文章编号: 1001-0920 (2014) 10-1729-08 **DOI:** 10.13195/j.kzyjc.2013.1173

一种面向对象的多角色蚁群算法及其TSP问题求解

杜鹏桢, 唐振民, 孙 研

(南京理工大学 计算机科学与工程学院，南京 210094)

摘 要: 蚁群算法的改进大多从算法本身入手或与其他算法相结合, 未充分利用待解决问题所包含的信息, 提升效果较为有限. 对此, 提出一种面向对象的多角色蚁群算法. 该算法充分利用旅行商问题 (TSP) 对象的空间信息, 采用 k-均值聚类将城市划分为不同类别; 同时, 对蚁群进行角色划分, 不同角色的蚁群针对城市类别关系执行各自不同的搜索策略, 增强了蚁群的搜索能力, 较大幅度地提高了求解质量. 每进行一次迭代, 仅各角色最优个体进行信息素更新, 防止算法退化为随机的贪婪搜索. 将精英策略与跳出局部最优相结合可避免算法的停滞. 50 个经典 TSP 实例仿真实验表明: 所提出的算法可以在较少的迭代次数内获得或非常接近于问题的已知最优解; 对于大规模 TSP 问题所得结果也远超所对比的算法.

关键词: 蚁群算法；面向对象；多角色；k-均值；旅行商问题；2-Opt

中图分类号: TP18 **文献标志码:** A

An object-oriented multi-role ant colony optimization algorithm for solving TSP problem

DU Peng-zhen, *TANG Zhen-min*, *SUN Yan*

(College of Computer Science and Engineering，Nanjing University of Science and Technology，Nanjing 210094, China. Correspondent：DU Peng-zhen，E-mail：h.k@foxmail.com)

Abstract: Most of the improvements on the ant colony algorithm are based on the algorithm itself or combined with other algorithms, which underutilizes information of the problems to be solved, so the effect is not ideal. An object-oriented multi-role ant colony optimization(OMACO) algorithm is proposed to deal with the characteristics of traveling salesman problem(TSP) on the basis of k-means, which divides the cities into different categories with full use of the TSP spatial information and divides the colony into different roles. Each role solves the problem independently according to their respective strategies, which enhances the algorithm's search capability and improves the quality of solution. After each iteration, only the optimal solution of each role is applied to update the pheromone, which prevents premature convergence. The elitist strategy combined with local optima jumping is used again to the stagnation of the algorithm. Many experiments of 50 classic TSP instances show that the OMACO algorithm can obtain the known optimal solution in fewer iterations. For large-scale TSP instances, the proposed algorithm is also far better than the comparison algorithms.

Key words: ant colony optimization；object-oriented；multi-role；k-means；traveling salesman problem；2-Opt

0 引 言

旅行商问题 (TSP) 是经典的组合优化问题, 具有很强的现实意义, 已广泛地应用于计算机网络、物流配送、车辆路径、大规模集成电路布线等领域. 同时, TSP 又是 NP-Hard 问题, 随着问题规模的增大, 解空间迅速膨胀, 目前并未得到完全解决. TSP 求解算法分为精确型和启发型两种: 对于小型 TSP 问题, 采用传统的精确型算法处理可以从理论上获得最优解; 对于中大型 TSP 问题, 精确型算法因时间和空间消耗过于巨大而完全不可行, 一般采用启发式算法进行求解. 启发式算法不能保证获得最优解, 但可以在有限的时间里获得令人满意的解, 常见的启发式算法有贪婪算法、k-Opt、蚁群算法 (ACA)[1]、粒子群优化算法 (PSO) 等. 蚁群算法是一种模拟自然界蚁群觅食行为的智能优化算法, 最早由意大利学者 Dorigo 等[2]提出. 因其具有收敛速度快、鲁棒性较强、采用分布式

收稿日期: 2013-08-27；**修回日期:** 2013-12-29.

基金项目: 国家自然科学基金项目(91220301, 61371040)；高等学校学科创新引智计划课题(B13022).

作者简介: 杜鹏桢(1982−), 男, 博士生, 从事机器人及智能计算的研究；唐振民(1961−), 男, 教授, 博士生导师, 从事模式识别与智能系统等研究.

并行计算结构、易与其他算法结合等优点, 目前已成功地应用于TSP问题、车辆路由问题、作业车间调度等诸多方面. 但是, 蚁群算法在解决大规模优化问题时存在求解质量低、收敛速度慢、易停滞等缺点.

针对以上问题, 人们提出了很多改进算法. 文献[2]改进了选择方式, 将信息素的更新分为局部和全局两种, 达到了求解质量与收敛速度之间的平衡. 文献[3]通过设置信息素阈值, 避免了算法的早熟, 提高了求解质量, 但迭代次数增加, 收敛速度有所下降. 文献[4]对所得解进行排序, 只对最优的ω个路径进行信息素更新, 提高了收敛速度, 但在某些情况下更容易陷入局部最优. 文献[5-6]分别以不同的方式加强信息素的对比度, 加快了收敛速度, 提高了求解质量, 但对于中大规模优化问题效果不理想. 文献[7-8]分别对信息素和最优解进行变异, 提高了算法的搜索能力, 但没有对算法易陷于局部最优的情况进行考虑. 文献[9-10]采用不同的并行技术对算法进行加速, 但对算法本身没有太大改进. 文献[11-12]利用PSO对蚁群算法的参数选取进行优化, 将蚁群算法作为PSO的适应度函数, 对求解质量提高有限, 而且算法时间复杂度上升较大.

