动态凸包引导的偏优规划蚁群算法求解 TSP 问题

马学森[1,2]，宫帅[1]，朱建[1]，唐昊[3]

（1. 合肥工业大学计算机与信息学院，安徽 合肥 230009；2. 广东三水合肥工业大学研究院，广东 佛山 528000；
3. 合肥工业大学电气与自动化工程学院，安徽 合肥 230009）

摘 要：针对蚁群算法搜索空间大、收敛速度慢、容易陷入局部最优等缺陷，提出一种基于动态凸包引导的偏优规划蚁群算法。改进后的算法动态控制蚂蚁的待选城市范围，有助于在跳出局部最优并向全局最优逼近的基础上减少蚂蚁搜索空间；同时，引入延陷漂流因子和基于待选城市构建的凸包来干预当前蚂蚁的城市选择，增加算法前期解的多样性并提高蚂蚁的偏优规划能力；再利用局部与整体相结合的完整路径信息、凸包的构建信息来协调信息素的更新，引导后继蚂蚁路径偏优规划，提高算法的求解精度；设计具有收敛性的信息素最大最小值限制策略，既加快算法的求解速度又避免算法过早停滞；最后在 4 种经典 TSP 模型上应用改进后的算法。仿真结果表明，所提算法在求解精度和收敛速度等方面均有显著提高，且具有较好的适用性。

关键词：蚁群算法；二维凸包；TSP；偏优规划

中图分类号：TP18

文献标识码：A

doi: 10.11959/j.issn.1000−436x.2018218

Ant colony algorithm of partially optimal programming based on dynamic convex hull guidance for solving TSP problem

MA Xuesen[1,2], GONG Shuai[1], ZHU Jian[1], TANG Hao[3]

1. School of Computer and Information, Hefei University of Technology, Hefei 230009, China
2. Research Institute of Sanshui & Hefei University of Technology in Guangdong, Foshan 528000, China
3. School of electrical and Automation Engineering, Hefei University of Technology, Hefei 230009, China

Abstract: To solve basic ant colony algorithm's drawbacks of large search space, low convergence rate and easiness of trapping in local optimal solution, an ant colony algorithm of partially optimal programming based on dynamic convex hull guidance was proposed. The improved algorithm dynamically controlled the urban selection range of the ants, which could reduce the search space of ants on basis of helping the algorithm to jump out of local optimal solution to global optimal solution. Meanwhile, the delayed drift factor and the convex hull constructed by the cities to be chosen were introduced to intervene the current ants' urban choice, it could increase the diversity of the early solution of the algorithm and improve the ability of ants' partially optimal programming. Then the pheromone updating was coordinated by using construction information of convex hull and the complete path information that combined local with whole, it could improve the accuracy of the algorithm by guiding the subsequent ants to partially optimal programming. The pheromone maximum and minimum limit strategy with convergence was designed to avoid the algorithm's premature stagnation and accelerate the solving speed of the algorithm. Finally, the proposed algorithm was applied to four classic TSP models. Simulation results show that the algorithm has better optimal solution, higher convergence rate and better applicability.

Key words: ant colony algorithm, convex hull, TSP, partially optimal programming

收稿日期：2017−07−17；**修回日期**：2018−07−02

基金项目：国家自然科学基金资助项目（No.61573126）；广东省科技发展专项基金资助项目（No.2017A010101001）；中央高校基本科研业务费专项基金资助项目（No.JZ2016HGBZ1032）；国家留学基金资助项目

Foundation Items: The National Natural Science Foundation of China (No.61573126), The Special Funds for Science and Technology Development of Guangdong Province (No.2017A010101001), The Central University Basic Business Expenses Special Funding for Scientific Research Project (No.JZ2016HGBZ1032), China Scholarship Council Foundation

