法,在处理同等规模问题的条件下,具有较低的运算时间开销。基本 ACS 算法可以描述为:算法初始化时,使 m 个蚂蚁分布(使用随机分布方法)到 n 个节点中的 m 个节点上($m\geqslant n$),每个蚂蚁通过运用状态转移方法构造路径解。在构造解的同时,全局信息素更新方法与局部信息素更新方法被运用来模拟天然蚂蚁路径搜索过程中信息素的释放-消散过程,逐步优化每个蚂蚁的路径选择,从而达到求解 TSP 问题最优路径的目的。这种通过参考信息素与先验路径构造经验进行最优解探索的过程,在机器学习领域被称作 Q-learning 学习[18],它是一种典型的强化学习(Reinforcement Learning)算法[19],具备收敛性[20-21],目前被广泛应用于人工智能研究领域。

ACS 算法的模型状态转移方法由以下公式推导得知:

$$s=\begin{cases}\arg\max\limits_{u\in J_k(r)}\{[\tau(r,u)]^{\alpha}\cdot[\eta(r,u)]^{\beta}\}, & \text{if } q\leqslant q_0\\ \arg\max\limits_{u\in J_k(r)}S, & \text{otherwise}\end{cases} \tag{2}$$

其中,r 为第 k 个蚂蚁当前所在节点位置;s 为第 k 个蚂蚁即将移动到的节点;$J_k(r)$为由 r 出发的蚂蚁可移动的节点集合;u 为 $J_k(r)$中的节点;$\tau(r,u)$为 r 到 u 节点边上的信息素;$\eta(r,u)$为 r 到 u 节点边上的期望启发值,实验过程中采用 r 到 u 节点间距离的倒数 L_{r-u}^{-1}作为启发值,L_{r-u}^{-1}又可看作是 r 与 u 节点间的能见度。α 为信息启发因子,取值范围设定为(0,1);β 为期望启发因子,可取大于 0 的整数;q 为[0,1]区间中的随机数,q_0 为取值范围为(0,1)的参数,该值控制蚂蚁的路径选择过程。

式(2)中 S 由 AS 算法[14]的状态转移方法导出:

$$S=\frac{[\tau(r,u)]^{\alpha}\cdot[\eta(r,u)]^{\beta}}{sum} \tag{3}$$

$$sum=\sum_{w\in J_k(r)}[\tau(r,w)]^{\alpha}\cdot[\eta(r,w)]^{\beta} \tag{4}$$

ACS 算法的全局信息素更新规则为:

$$\tau(r,s)\leftarrow(1-\rho)\cdot\tau(r,s)+\rho\cdot\Delta\tau(r,s) \tag{5}$$

其中,

$$\Delta\tau(r,s)=\begin{cases}(L_{\text{global-best}})^{-1}, & \text{if } (r,s)\in global\text{-}best\text{-}tour\\ 0, & \text{otherwise}\end{cases} \tag{6}$$

其中,$\tau(r,s)$为 r 到 s 节点边上的信息素值。ρ 为全局信息素挥发参数,取值范围设定为(0,1),控制全局路径上的信息素消散。$L_{\text{global-best}}$为全局最优路径长度。

ACS 算法的局部信息素更新规则为:

$$\tau(r,s)\leftarrow(1-\varepsilon)\cdot\tau(r,s)+\varepsilon\cdot\Delta\tau(r,s) \tag{7}$$

其中,$\Delta\tau(r,s)=\tau_0=(n\cdot L_m)^{-1}$,$L_m$ 为由临近最短方法计算所得的路径长度,n 为 TSP 问题总节点数。ε 为局部信息素挥发参数。取值范围设定为(0,1),其控制局部信息素的消散。

ACS 算法的伪代码如算法 1 所示。

算法 1　ACS 算法

输入:TSP 问题坐标,Nc,m,ε,ρ,q_0

输出:给定 TSP 问题最优路径长度

//对每条边初始化路径信息素与期望启发值

1. For each edge (r,s)

//由定义初始化 τ_0

2. $\tau(r,s)=\tau_0$;