上述算法均从ACA本身着手进行改进或与其他算法相结合, 在一定程度上割裂了算法与求解对象的关系, 并未充分利用求解对象所包含的信息, 间接影响了算法的有效性. 鉴于此, 本文针对TSP问题, 提出一种面向对象的多角色蚁群算法(OMACO). OMACO首先对TSP的城市进行分类, 并对蚁群进行角色划分; 然后, 不同角色的蚁群通过引入的赏罚因子对不同的城市类别关系执行不同的搜索策略; 最后, 为了进一步提高求解质量, 采用一种简化的2-Opt算法, 在较低的时间复杂度下对最优解进行局部优化. 需要注意的是, 本文算法的这种行为与现有增强信息素对比度的算法[5-6]表面上较为类似, 但本质并不相同. 现有算法较少考虑对象信息, 仅从算法本身进行改进, 而本文的改进方法是在城市分类的基础上进行的, 充分利用了对象所包含的信息, 这也是本文创新点之一. 本文这种理念并不仅限于TSP问题, 对于所有可分类对象都适用, 可作为一种提高蚁群算法搜索能力的普适方法. 对50个TSP经典实例进行的仿真实验表明, OMACO可以在较少的迭代次数内获得或非常接近于问题的已知最优解, 求解精度较所对比的算法提高了76%.

1 基本蚁群算法

利用蚁群算法对TSP问题的求解, 实际上是模仿若干蚂蚁并行搜索路径的过程. 当蚂蚁完成一次路径的搜索时, 其走过的路径即为TSP问题的可行解. 算法开始时, 把m只蚂蚁随机放入n个城市, 则t时刻第k只蚂蚁按下式所得概率p_{ij}^k采用轮盘赌策略来决定下一个城市:

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

其中: $\tau_{iu}(t)$为t时刻城市i到城市u之间的信息素数量, α为信息素的相对重要程度, $\eta_{iu}=1/d_{iu}$为城市i到城市u的启发式因子(能见度), d_{iu}为城市i与城市u的距离, β为启发式因子的重要程度, allowed_k为第k只蚂蚁未遍历过的城市集合.

在搜索过程中, 蚂蚁会释放信息素使整个系统的正反馈增强, 有利于算法更快地收敛. 蚁周模型更偏重于全局信息, 使用较为广泛, 其更新规则为

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

其中

$$\Delta\tau_{ij}(t)=\sum\Delta\tau_{ij}^k(t);\tag{3}$$

$$\Delta\tau_{ij}^k(t)=\begin{cases}Q/L_k, & (i,j)\in\text{ tour by ant }k;\\ 0, & \text{otherwise;}\end{cases}\tag{4}$$

ρ为挥发系数, 该系数增大可使算法随机性加强, 收敛速度降低, 减小则加速算法收敛, 但求解质量下降; Q为信息素总量; L_k为第k只蚂蚁本次迭代所走过的路径长度.

2 面向对象的多角色蚁群算法

相对于基本ACA, OMACO主要有5点改进: 1) 充分考虑解空间所包含信息, 使用k-means进行城市分类; 2) 蚁群分为普通蚁群、开拓蚁群和寻优蚁群, 制定不同的搜索策略, 并在路径选择方式上引入赏罚因子; 3) 仅对各角色最优路径进行信息素的更新; 4) 每完成一次迭代, 对最短路径执行简化的2-Opt操作, 进行局部寻优; 5) 执行精英策略与跳出局部最优相结合的方式.

2.1 TSP城市的分类

现有算法对TSP城市无差别对待, 忽略了TSP问题本身所包含的空间信息. 本算法采用k-means对城市进行分类, 同时引入置信区间, 从已分类城市中分离出无类城市, 为多角色的蚁群执行不同的搜索策略提供前提条件.

2.1.1 k-means聚类及类别数目的自适应

k-means聚类是一种非常经典的聚类算法, 简洁、高效、便于理解, 现实中运用广泛. 针对TSP问题, 其基本流程如下.

Step 1: 从城市集合$\text{cities}=\{c_1,c_2,\cdots,c_n\}$中随

机选取 k 个城市作为类中心, 构建集合 $\text{center} = \{s_1, s_2, \cdots, s_k\}$.

Step 2: $\forall c_i \in \text{cities}$ 按下式划入类 j:

$$j = \text{argmin}(d_{iu}),\ u = 1, 2, \cdots, k; \tag{5}$$

$$d_{iu} = \| c_i - s_u \|,\ s_u \in \text{center}. \tag{6}$$

其中 $\| \cdot \|$ 表示欧氏距离.

Step 3: 按下式更新类中心:

$$s_j(t+1) = \frac{\sum\limits_{c_i \in \text{class}_j(t)} c_i}{n_j(t)},\ s_j \in \text{center}. \tag{7}$$

其中: $\text{class}_j(t)$ 为 t 时刻类 j 的城市集合, $n_j(t)$ 为 $\text{class}_j(t)$ 中城市数目.

Step 4: 若 center 被更新, 则转 Step 2, Step 3, 否则算法完成.