1 引言

TSP（travelling salesman problem）[1]又被称为旅行商问题，该问题具体描述为：有位旅行商需要访问 n 个城市，他必须要走完所有城市，然后回到原点，每个城市只能选择一次，目标是形成一条最短路径。TSP 作为一种经典 NP 难题，受到众多专家和学者的关注。他们利用组合优化提出很多启发式搜索算法，如遗传算法[2-3]、模拟退火算法[4]、人工神经网络[5]、禁忌搜索算法[6]、蚁群算法[7]、免疫算法[8]等。其中，蚁群算法作为一种仿生学算法，在 20 世纪 90 年代由 Dorigo 等[9-10]提出，由于它具有较好的分布式计算能力和较强的顽健性，同时易与其他算法融合，被广泛应用于一些 NP 难题的求解上。

蚁群算法是从自然界中蚂蚁觅食的群体行为得到启发而提出的，蚂蚁觅食时会在已走路径上释放信息素，信息素浓度越高的路径被后继蚂蚁选择的概率越大。蚁群算法求解 TSP 过程为：将 m 只蚂蚁随机放到 n 个城市节点上，每个蚂蚁按照一定的依据从没有访问过的城市中选择下一跳城市节点，同时每行走一步或完成整个城市访问后更新所有路径上的信息素浓度。

但蚁群算法求解时蚂蚁的城市选择范围较大，同时非优路径的多次出现和有些路径上的信息素会增加算法的迭代次数及收敛于局部最优解的概率。为弥补上述缺陷，本文基于二维凸包对蚁群算法进行改进，首先动态筛选蚂蚁的待选邻居城市节点，在有助于算法跳出局部最优的基础上，减少蚂蚁的城市选择范围；其次将延陷漂流因子和二维凸包融入城市选择策略，增加算法前期解的多样性，减少非优路径的出现并提高算法收敛速度；最后将完整的路径信息、二维凸包信息融入信息素更新策略，提高蚂蚁的路径寻优能力，并用改进的信息素最大最小值限制策略来进一步加快算法的收敛速度。

二维凸包对本文算法的作用体现为：利用每个当前城市节点与待选邻居节点构建的凸包区域范围之间的关系来影响蚂蚁下一跳城市节点的选择；当所有蚂蚁遍历完成，对迭代最优蚂蚁行走路径上的城市节点分别与其待选邻居节点再次构建凸包，通过构建的凸包及存在的凸包角（如图 1(a)和图 1(b)所示，其中，图 1(b)不存在城市 i 的凸包角）来获取蚂蚁遍历的方向与深度，协调后继蚂蚁的走向。将已构建凸包的区域范围与凸包角融入蚁群算法，更加有利于蚂蚁的行走与协作，并可以加快算法逼近于最优解。

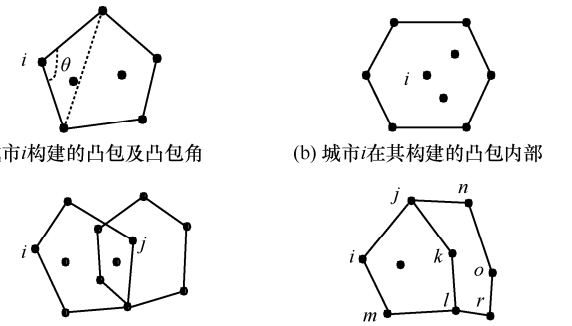
最后，在 Oliver30、Eil51、St70、Eil76 这 4 个经典的 TSP 模型上验证具有收敛性的信息素最大最小值限制策略的有效性以及本文算法的精确性、收敛的快速性与适用性。其中，在 Eil51 上以迭代次数作为衡量指标对所提出的信息素最大最小值限制策略进行验证，同时延伸到其他 3 个 TSP 模型上也是有效的；以 Oliver30 和 Eil51 为例对本文算法的精确性和收敛的快速性进行详细验证，并引用到具有不同城市规模和位置的 St70 和 Eil76 中进行实验，进一步表明本文算法能在保证求解精度的基础上快速收敛，验证了本文算法的适用性。