3. $\eta(r,s)=L_{r-s}^{-1}$;

4. End-For

5. For k=1 to m

//初始化第 k 个蚂蚁的位置,通过随机数放置蚂蚁

6. Set r_{k1} be the starting city for ant k;

//$J_k(r_{k1})$为第 k 个蚂蚁从 r_{k1}出发可移动到的节点集合

7. $J_k(r_{k1})=\{1,\cdots,n\}-r_{k1}$;

//r_{k1}是第 k 个蚂蚁现在所在的位置

8. /$r_k=r_{k1}$;

9. End-For

// 构建最优路径与局部信息素更新

10. For i=1 to n

11. If i<n

12. For k=1 to m

//蚂蚁通过状态转移法则选择下一个节点

13. Choose the next city s_k according to state transition rule (2);

14. $J_k(s_k)=J_k(r_k)-s_k$;

//$Tour_k$ 存储第 k 个蚂蚁所经过的路径

15. $Tour_k(i)=(r_k,s_k)$;

16. End-For

17. Else

//生成环路

18. For k=1 to m
19. $s_k = r_k$;
20. $Tour_k(i)=(r_k,s_k)$;
21. End-For
22. End-If
//局部信息素更新
23. For k=1 to m
24. $\tau(r_k,s_k)=(1-\varepsilon)\cdot\tau(r_k,s_k)+\varepsilon\cdot\tau_0$;
25. $s_k=r_k$;
26. End-For
27. End-For
//全局路径更新与全局信息素更新
28. For k=1 to m
//L_k 是由第 k 个蚂蚁所构建的路径长度
29. Calculate L_k;
30. End-For
31. Calculate L_{best};
//全局信息素更新
32. For each edge (r,s)
33. $\tau(r_k,s_k)=(1-\rho)\cdot\tau(r_k,s_k)+\rho\cdot L_{best}^{-1}$;
34. End-For
35. If (End_Condition=True)
//满足终止条件时,输出最优路径
36. Print the shortest of L_k;
37. Else
//重复路径构造过程,直到满足终止条件
38. Goto 10;
39. End-If

3 ACS 算法优化方案

3.1 2-opt 邻域搜索策略

由于基础 ACS 算法中没有检测消除蚂蚁搜索路径交叉(如图 1 所示)的邻域搜索策略,因此在求解相对较大规模($n>30$)的 TSP 问题时,路径出现交叉情况的概率增加,导致一定迭代数下的求解精度较差。本文通过添加 2-opt 邻域搜索策略,消除 ACS 算法构造路径中的交叉情况,提升算法对于较大规模 TSP 问题的求解精度。

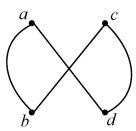

图 1 路径交叉示意图

2-opt 邻域搜索策略可描述为:当 ACS 算法求解完成之后,搜索寻访序列 T,若 T 中存在如图 1 所示的路径交叉情况,则置换交叉位置所对应的路径节点,重新连接 a 与 c、b 与 d(如图 2 所示),并同时更新寻访序列 T 与最短周游距离 L_{best}。重复以上步骤,多次检测寻访序列 T,直到最短周游距离 L_{best} 无法取得更小值。

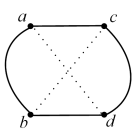

图 2 交叉路径节点置换示意图

2-opt 邻域搜索策略的伪代码如算法 2 所示。

算法 2 2-opt 邻域搜索算法

输入:ACS 算法搜索所得最优路径

输出:通过 2-opt 消除路径交叉后新的最优路径

//读取 ACS 算法所得最优路径
1. Initialize
2. Do{
//搜索寻访路径 T 探测交叉
3. Search T;
//依据路径长度进行检测
4. If ($L_{a-c}+L_{b-d}<L_{a-d}+L_{b-c}$)
//重新连接路径
5. Link a to c;
6. Link b to d;
//使用 L_{raw} 记录原始最优路径解
7. $L_{raw}=L_{best}$;
//更新寻访路径与最优路径
8. Renovate T;
9. Renovate L_{best};
10. End-If
//再次探测无法取得更优的路径时退出搜索
11. }While($L_{best}<L_{raw}$)
12. End