城市类别数目过大或过小都会导致种群多样性下降, 最终影响算法的求解质量. 经大量测试, 类别数目应依下式确定:

$$k = \begin{cases} \lfloor n/25 \rfloor,\ n \geqslant 125; \\ 4,\ \text{otherwise}. \end{cases} \tag{8}$$

其中: k 为待定城市类别数目, n 为 TSP 城市数目.

2.1.2　分离无类城市

蚁群在选择下一城市的过程中, 算法会根据城市类别关系进行鼓励或惩罚. 通过引入置信区间, 分离无类城市, 可以缓冲城市类别间的壁垒, 进一步提高算法的多样性.

对于 n 个城市的 TSP 问题, 每个城市到各自类别中心的距离构建集合 $\text{dist} = \{d_1, d_2, \cdots, d_n\}$. 于是满足下式的城市将被分离为无类城市:

$$d_i - \mu \geqslant \varepsilon\sigma,\ i = 1, 2, \cdots, n. \tag{9}$$

其中: μ 为 dist 均值; σ 为标准差; $\varepsilon \in [1, 2]$ 为分离因子, 取值过小, 无类城市数目增加, 取值过大, 无类城市数目减少.

2.2　蚁群角色的划分及相应策略

OMACO 把蚁群分为 3 种角色: 1) 普通蚁群; 2) 开拓蚁群; 3) 寻优蚁群. 一般开拓蚁群和寻优蚁群数量均为普通蚁群的一半. 算法开始时, 3 种角色以不同的策略同时进行路径搜索, 其差别具体反映在城市选择策略的不同.

蚁群的城市选择策略以城市类别关系为基础. 为了表示城市类别之间的差异, 首先引入一个类别算子

$$\text{sgn}(i, j) = \begin{cases} 1,\ i \neq j,\ i \in \text{class}_k,\ j \in \text{class}_k; \\ -1,\ i \in \text{class}_l,\ j \in \text{class}_k,\ l \neq k; \\ 0,\ \text{otherwise}. \end{cases} \tag{10}$$

其中: i 和 j 均为城市编号, class_l 表示类别为 l 的城市集合, class_k 表示类别为 k 的城市集合. 当 $\text{sgn}(i, j) = 1$ 时, 表示城市 i 与城市 j 同在一个类别; 当 $\text{sgn}(i, j) = -1$ 时, 表示城市 i 与城市 j 分属不同类别; 当 $\text{sgn}(i, j) = 0$ 时, 表示城市 i 与城市 j 至少有一个为无类城市.

针对不同的类别关系, 本文提出 3 种搜索策略, 具体如下.

策略 1　兼顾类内、类间、类与无类之间的搜索, 以期达到全局最优和收敛速度的平衡.

策略 2　强调类间城市的搜索, 注重全局最优.

策略 3　强调同类城市的搜索, 注重局部路径的调优, 加快收敛速度.

OMACO 中采用下式来确定转移到下一个城市的概率:

$$p_{ij}^k = \begin{cases} \dfrac{\tau_{ij}(t)\eta_{ij}^3 \xi^{\gamma \text{sgn}(i,j)}}{\sum\limits_{u \in \text{allowed}_k} \tau_{iu}(t)\eta_{iu}^3 \xi^{\gamma \text{sgn}(i,u)}},\ j \in \text{allowed}_k; \\ 0,\ \text{otherwise}. \end{cases} \tag{11}$$

其中: $\xi \in [1.5, 8]$ 为赏罚因子, 表示蚁群对类别关系的喜好程度; $\gamma \in \{-1, 0, 1\}$ 决定策略的选择.

普通蚁群采用策略 1, 此时 $\gamma = 0$, 对各种城市类别关系无差别对待.

开拓蚁群采用策略 2, 此时 $\gamma = -1$, 放大类间城市的选择概率, 缩小类内城市的选择概率.

寻优蚁群采用策略 3, 此时 $\gamma = 1$, 放大类内城市的选择概率, 缩小类间城市的选择概率.

不同的搜索策略直接影响最终路径的形成. 图 1 为所有蚂蚁从同一城市出发, 某一次迭代经过 9 次搜索后, 各角色所获得的最优路径片断.

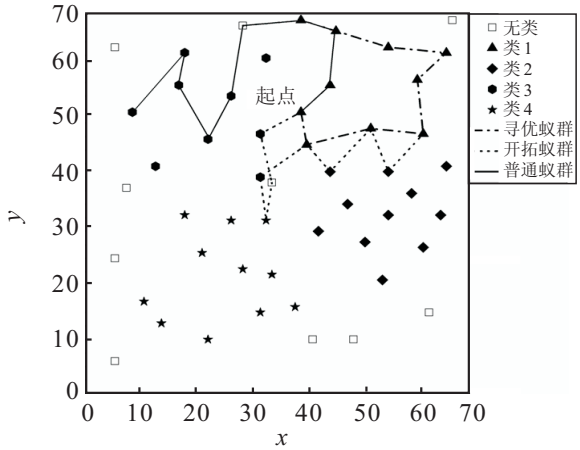


图 1　TSP 实例 eil51 中搜索策略对路径的影响

2.3　路径的局部优化

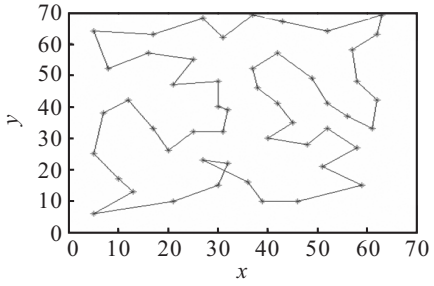
2-Opt[13]是一种时间复杂度为 $O(n^2)$ 的局部优化算子, 经常用于 TSP 路径的局部优化. 由于其过于依