2 相关工作

众多专家和学者针对蚁群算法的诸多缺陷进行了不同的改进研究。在信息素或节点选择策略方面[11-14]，文献[11]提出了一种释放在城市节点上的方向性信息素和探索因子，但没有考虑城市节点的位置及相对方向，随着城市规模的逐渐扩大，求解误差随之增加，同时收敛速度的提高并不显著。在最大最小蚂蚁系统方面[15-18]，文献[17]提出基于混沌的最大最小蚁群算法，采用 tent 混沌映射的方法产生初始信息素，当算法陷入长时间停滞状态时利用混沌映射扰动信息素，虽能扩大蚁群的搜索范围，但也进一步增加了算法的迭代次数。在图论优化蚁群方面[19-21]，文献[19]对整体 TSP 模型构建凸包，以凸包顶点作为蚂蚁出发点进行搜索，来减少算法的迭代次数，但该文只是对 TSP 模型进行凸包的构建，从而影响蚂蚁的初始位置分布，后续并没有把凸包融入蚁群算法，对蚁群算法本身没有实质性的改变。在蚂蚁多态方面[22-25]，文献[22]提出一种基于动态局部搜索的蚁群算法，为每个蚂蚁分配一个侦查蚁，增加蚂蚁局部搜索能力，并根据迭代效果动态更新信息素，但该算法实质上是动态增加参与迭代的蚂蚁数，并没有显著提高算法的求解精度。

在上述研究的基础上并针对其存在的缺陷，本文利用凸包的构建及构建的区域范围与凸包角，提出一种基于动态凸包引导的偏优规划蚁群算法（ACADCG, ant colony algorithm based on dynamic convex guidance），来求解 TSP 全部城市节点遍历问题。ACADCG 限制蚂蚁下一跳的待选邻居节点

范围，减小搜索空间，此外，随着蚂蚁待选邻居节点改变而动态变化的凸包（如图 1(c)和图 1(d)所示，其中，图 1(d)存在的 2 个凸包分别为 *ijklm* 和 *ijnorlm*，有助于算法跳出局部最优；ACADCG 还引入延陷漂流因子来增加城市选择的随机性，提高算法前期解的多样性；同时将凸包的区域范围应用到蚂蚁选择城市节点概率式中，引导蚂蚁下一跳的偏优规划，减少交叉路径的出现；当所有蚂蚁遍历完，通过改进的路径信息、已构建的凸包及凸包角来修改信息素的更新规则，利用该规则来弱化导向非优解路径上的信息素及非优解包含的非优路径上的信息素，引导后继蚂蚁路径偏优规划，并引入与当前迭代次数相关的策略限制信息素上下限，进一步减少算法的迭代次数。



(a) 城市*i*构建的凸包及凸包角　(b) 城市*i*在其构建的凸包内部

(c) 不同的城市*i*与*j*构建的凸包　(d) 不同迭代中相同城市*i*构建的凸包

图 1　凸包图

3　基于动态凸包引导的改进蚁群算法

本文针对蚁群算法搜索空间大、易陷入局部最优及收敛速度慢等缺陷，在蚂蚁的待选邻居城市范围、城市选择策略、信息素更新与控制策略上进行了改进。

3.1　待选邻居城市范围的控制

在蚁群算法中，位于城市 i 的第 k 只蚂蚁选择下一跳城市节点时，需要按照既定策略在剩下的所有城市中选择一个，存在选择空间规模较大、搜索时间较长的缺陷。而在已形成的最优路径中，与某一城市相连的前后节点必在距离该城市最近的若干个节点中，即蚂蚁选择下一节点时，只需在距离该城市最近的若干个待选邻居节点中访问便可。本文把当前所有剩余城市到蚂蚁 k 所在城市的距离按从小到大排序，在排序结果中选择前$\lceil \lambda n_1 \rceil$（其中，$\lceil \bullet \rceil$为向上取整，$\lambda$ 为蚂蚁选择范围参数，n_1 为当前所有剩余城市数）个城市作为蚂蚁待选邻居城市节点集合 $allowed_k$。此策略将大大缩小搜索空间，节省搜索时间，同时有利于算法后续对蚂蚁遍历方向的控制。