3.2 α,β 参数优化与算法并行化

根据 ACS 模型的导出公式,α,β 对于单个蚂蚁而言,在路径选择过程中具有较大的控制作用,对 ACS 算法求解最优路径的过程有较大的影响。较好的 α,β 值能使得在较差路径上的信息素、期望启发值大幅减少,降低较差路径的选择概率,从而缩短单个蚂蚁在较差路径上的停留时间,降低 ACS 算法对于 TSP 问题整体求解的时间开销。本文采取对 α,β 在可取值范围内进行调节的方式,简单优化 α,β,取得了一定效果。

同时,由于蚁群算法在生成路径时,每个蚂蚁的路径生成过程较为独立,从算法 1 的描述中又可以看出整个 ACS 算法在生成最优解过程中采用了较多的 FOR 循环,因此蚁群算法具有较容易实现并行计算的优点。通过采用具有更快 FOR 语句执行速度的 C 语言编码,结合 OpenMP 并行编程方法,实验中实现了 ACS-2-opt 算法共享内存式多线程并发求解 TSP 问题。在多核 CPU 多线程计算 TSP 问题的实验中,该算法取得了较为明显的并行计算加速效果。

4 算法实验结果分析

4.1 串行 ACS 算法添加 2-opt 策略的实验结果分析

添加 2-opt 策略前后,ACS 算法串行求解 TSP 问题的实验结果如表 1 所列。

表 1 添加 2-opt 策略 ACS 串行求解 TSP 问题的比较实验

问题名	已知最优解	ACS 最优解	偏离量/%	ACS-2-opt 最优解	偏离量%
Oilver 30	423.76	449.186	6.00	424.462	0.17
Elion 50	428.47	445.532	3.98	429.48	0.24
Elion 75	542.31	567.393	4.63	546.162	0.71
KroA 100	21282	24135	13.40	21920	2.99

实验中设定算法的基本参数为:迭代数 $Nc=1000$,蚂蚁数 $m=30$,$\alpha=0.5$,$\beta=1$,$\rho=0.9$,$\varepsilon=0.1$,$q_0=0.9$,CPU 为

i5-6300HQ 4 核心处理器主频率 2.30GHz,睿频功能关闭,对每个 TSP 问题重复计算 10 次,然后取其最优值。已知最优解数据来自于 TSPLIB 数据库[22]。偏离量由式(8)计算得到。

$$偏离量=\frac{算法最优解-已知最优解}{已知最优解}\times 100\% \tag{8}$$

从表 1 的数据可以得知,在添加了 2-opt 策略后,由于 2-opt 策略解除了将原有求解路径中的路径交叉情况,因此相较于基本 ACS 算法,求 TSP 问题最优解的精度有了明显的提高。其中对 30,50,75 点规模的 TSP 问题都能达到低于 1%偏离量的求解精度,同时考虑到算法在实现时采用的数据类型与 TSPLIB 中算法求解使用的数据类型之间不同造成的精度误差,ACS-2-opt 算法与基础 ACS 算法相比,在设定同等迭代次数、处理同等规模问题的条件下,具有更好的求解精度。在求解 KroA 100 问题时,由于问题规模扩大,而迭代次数 Nc 与蚂蚁数 m 仍保持不变,相对于求解的 TSP 问题的规模路径构造次数不足、蚂蚁数偏少,导致蚂蚁可学习的路径构造经验不足,求解精度出现了大幅下降态势。