赖问题本身特性, 单独使用易陷入局部最优, 经常与其他算法结合使用. 蚁群算法通过逐步搜索获得最终解, 这种逐步搜索的方式容易形成相邻两点颠倒, 在接近最优解时, 很难进一步优化或优化代价太高, 如图 2(a) 所示. 为了避免不必要的迭代和提高算法的求解质量, 在 OMACO 中使用简化的 2-Opt 进行局部调优, 时间复杂度为 $O(n)$. 设路径 $\text{route} = \{c_0, c_1, \cdots, c_{n-1}\}$, n 为路径中城市数目; 设 $d(i,j)$ 为 c_i 与 c_j 的距离. 为便于算法流程说明, 针对 $\forall i \in Z$, 当 $i \geqslant n$ 时, c_i 等同于 $c_{i\%n}$. 算法起始令 $i = 0$.

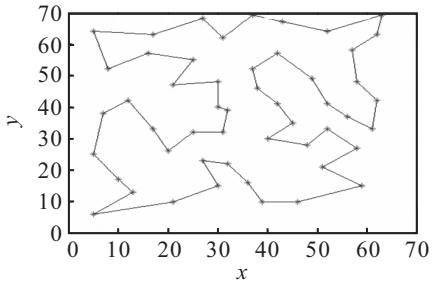
Step 1: 若 $d(i,i+1)+d(i+2,i+3) > d(i,i+2)+d(i+1,i+3)$, 则互换 c_{i+1} 与 c_{i+2};

Step 2: 若 $i < n$, 则 $i \leftarrow i+1$ 执行 Step 1, 否则算法完成.

优化后效果如图 2(b) 所示.



(a) 局部优化前



(b) 局部优化后

图 2 局部优化前后对比

2.4 跳出局部最优

OMACO 采用精英策略与跳出局部最优相结合的方式, 如果一个最优解连续规定次数无法被更新, 则保留该解, 并重置信息素. 由于优化后期信息素积累过多, 容易导致算法停滞, 本文方式首先保留局部最优解, 然后重置相应路径上的信息素, 避免了算法停滞, 增强了算法跳出局部最优的能力.

2.5 信息素的更新方式

基本蚁群算法的信息素更新方式见式 (4), 在完成一次迭代后, 所有蚂蚁均进行信息素的更新. 这种更新方式增强了系统的正反馈, 有利于算法更快地收敛. 但优化后期较优路径上的信息素会积累较多, 影响种群的多样性, 算法容易陷入局部最优而停滞, 最终导致求解质量的下降. OMACO 仅从 3 个角色的蚁群中选取最短路径进行信息素的更新, 结合 2.4 节重置信息素的处理方式, 有效地避免了信息素积累过多的问题. 同时, 因本文算法是多角色蚁群算法, 3 条最短路径是在不同的策略下形成的, 差别较大 (见图 1), 故有效地提高了种群的多样性, 最终提高了算法的求解质量. 具体更新方式如下:

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}(t). \tag{12}$$

其中

$$\Delta\tau_{ij}(t) = \Delta\tau_{ij}^{n} + \Delta\tau_{ij}^{d} + \Delta\tau_{ij}^{o}; \tag{13}$$

$$\Delta\tau_{in}^{s}(t) = \begin{cases} Q/L_{\text{best}}^{s}(t),\ (i,j) \in \text{ tour by best ant},\\ \quad s \in \{n,d,o\};\\ 0,\ \text{otherwise}; \end{cases} \tag{14}$$

$\Delta\tau_{ij}^{n}$ 为普通蚁群的信息素增量, $\Delta\tau_{ij}^{d}$ 为开拓蚁群的信息素增量, $\Delta\tau_{ij}^{o}$ 为寻优蚁群的信息素增量; $L_{\text{best}}^{s}(t)$ 为角色 s 的蚁群当前最优路径长度.

3 仿真实验及结果

为了验证 OMACO 的有效性, 本文从 TSPLIB 标准库 (http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/) 中选取 50 个经典实例进行仿真实验. 当前多数文献在求解小规模 TSP 问题时, 都可以达到较为理想的效果, 但对于中大型 TSP 问题, 其效果较差或不进行论述. 本文对小规模和中大规模 TSP 都进行了详细的对比实验. 编程软件为 g++ 4.3, CPU 为 i7-3770 K, 内存 8 G, Suse Linux Enterprise Server 11 sp2 64 位操作系统.