ACADCG 算法在迭代时，求得的全局最优解连续 N_1 次（N_1 值与最大迭代次数 $N_{\max}$ 相关）迭代后仍然不发生变化，则可能暂时陷入局部最优，此时，动态调整参数 $\lambda = \lambda + 0.05$，帮助算法跳出局部最优。

3.2　城市选择策略

传统蚁群算法求解时，概率计算式 $P_{ij}^k(t)$ 主导着蚂蚁的城市选择，但有时算法容易较早地陷入局部最优且解的精度并不高。ACADCG 算法将延陷漂流因子引入城市选择策略，避免算法较早陷入局部最优；同时将二维凸包引入城市选择策略来提高解的精度。

3.2.1　延陷漂流因子

为了避免算法过早地陷入局部最优，提出延陷漂流因子 μ（如式(1)所示），来增加蚂蚁前期城市选择的随机性，引导蚂蚁走向。

$$\mu(t+n_2) = [\mu(t) - 0.2N_c] \tag{1}$$

其中，$[\bullet]$为四舍五入取整，N_c 为算法当前的迭代次数，$\mu(t)$为 t 时刻延陷漂流因子值，$\mu(t+n_2)$为经过时间 n_2，所有蚂蚁完成一次遍历后更新的延陷漂流因子值。

所有蚂蚁每完成一次迭代，便更新一次 μ 值。迭代过程中，蚂蚁 k 在选择下个城市时，会首先生成一个 0~n（n 为城市数）之间的随机数 e，若 $e < \mu$，则在已筛选的待选邻居城市节点集合 $allowed_k$ 中随机选择一个城市节点；反之，则根据 3.2.2 节中的 GCAPOP 方法选择下一跳城市。

由于 μ 随着迭代次数的增加而递减，在初始阶段，蚂蚁选择城市的随机性大，会增加算法解的多样性。不过 μ 值最终变为非正数，蚂蚁按照本文所设概率计算式(3)进行选择后，信息素在最优路径上会持续增加，算法将最终收敛于最优解。

3.2.2　引导当前蚂蚁偏优规划的凸包

当蚂蚁位于城市 i 时，如果下一跳城市 j 与蚂蚁在上一跳城市 h 时未遍历的待选邻居城市偏离较远，则蚂蚁后续可能会偏离这些未遍历的待选邻居城市，一旦持续偏离将引起蚂蚁折回遍历，而蚂蚁在折回遍历时极易产生交叉路径，降低解的精度。

为解决上述问题，本文提出引导当前蚂蚁偏优规划方法 GCAPOP，用二维凸包引导蚂蚁下一跳城市的选择，使蚂蚁优先对最近已遍历城市凸包区域

范围交集中未遍历的城市进行访问（不包括交集中边上城市节点），减少蚂蚁因折回遍历而可能引起的路径交叉（含有交叉路径的解为非优解）。其具体过程为：ACADCG 算法中的蚂蚁位于城市 h 时，需要从城市 h 凸包的区域范围中（除城市 h 外）选取下一个城市 i，当蚂蚁到达城市 i 时，将需要再次从城市 i 凸包的区域范围中（除城市 i 外）选取下一跳城市 j，此时增加城市 h 凸包与城市 i 凸包区域范围交集 C_{hi} 中城市被选为下一跳城市 j 的概率，若蚂蚁依据概率选择城市，则具体采用轮盘赌的方式选出下一跳城市节点 j。传统蚁群算法中蚂蚁选择下一跳节点的概率如式(2)所示。ACADCG 算法中蚂蚁选择下一跳节点的概率如式(3)所示。