4.2　串行 ACS-2-opt 算法 α,β 参数优化实验结果分析

串行 ACS-2-opt 处理 Elion 50 调参数优化实验的参数设定为:迭代数 $Nc=1000$,蚂蚁数 $m=25$,$\rho=0.9$,$\varepsilon=0.1$,$q_0=0.9$,实验所用 CPU 为 i5-6300HQ 4 核心处理器主频率 2.30 GHz,睿频功能关闭。设定 β 为 1,在[0.1,1]区间内以 0.1 为进位调节参数 α。实验结果如图 3 所示。

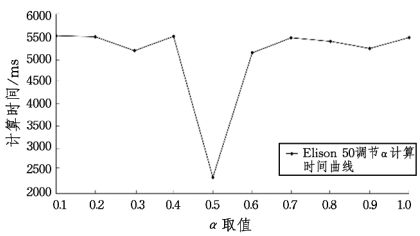

图 3　Elion 50 调节 α 计算时间曲线

从图 3 中可以得知,当 $\alpha=0.5$ 时,算法的计算速度明显加快,所用时间为 2361.4ms,相比于其他 α 取值的平均计算时间 5367.42ms 缩短了 43%,即缩短近一半计算时间。同时注意到在除 0.5 取值位点上,计算时间也有小幅度的波动,这是由于 ACS-2-opt 初始化时,蚂蚁分布节点的位置有所不同,单个蚂蚁在路径探索上的时间有差别。由图 3 中数据可以认为在除 $\alpha=0.5$ 以外的 α 取值上,ACS-2-opt 算法对同等规模的 TSP 问题具有相近的计算时间开销。而当 $\alpha=0.5$ 时由于信息素消散达到一个比较良好的状态,使得单个蚂蚁在较差路径上停留的时间缩短,进而算法的计算速度得到提升。

设定 α 为 0.1,其他参数的设定与设置 α 的实验相同。在[1,10]区间内以 1 为进位调节参数 β。实验结果如图 4 所示。

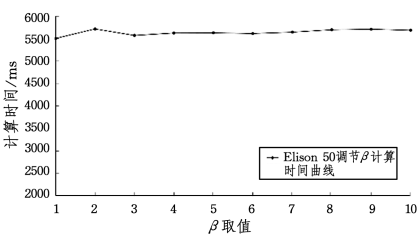

图 4　Elion 50 调节 β 计算时间曲线

与调节 α 参数实验相比,调节参数 β 后,在不同的 β 取值上 ACS-2-opt 对于同等规模问题在相等的迭代次数下,所用的计算时间变化较小,计算时间曲线近似一条直线,仅因初始化蚂蚁位置随机性导致小幅度计算时间波动。由此可以得知,在本文实验所采用的期望启发值取节点间能见度的意义下,在实验给定范围内调节 β 对 ACS-2-opt 算法的求解时间并无明显影响。在实验条件下,可以始终设定 $\beta=1$,使得 ACS-2-opt 算法退化为有信息素启发因子 α、全局信息素挥发参数 ρ、局部信息素挥发参数 ε 与控制参数 q_0 的四参数模型,减少了 ACS 算法所用的参数数目。

4.3　ACS-2-opt 串并行计算的实验结果分析

ACS-2-opt 算法串行与并行求解 TSP 问题的实验数据如表 2 所列。实验中设定算法的基本参为 $Nc=1000$,$m=30$,根据调参实验重新设定 $\alpha=0.5$,$\rho=1$,$\rho=0.9$,$\varepsilon=0.1$,$q_0=0.9$,CPU 为 i5-6300HQ 4 核心处理器主频率 2.30GHz,睿频功能关闭,对每个 TSP 问题重复计算 10 次,取其最优值,已知最优解数据来自于 TSPLIB 数据库[22]。其中 KroA 100、KroA 150、KroA 200 采用取整计算,实验采用 OpenMP 实现算法并行。加速比由式(9)计算得到:

$$加速比=\frac{单线程平均计算时间}{n\ 个线程平均计算时间} \tag{9}$$

表 2　ACS-2-opt 串并行求解 TSP 问题的实验结果

问题名	已知最优解	单线程最优解	平均计算时间/ms	四线程最优解	平均计算时间/ms	加速比
Elion 75	542.31	546.162	5997.61	551.233	1834.78	3.27
KroA 100	21282	21920	10892.92	21911	3101.79	3.51
KroA 150	26524	27335	25107.47	27069	6828.38	3.68
KroA 200	29368	31837	45206.56	31874	12753.62	3.54