3.1 求解质量对比

ACS[2] 是蚁群算法中应用较为广泛的一种改进模型, 有文献表明, 其求解质量优于基本的蚁群算法, 且在求解质量和收敛速度方面达到了较好的平衡. 在本实验中, 分别用 ACS 和 OMACO 对 50 个实例进行 30 次运算, 每次进行 1 000 次迭代. ACS 蚂蚁数 (m) 取值等同于城市数, 其余参数为 $\rho = 0.5$, $\alpha = 1.0$, $\beta = 3.0$, $Q = 120$. OMACO 中 m 取值为城市数的 1.5 倍, 其余参数为 $\rho = 0.9$, $\varepsilon = 1.5$, $\xi = 6.0$, $Q = 120$. 所有测试均采用全浮点数运算, 结果见表 1.

在表 1 中, 已知最优解为当前所知的最短路径长度, 其中部分数据来自于 TSPLIB 标准库, 其余部分参考文献 [14]. 算法所得最优解为 30 次运算所得最短路径. 偏差表示所得最优解与已知最优解之间的偏离程度. 在所有测试结果中, OMACO 均远超 ACS. ACS 平均偏差为 8.5%, OMACO 平均偏差为 1.96%, 即在求解精度方面 OMACO 较 ACS 有 76% 的提高. 相对于 TSP 实例 kroB150、tsp225 和 ulysses22, OMACO 所得

表 1　求解质量对比

序号	TSP 实例	已知最优解	算法所得最优解		偏差/%	
			ACS	OMACO	ACS	OMACO
1	a280	2 579.00	3 005.50	2 606.32	16.54	1.06
2	ali535	2 023.39	2 353.73	2 093.31	16.33	3.46
3	att48	33 523.71	34 379.70	33 523.71	2.55	0
4	berlin52	7 542.00	7 590.07	7 544.36	0.64	0.03
5	bier127	118 282.00	123 365.27	118 767.45	4.30	0.41
6	burma14	30.88	30.88	30.88	0	0
7	ch130	6 110.86	6 506.71	6 127.39	6.48	0.27
8	ch150	6 528.00	6 665.42	6 546.91	2.11	0.29
9	d198	15 780.00	16 861.08	15 899.33	6.85	0.76
10	d493	35 002.00	39 983.51	36 146.52	14.23	3.27
11	d657	48 912.00	56 535.73	53 095.98	15.59	8.55
12	eil51	426.00	440.79	428.87	3.47	0.67
13	eil76	538.00	556.94	538.36	3.52	0.07
14	eil101	629.00	685.85	642.03	9.04	2.07
15	fl417	11 861.00	12 661.42	11 991.18	6.75	1.10
16	gil262	2 378.00	2 546.80	2 406.96	7.10	1.22
17	gr96	512.31	539.57	512.49	5.32	0.04
18	gr137	698.53	779.70	703.68	11.62	0.74
19	gr202	—	539.89	488.97	10.41	0
20	gr229	—	1 779.42	1 666.46	6.78	0
21	gr431	1 714.14	2 192.29	1 868.73	27.89	9.02
22	gr666	2 943.58	3 630.78	3240.29	23.35	10.08
23	kroA100	21 282.00	22 319.01	21 320.96	4.87	0.18
24	kroA200	29 368.00	31 419.63	29 457.69	6.99	0.31
25	kroB100	22 141.00	23 584.78	22 196.53	6.52	0.25
26	kroB150	26 130.00	27 660.02	26 127.35	5.86	0
27	kroB200	29 437.00	31 523.68	29 819.04	7.09	1.30
28	kroC100	20 749.00	21 294.39	20 750.76	2.63	0.01
29	kroD100	21 294.00	22 640.24	21 415.86	6.32	0.57
30	kroE100	22 068.00	23 085.56	22 107.52	4.61	0.18
31	lin105	14 379.00	14 618.75	14 382.99	1.67	0.03
32	lin318	42 029.00	45 815.46	42 371.29	9.00	0.81
33	p654	34 643.00	40 362.35	37 118.84	16.51	7.15
34	pcb442	50 778.00	58 410.35	53 842.84	15.03	6.04
35	pr76	108 159.00	114 548.57	108 202.14	5.91	0.04
36	pr107	44 303.00	46 478.14	44 620.18	4.91	0.72
37	pr226	80 369.00	82 983.87	80 382.45	3.25	0.02
38	pr439	107 217.00	113 797.02	108 651.12	6.14	1.34
39	rat99	1 211.00	1 270.56	1 219.86	4.92	0.73
40	rat195	2 323.00	2 421.73	2 342.24	4.25	0.83
41	rat575	6 773.00	7 865.38	7 332.81	16.13	8.27
42	rat783	8 806.00	10 382.06	9 536.24	17.90	8.29
43	rd400	15 281.00	17 128.43	15 417.16	12.09	0.89
44	st70	675.00	701.47	678.62	3.92	0.54
45	ts225	—	130 864.42	127 090.80	2.97	0
46	tsp225	3 916.00	4 160.95	3 894.46	6.26	0
47	u574	36 905.00	42 789.92	38 603.37	15.95	4.60
48	u724	41 910.00	48 369.14	44 408.27	15.41	5.96
49	ulysses22	75.67	75.88	75.31	0	0
50	vm1 084	239 297.00	280 979.88	250 172.17	17.42	4.54

最优解略小于已知最优解, 其原因是计算机存在浮点数规整化且精度有限. 对于同一条回路, 从不同的起点计算, 很可能得到不同的长度, 但相差不大. 另外, 表 1 中所列已知最优解, 有部分为整数解, 其值小于浮点数解, 例如: eil51 已知最优整数解为 426, 已知最优浮点数解为 429.98. 鉴于以上原因, 表 1 中

部分偏差在0.5%以下的解, 实质上已经达到了已知最优. 在50组测试中, OMACO偏差低于0.5%的有20组, 比例高达40%; ACS有2组偏差低于0.5%, 比例仅为4%. 如果以达到已知最优解为准, 则OMACO的求解精度为ACS的10倍. 综合以上分析, OMACO在求解质量上远超ACS, 优势非常明显.