$$P_{ij}^{k}(t)=\begin{cases}\dfrac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}(t)]^{\beta}}{\sum_{j\in allowed_k}[\tau_{ij}(t)]^{\alpha}[\eta_{ij}(t)]^{\beta}}, j\in allowed_k\\ 0 \qquad\qquad\qquad\qquad ,\text{其他}\end{cases} \tag{2}$$

$$P_{ij-new}^{k}(t)=\begin{cases}P_{ij}^{k}(t)+p, j\in C_{hi}\\ P_{ij}^{k}(t), \text{其他}\end{cases} \tag{3}$$

其中，式(2)中传统蚁群算法 $P_{ij}^{k}(t)$ 为 t 时刻位于城市 i 的蚂蚁 k 选择城市 j 为目标城市的概率；已筛选的待选邻居城市节点集合 $allowed_k$ 为蚂蚁 k 下一步能够选择的城市；$\tau_{ij}(t)$ 为 t 时刻路径 (i,j) 上存在的信息素浓度；$\eta_{ij}=\dfrac{1}{d_{ij}}$ 为蚂蚁从城市 i 到城市 j 的启发函数；α、β 分别为信息素浓度和启发函数的权重值；$P_{ij-\text{new}}^{k}(t)$ 为凸包融入城市选择策略求得的路径选择概率；C_{hi} 为上一跳城市 h 凸包与当前城市 i 凸包区域范围交集中的城市节点集合（城市节点 i 与交集中边上城市节点除外，其中若城市 h 不存在，则集合为空）；p 为概率增量。

基于 GCAPOP 方法的 ACADCG 算法增加了凸包区域范围交集中城市节点被选为下一跳节点的概率，提高了算法的收敛速度及解的质量，但并没有排除其他待选邻居节点被选择的可能，不会减少解的多样性，同时为了避免算法较易陷入局部最优，可利用本文所提的延陷漂流因子及改进的信息素更新策略。

3.3 信息素更新策略

ACADCG 算法中的蚂蚁每行走一步便局部更新信息素，当所有蚂蚁一次遍历完成后，便全局更新信息素，最后对信息素范围进行控制。

3.3.1 引导后继蚂蚁偏优规划的凸包

经实验发现，蚂蚁持续遍历某一侧的多个城市后，转向另一侧剩余城市节点继续遍历时，较易产生交叉路径（交叉路径为非优路径）。随着蚂蚁已访问城市的增加，这种非优路径出现的概率就越大。图 2 是 Eil51 在 $\alpha=1$、$\beta=5$ 时一只蚂蚁按照传统蚁群算法中 $P_{ij}^{k}(t)$ 形成的遍历图（为清晰获取蚂蚁遍历前后次序，未将遍历路径完全闭合），在第 39 步，开始出现交叉路径，在之后的遍历中，交叉路径又多次发生。

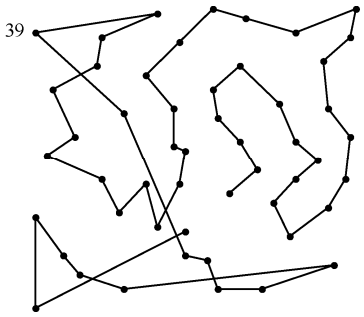


图 2 一次遍历中蚂蚁路径选择情况（Eil51）

为解决上述问题，提出引导后继蚂蚁偏优规划方法 GSAPOP，将二维凸包融入信息素更新策略来弱化引起交叉的路径上的信息素及已形成交叉的路径上的信息素，引导后继蚂蚁路径选择，减少非优路径的产生。

算法每完成一次迭代，对迭代最优蚂蚁走过的路径进行信息素更新时，GSAPOP 方法把更新分为 2 种情况（把迭代最优路径上的城市节点分别与其待选邻居节点构建凸包）。