通过表 2 中的数据可以得知,ACS-2-opt 算法在并行计算与串行计算下分别求解 75,100,150,200 点问题时的最优解精度相近,但并行计算时间相比于串行计算时间大幅降低。串并行计算 TSP 实验中,求解 4 个 TSP 问题在四核心四线程处理器上的并行加速比都超过了 3.00,说明 ACS-2-opt 算法具有良好的并行计算加速性能,这也再一次验证了 ACO 算法系列适合于并行处理。另一方面,从表 2 可知,加速比出现了先增加后降低的性态表现。在计算问题规模达到 150 点时,加速比达到 3.68 的最高值,而后在 200 点规模问题上又出现下降,这种情况可能是进程间通讯时间或算法中串行计算部分时间随 TSP 问题规模的扩大所致,有待在未来的研究中进一步进行验证。

结束语　通过对基本 ACS 模型添加 2-opt 邻域搜索策略与并行化,本文对于中小规模($n<200$)TSP 问题实现了相较基本 ACS 算法更快的求解速度与更好的求解精度。但是对于更大规模的 TSP 问题,例如 200 点的 TSP 问题,由于实验设定的迭代次数较低、蚂蚁数较少,蚂蚁学习过程相对问题规模不足,出现求解精度大幅下降的态势。由此,ACS 算法在大规模 TSP 问题的求解上,仍具有较大的可优化空间。一方面可对邻域搜索算法进行改进,例如引入 3-opt 策略(三元素交换优化选择策略)或具有更高优化水平的 Lin-Kernighan-Helsgaun 算法,以进一步提高 ACS 算法求解 TSP 问题的求解精度与求解规模;另一方面可通过使用具有更强运算速度的硬件设备和更加高效的并行化编码优化策略,来大幅度提

升 ACS 算法求解 TSP 问题的效率。这两种方案是今后对 ACS 算法优化研究的主要方向。同时在实验过程中，我们注意到迭代次数、启发因子等主要参数的设定对于 ACS 算法的求解精度有着较大的影响。在较低的迭代次数下，例如迭代次数仅设定为一次，算法所得路径具有大量交叉，且远远偏离最优路径，仅利用 2-opt 策略是无法较好处理这种情况的，但迭代数过高又会导致求解时间过长。采用混合诸如 PSO 算法、GA 算法的启发式算法，可优化参数设定、降低算法迭代次数，这也是 ACS 算法与 ACO 系列算法理组合优化问题的研究方向之一。

参 考 文 献

[1] BEKTAS T. The multiple traveling salesman problem: an overview of formulations and solution procedures [J]. Omega, 2006, 34(3): 209-219.

[2] DHAMMPAL R, SANJAY K, PATLE V K. Route optimisation by ant colony optimisationtechnique [J]. Procedia Computer Science, 2016(92): 48-55.

[3] ZHANG Z, YAO Y S, ZHANG J H. Algorithm evolution from traveling salesman problem to vehicle routing problem [J]. Applied Mechanics and Materials, 2013, 411: 1872-1875.

[4] SAVURAN H, KARAKAYA M. Efficient route planning for an unmanned air vehicle deployed on a moving carrier [J]. Soft Computing, 2016, 20(7): 2905-2920.

[5] GIBSON B, WILKINSON M, KELLY D. Let the pigeon drive the bus: pigeons can plan future routes in a room [J]. Animal Cognition, 2012, 15(3): 379-391.

[6] 严晨，王直杰. 以 TSP 为代表的组合优化问题研究现状与展望 [J]. 计算机仿真，2007，24(6)：171-174.