文献[6]的算法(简记为A6)采用了增强信息素对比度的方式改进ACS, 在表现形式上与本文算法有一定类似, 因此单独与A6进行了对比, 见表2. 从表2中可以看出, 由于OMACO充分利用了对象的信息, 较大幅度地增强了种群的多样性, 求解质量高于A6.

表 2 与A6纵向对比

实例(已知最优解)	算法	所得最优解	平均值
eil51(426)	A6	429.74	437.20
	OMACO	428.87	428.96
st70(675.00)	A6	682.31	689.92
	OMACO	678.62	681.38
eil76(538.00)	A6	554.43	550.21
	OMACO	538.36	540.31

在与ACS和A6纵向比较的基础上, 本文与其他几种常见的TSP求解算法进行了横向比较, 其中包括: IMGRA[14]、DGSO[15]、SADPSO[16]和文献[11]算法(简记为A11), 结果见表3和表4. 从中可以看出, OMACO的求解质量高于所有对比算法, 尤其在表4中, OMACO的平均值优于对比算法的所得最优解, 从而进一步表明了OMACO具有很强的稳定性.

表 3 eil51横向对比

实例(已知最优解)	算法	所得最优解	平均值
eil51(426)	IMGRA	429.53	—
	DGSO	428.87	429.47
	SADPSO	436.77	440.78
	A11	447	477
	OMACO	428.87	428.96

表 4 与算法IMGRA和A11横向对比

实例(已知最优解)	算法	所得最优解	平均值
st70(675.00)	IMGRA	725.51	—
	A11	712	723
	OMACO	678.62	682.16
ch150(6 528.00)	IMGRA	6 971.77	—
	A11	6 664	7 062
	OMACO	6 546.91	6 556.42
gil262(2 378.00)	IMGRA	2 617.37	—
	A11	2 564	2 639
	OMACO	2 406.96	2 436.78

3.2 求解质量的统计对比

为了更直观地对求解质量做进一步测试, 在50个TSP实例中选取eil51、kroB200、lin318和pr439各进行100次实验, 每次进行1 000次迭代. 实验环境和参数均与3.1节相同. 图3为统计分布图. 从图3中可以明显看出, OMACO的求解质量对比ACS有显著提高, 而且OMACO所得解较为集中, 表明了算法有较好的稳定性和较强的鲁棒性. 所求得4个实例的最短路径如图4所示.