1) 城市节点 i 为凸包上的凸点（图 1(a)）。此时，按照式(8)中凸点的取值来更新城市节点 i 与下个被选节点 j 之间边的信息素。因为当城市节点 i 为凸包上的凸点时，位于其一侧的待选邻居节点会引导蚂蚁向某一侧遍历。此外，随着城市凸包角 θ 变小或剩余节点数量减少，蚂蚁继续沿着此侧方向持续遍历的可能性就越大，导致蚂蚁完成单侧城市遍历转向另一侧节点访问时，其转向幅度与转向距离会随之变大，路径交叉极易发生，因此，通过式(8)中的 $\Delta\tau$ 对 i—j 边信息素更新幅度进行不同程度地削减，来减少后继蚂蚁中因交叉路径的出现而形成非优解。这里，算法对路径上信息素的削减并不是完全建立在大概率基础上的，而是在蚁群已经搜索到一些非优解后，对那些满足条件的路径进行信

息素修正，减少非优解中已形成交叉的路径及引起交叉的路径对后继蚂蚁的影响，有效地提高蚂蚁整体求解精度。

2) 城市节点 i 在凸包内部（图 1(b)）。此时，按照式(8)中凸包内部的取值来更新城市节点 i 与下个被选节点 j 之间边的信息素。

ACADCG 算法中 GSAPOP 方法会削减起始城市为凸点的交叉路径及引起交叉的路径（图 2）上的信息素更新幅度，有利于后继蚂蚁向更优路径遍历。

3.3.2 偏优信息素更新

1) 信息素局部更新：蚂蚁从当前节点 i 移动到下个节点 j 后，利用式(4)更新路径边 i—j 上信息素。

$$\tau_{ij}(t+1)=(1-\rho_1)\tau_{ij}(t)+\rho_1\tau_0 \tag{4}$$

$$\tau_0=\frac{1}{n^2 d_{\min}} \tag{5}$$

其中，τ_0 是初始信息素值，$\tau_{ij}(t)$ 与式(2)中含义相同，$\tau_{ij}(t+1)$ 为下个时刻蚂蚁经过路径 (i,j) 更新后的信息素值，n 为城市节点数量，$d_{\min}$ 为任意 2 个城市之间的最短长度，ρ_1 为局部信息素挥发系数。

采用信息素局部更新可以适度地减少蚂蚁所走过路径的信息素，使后继蚂蚁选中该路径的可能性减小，增加解的多样性，抑制算法过早停滞。

2) 信息素全局更新：每次迭代结束后，更新最优蚂蚁走过的路径上信息素，在综合考虑局部路径信息（d_{ij}）、整体路径信息（全局最优路径长度 $f(s^{\text{best}})$ 或迭代最优路径长度 $f(s_1^{\text{best}})$）的基础上，将凸包信息融入更新中。更新步骤按照最优蚂蚁获得的全局最优路径与迭代最优路径，并结合当前迭代次数能否整除 ω（$\omega=5$ 为多次实验调控所得的较优值）分别进行。

步骤 1　当迭代次数是 ω 的整数倍时，采用式(6)更新全局最优蚂蚁走过路径的边的信息素。

$$\tau_{ij}(t+n_2)=(1-\rho)\tau_{ij}(t)+\Delta\tau_{ij}^{\text{best}} \tag{6}$$

$$\Delta\tau_{ij}^{\text{best}}=\frac{d_i^{\min}+d_j^{\min}}{2d_{ij}}\frac{1}{f(s^{\text{best}})} \tag{7}$$