[7] HOLLAND J H. Adaption in natural and artificial system[J]. Ann Arbor, 1975, 6(2): 126-137.

[8] SHABANPOUR M, YADOLLHI M, HASANI M M. A new method to solve the multi traveling salesman problem with the combination of genetic algorithm and clustering technique[J]. International Journal of Computer Science and Network Security, 2017, 17(5): 221-230.

[9] AVSAR B, ALIABADI D E. Parallelized neural network system for solvinge uclidean traveling salesman problem [J]. Applied Soft Computing, 2015, 34(1): 862-873.

[10] BALI O, ELLOUMI W, ABRAHAM A, et al. GPU PSO and ACO applied to TSP for vehicle security tracking [J]. Journal of Information Assurance and Security, 2016, 11(6): 369-384.

[11] EZUGWU A E S, ADEWUMI A O, FRINCU M E. Simulated annealing based symbiotic organisms search optimization algorithm for traveling salesman problem [J]. Expert Systems with Applications, 2017, 77: 189-210.

[12] DORIGO M. Ant Colony Optimization [M]. Cambridge: MIT Press, 2004.

[13] COLORNI A, DORIGO M, MAFFIOLI F. Heuristics from nature for hard combinatorial optimization problems [J]. International Transactions in Operational Research, 1996, 3(1): 1-21.

[14] DORIGO 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.

[15] BAI J, YANG G K, CHEN Y W, et al. A model induced max-min ant colony optimization for asymmetric traveling salesman problem [J]. Applied Soft Computing, 2013, 13(3): 1365-1375.

[16] DORIGO M, MANIEZZO V, COLORNI A. The ant system: optimization by a colony of cooperating agents [J]. IEEE Transactions on Systems, Man, and Cybernettics-Part B, 1996, 26(1): 1-13.

[17] CROES G A. A menthod for solving traveling salesman problems [J]. Operations Research, 1958, 6(6): 791-812.

[18] TONG Z, XIAO Z, LI K, et al. Proactive scheduling in distributed computing-A reinforcement learning approach[J]. Journal of Parallel and Distributed Computing, 2014, 74(7): 2662-2672.

[19] AlEXANDER L S, LI L H, ERIC W, et al. Pac model-free reinforcement learning [C]// Proceeding of International Conference on Machine Learning. 2006: 881-888.

[20] CARLOS R, CSABA S. Q-learning combined with spreading: convergence and results [C]// Proceedings of the ISRF-IEE International Conference: Intelligent and Cognitive Systems (Neural Networks Symposium). 1996: 32-36.

[21] 黄瀚，郝志峰，吴国春，等. 蚁群算法的收敛速度分析 [J]. 计算机学报，2007，30(8)：1344-1353.

[22] REINELT G. TSPLIB-A traveling salesman problem library [J]. ORSA Journal on Computing, 1991, 3(4): 376-384.

(上接第 112 页)

[9] 王珏，王任，苗夺谦，等. 基于 Rough Set 理论的数据浓缩[J]. 计算机学报，1998(5)：393-400.

[10] 苗夺谦，胡桂荣. 知识约简的一种启发式算法[J]. 计算机研究与发展，1999(6)：681-684.

[11] 贾平，代建华，潘云鹤，等. 一种基于互信息增益率的新属性约简算法[J]. 浙江大学学报，2006(6)：1041-1045.

[12] LI M, SHANG C X, FENG S Z, et al. Quick attribute reduction in inconsistent decision tables[J]. Information Sciences, 2014, 254: 155-180.

[13] CHEN D, WANG C, HU Q. New approach to attribute reduction of consistent and inconsistent covering decision systems with covering rough sets [J]. Information Sciences, 2007, 177(17): 3500-3518.

[14] ZHAO Y, YAO Y Y, LUO F. Data analysis based on discernibility andindiscernibility [J]. Information Sciences, 2007, 177(22): 4959-4976.

[15] 张小红，裴道武，代建华. 模糊数学与 rough 集理论[M]. 北京：清华大学出版社，2013：264-265.