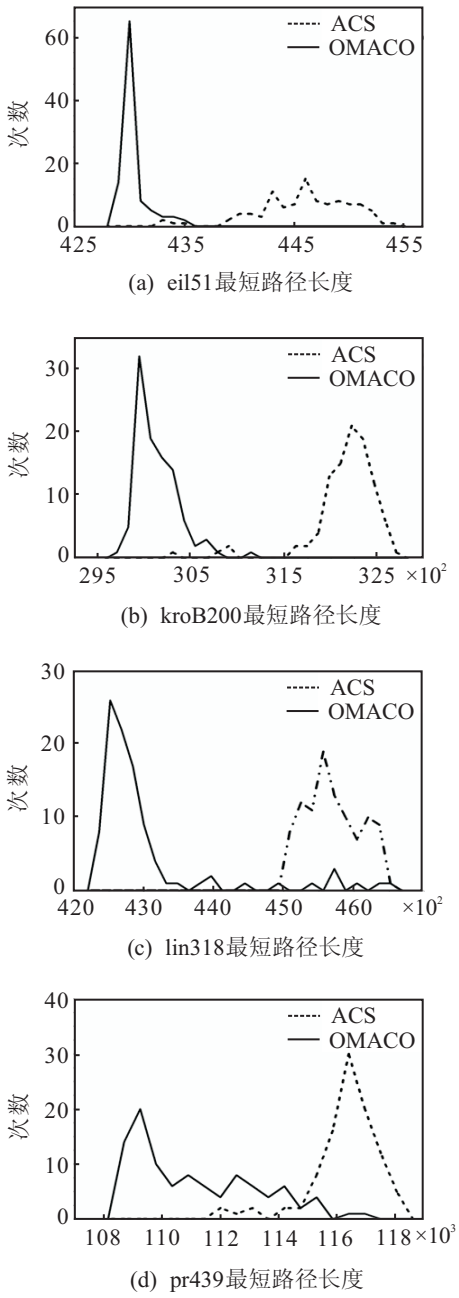


图 3 所得最优解统计分布

3.3 收敛速度的统计对比

在3.2节的测试中, 统计了两种算法达到最优解的迭代代数, 图5为迭代代数统计分布图. 从图5中可以看出, OMACO达到最优解的迭代代数明显小

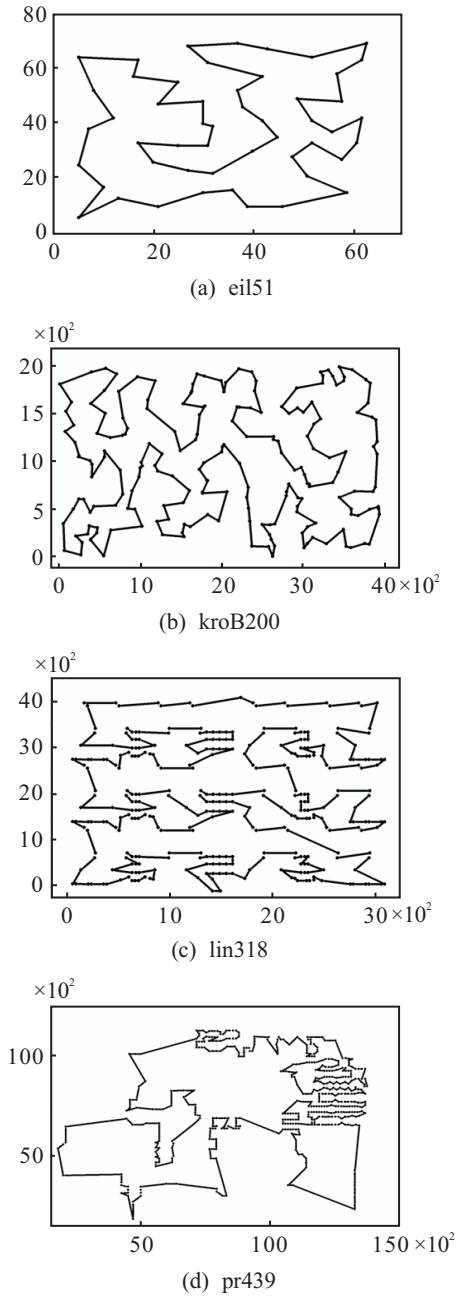


图 4　最短路径

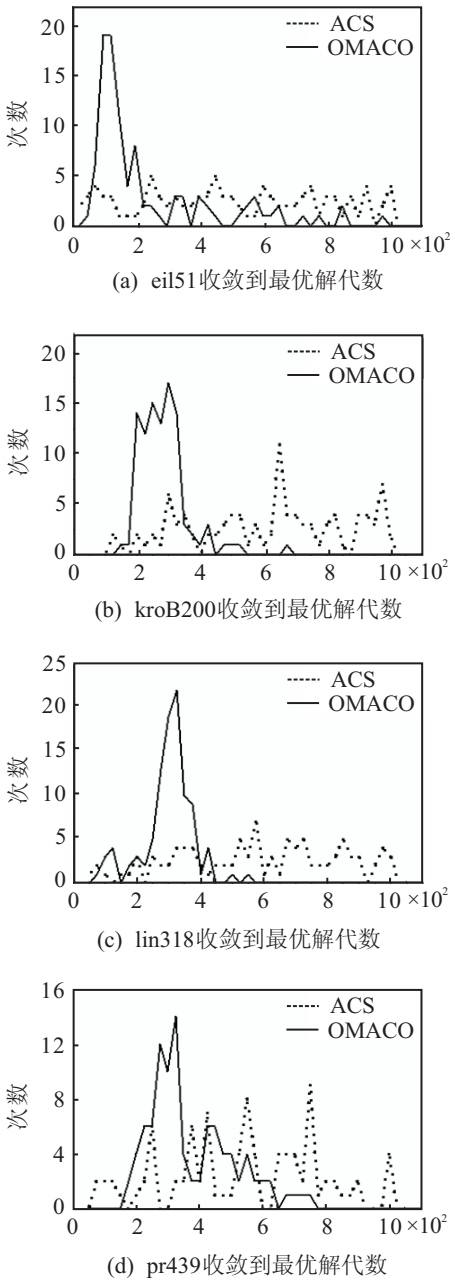


图 5　迭代代数统计分布图

于 ACS, 且相对比较集中, 表明 OMACO 非常稳定, 有较强的搜索能力. 同时, OMACO 达到最优的迭代代数随问题规模的扩大而稳步上升, 表明算法具有很强的鲁棒性.

结合图 3 还可以看出, OMACO 有着较强的跳出局部最优的能力. 以实例 pr439 为例, ACS 有 7 次在 200 代以内就已经达到了最优, 而由 3.2 节分析可知, ACS 的求解质量相对较低, 这实质上表明 ACS 已陷入局部最优, 算法停滞. OMACO 则不同, 达到最优解的迭代次数相对集中, 结合 OMACO 有较高的求解质量, 表明了在达到较高质量解之前 OMACO 没有停滞, 具有很强的跳出局部最优的能力. 表 5 对数据进行了更深层次的比对, 其结果与上述一致. 对于 eil51、kroB200 和 lin318, OMACO 达到最优解所用代数均值约为 ACS 的一半. 从整体上看, OMACO 具有较高的收敛速度, 同时具有很强的跳出局部最优的能力.

表 5　迭代代数统计对比 (均值: μ, 标准差: ε)

算法	eil51		kroB200		lin318		pr439	
	μ	ε	μ	ε	μ	ε	μ	ε
ACS	498	286	614	243	582	251	497	256
OMACO	241	101	284	78	300	82	410	158

4　结　论

在对蚁群算法进行深入研究和充分利用问题本身所包含信息的基础上, 本文提出了一种面向对象的多角色蚁群算法 (OMACO). 传统改进算法没有考虑求解对象本身信息, 对算法搜索能力提升较为有限.