步骤 2　当迭代次数不是 ω 的整数倍时，采用式(8)更新迭代最优蚂蚁走过路径的边的信息素，其中，利用 $\Delta\tau$ 进一步来控制信息素更新幅度。

$$\tau_{ij}(t+n_2)=(1-\rho)\tau_{ij}(t)+\Delta\tau_{ij}^{\text{best}}-\frac{\Delta\tau}{2} \tag{8}$$

$$\Delta\tau_{ij}^{\text{best}}=\frac{d_i^{\min}+d_j^{\min}}{2d_{ij}}\frac{1}{f(s_1^{\text{best}})} \tag{9}$$

式(8)中的 $\Delta\tau$ 是以凸包角为基础进行构建的，同时，为了真实地反映出已形成的凸包角对蚂蚁遍历的影响，结合算法前期、中期和后期，对角度范围进行了 3 段划分，并对于不同阶段的权重因子也进行了不同值的设定。其中，$\Delta\tau$ 在迭代最优路径上 4 个不同的城市区间段的具体取值规则如下。

1) 迭代最优路径上前 $\left\lceil\frac{n}{3}\right\rceil$ 个城市之间路径边的信息素更新。若当前路径边 i—j 的起始城市 i 为凸包上的凸点，则 $\Delta\tau$ 取值如式(10)所示；若当前路径边 i—j 的起始城市 i 在凸包内部，则令 $\Delta\tau=0$。

$$\Delta\tau=\begin{cases}\lambda_1\dfrac{(\pi-\theta)}{1.5\pi}(1-\rho)\tau_{ij}(t), 0\leqslant\theta\leqslant\dfrac{\pi}{3}\\ \lambda_1\dfrac{\pi-\theta}{2\pi}(1-\rho)\tau_{ij}(t), \dfrac{\pi}{3}<\theta\leqslant\dfrac{2\pi}{3}\\ \lambda_1\dfrac{\pi-\theta}{2.5\pi}(1-\rho)\tau_{ij}(t), \dfrac{2\pi}{3}<\theta\leqslant\pi\end{cases} \tag{10}$$

2) 迭代最优路径上第 $\left\lceil\frac{n}{3}\right\rceil$ 个城市至第 $2\left\lceil\frac{n}{3}\right\rceil$ 个城市之间路径边的信息素更新。若当前路径边 i—j 的起始城市 i 为凸包上的凸点，则 $\Delta\tau$ 取值如式(11)所示；若当前路径边 i—j 的起始城市 i 在凸包内部，则令 $\Delta\tau=0$。

$$\Delta\tau=\begin{cases}\lambda_2\dfrac{(\pi-\theta)}{\pi}(1-\rho)\tau_{ij}(t), 0\leqslant\theta\leqslant\dfrac{\pi}{3}\\ \lambda_2\dfrac{\pi-\theta}{1.5\pi}(1-\rho)\tau_{ij}(t), \dfrac{\pi}{3}<\theta\leqslant\dfrac{2\pi}{3}\\ \lambda_2\dfrac{\pi-\theta}{2\pi}(1-\rho)\tau_{ij}(t), \dfrac{2\pi}{3}<\theta\leqslant\pi\end{cases} \tag{11}$$

3) 迭代最优路径上第 $2\left\lceil\frac{n}{3}\right\rceil$ 个城市至第 n 个城市之间路径边的信息素更新。若当前路径边 i—j 的起始城市 i 为凸包上的凸点，则 $\Delta\tau$ 取值如式(12)所示；若当前路径边 i—j 的起始城市 i 在凸包内部，则令 $\Delta\tau=0$。

$$\Delta\tau=\begin{cases}\lambda_3\dfrac{2(\pi-\theta)}{\pi}(1-\rho)\tau_{ij}(t), 0\leqslant\theta\leqslant\dfrac{\pi}{6}\\ \lambda_3\dfrac{1.5(\pi-\theta)}{\pi}(1-\rho)\tau_{ij}(t), \dfrac{\pi}{6}<\theta\leqslant\dfrac{5\pi}{6}\\ \lambda_3\dfrac{\pi-\theta}{\pi}(1-\rho)\tau_{ij}(t), \dfrac{5\pi}{6}<\theta\leqslant\pi\end{cases} \quad (12)$$