OMACO充分考虑了TSP的空间信息, 进而对城市进行分类; 在此基础上, 对蚁群进行了角色的划分, 大幅度提高了蚁群的多样性; 在求解精度上较对比算法提升了76%; 如果以获得已知最优解为标准, 求解质量达到了对比算法的10倍. OMACO中引入了赏罚因子, 在增强多样性的同时, 大幅度提高了收敛速度, 在最好的情况下达到了对比算法的2倍. 仿真实验表明, OMACO是可行的、有效的, 对比传统的改进算法, 在求解质量和收敛速度方面均有大幅度提高. OMACO并不仅限于解决TSP问题, 将其应用于更多优化问题是下一步要努力的方向.

参考文献(References)

[1] Verma O P, Kumar P, Hanmandlu M, et al. High dynamic range optimal fuzzy color image enhancement using artificial ant colony system[J]. Applied Soft Computing, 2012, 12(1): 394-404.

[2] 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.

[3] Tang J, Ma Y, Guan J, et al. A Max-min ant system for the split delivery weighted vehicle routing problem[J]. Expert Systems with Applications, 2013, 40(18): 7468-7477.

[4] Bullnheimer B, Hartl R F, Strauss C. A new rank based version of the ant system: A computational study[J]. Central European J for Operations Research and Economics, 1999, 7(1): 25-38.

[5] Lutuksin T, Pongcharoen P. Best-worst ant colony system parameter investigation by using experimental design and analysis for course timetabling problem[C]. The 2nd Int Conf on Computer and Network Technology. Bangkok, 2010: 467-471.

[6] 孟祥萍, 片兆宇, 沈中玉, 等. 基于方向信息素协调的蚁群算法[J]. 控制与决策, 2013, 28(5): 782-786.
(Meng X P, Pian Z Y, Shen Z Y, et al. Ant algorithm based on direction-coordinating[J]. Control and Decision, 2013, 28(5): 782-786.)

[7] Zhao N, Wu Z, Zhao Y, et al. Ant colony optimization algorithm with mutation mechanism and its applications[J]. Expert Systems with Applications, 2010, 37(7): 4805-4810.

[8] 郭蕴华, 袁成. 一种异步航迹关联的变异蚁群算法[J]. 电子学报, 2012, 40(11): 2200-2205.
(Guo Y H, Yuan C. A mutation ant colony algorithm for the asynchronous track correlation[J]. Acta Electronic Sinica, 2012, 40(11): 2200-2205.)

[9] Cecilia J M, García J M, Nisbet A, et al. Enhancing data parallelism for ant colony optimization on gpus[J]. J of Parallel and Distributed Computing, 2013, 73(1): 42-51.

[10] 黎自强, 田茁君, 王奕首, 等. 求解平衡约束圆形Packing问题的快速启发式并行蚁群算法[J]. 计算机研究与发展, 2012, 49(9): 1899-1909.
(Li Z Q, Tian Z J, Wang Y H, et al. A fast heuristic parallel ant colony algorithm for circles packing problem with the equilibrium constraints[J]. J of Computer Research and Development, 2012, 49(9): 1899-1909.)

[11] 李擎, 张超, 陈鹏, 等. 一种基于粒子群参数优化的改进蚁群算法[J]. 控制与决策, 2013, 28(6): 873-878.
(Li Q, Zhang C, Chen P, et al. Improved ant colony optimization algorithm based on particle swarm optimization[J]. Control and Decision, 2013, 28(6): 873-878.)

[12] Huang C L, Huang W C, Chang H Y, et al. Hybridization strategies for continuous ant colony optimization and particle swarm optimization applied to data clustering[J]. Applied Soft Computing, 2013, 13(3): 3864-3872.

[13] Chiang C W, Lee W P, Heh J S. A 2-Opt based differential evolution for global optimization[J]. Applied Soft Computing, 2010, 10(4): 1200-1207.

[14] 饶卫振, 金淳, 陆林涛. 考虑边位置信息的求解ETSP问题改进贪婪算法[J]. 计算机学报, 2013, 36(4): 836-850.
(Rao W Z, Jin C, Lu L T. An improved greedy algorithm with information of edges' location for solving the euclidean traveling salesman problem[J]. Chinese J of Computers, 2013, 36(4): 836-850.)

[15] 周永权, 黄正新, 刘洪霞. 求解TSP问题的离散型萤火虫群优化算法[J]. 电子学报, 2012, 40(6): 1164-1170.
(Zhou Y Q, Huang Z X, Liu H X. Discrete glowworm swarm optimization algorithm for TSP problem[J]. Acta Electronic Sinica, 2012, 40(6): 1164-1170.)

[16] 张长胜, 孙吉贵, 欧阳丹彤. 一种自适应离散粒子群算法及其应用研究[J]. 电子学报, 2009, 37(2): 299-304.
(Zhang C S, Sun J G, Ouyang D T. A self-adaptive discrete particle swarm optimization algorithm[J]. Acta Electronic Sinica, 2009, 37(2): 299-304.)

(责任编辑: 李君玲)