4) 迭代最优路径上第 n 个城市到源城市之间路径边的信息素更新。由于此时所有节点已遍历完毕，令 $\Delta\tau=0$。

式(6)~式(12)中，$\tau_{ij}(t)$ 与式(2)中相同；$\tau_{ij}(t+n_2)$ 为经过 n_2 个时刻，所有蚂蚁完成一次遍历对路径 (i,j) 更新后的信息素值；ρ 为全局信息素挥发系数；$d_i^{\min}$ 为与城市 i 相距最近的城市距离；$d_j^{\min}$ 为与城市 j 相距最近的城市距离；d_{ij} 为城市 i 和城市 j 间的距离($\dfrac{d_i^{\min}+d_j^{\min}}{2d_{ij}}\in[0.6,1]$)；$f(s^{\text{best}})$ 为全局最优蚂蚁走过的路径长度；$f(s_1^{\text{best}})$ 为迭代最优蚂蚁走过的路径长度；λ_1、λ_2、λ_3 分别为不同值的权重因子（常数）；θ 为城市凸包角。步骤 2 中所包含的 n 与式(5)中含义相同。

3.3.3 具有收敛性的信息素最大最小值限制区间

当信息素更新完成后，为了抑制由于边与边上信息素浓度差距过大而引起算法停滞现象的发生，本文引入信息素最大最小值限制策略。

含有传统公式的信息素最大最小值限制策略虽然能防止信息素过多地聚集在某些路径上，但一定程度上也抑制了算法收敛速度。本文将迭代次数考虑到信息素最大最小值限制策略中，对算法前期、中期及后期进行适当干预，既避免了因不同路径上信息素浓度差距过大导致算法陷入局部最优，又提高算法中后期向全局最优解逼近的概率。

改进后的最大信息素和最小信息素的取值分别如式(13)与式(14)所示，其中，式(14)的形式与传统公式相一致。

$$\tau_{\max}=\frac{1}{1-\rho}\cdot\frac{1}{f}\cdot\frac{4}{3+\dfrac{e}{e^{N_{\max}/(N_{\max}-N_c+1)}}} \quad (13)$$

$$\tau_{\min}=\frac{\tau_{\max}(1-\sqrt[n]{p_{\text{best}}})}{(avg-1)\sqrt[n]{p_{\text{best}}}} \quad (14)$$

其中，当对全局最优路径进行信息素更新时 f 等于全局最优路径长度 $f(s^{\text{best}})$，否则 f 等于迭代最优路径长度 $f(s_1^{\text{best}})$；ρ 与式(6)中含义相同；$N_{\max}$ 为最大迭代次数；N_c 与式(1)中含义相同；avg 为城市数目的一半；n 与式(5)中含义相同；p_{best} 为蚂蚁迭代一次找到最优路径的概率，其值一般取 0.05 或 0.005，本文算法取 0.005。

3.4 ACADCG 算法的实现

ACADCG 算法实现步骤如下，算法流程如图 3 所示。

1) 对环境信息进行初始化。设置各参数 ω、α、β、ρ、ρ_1 的值，蚂蚁个数为 m，城市节点数量为 n。设定最大迭代次数 $N_{\max}$、λ 及 N_1 值。P、λ_1、λ_2、λ_3 分别为不同值的常数。令各条路径上信息素初始值 $\tau_0=\tau_{ij}(0)=\dfrac{1}{n^2d_{\min}}$。将 m 只蚂蚁随机放在 n 个城市上，初始时刻 $t=0$，$\mu=N$，$\Delta\tau_{ij}^{\text{best}}=\Delta\tau=0$，$n_1=n-1$，$N_c=0$，$\mu$ $N_t=0$。

2) 确定待选邻居城市节点集合。从当前所有剩余城市中挑选出较近的$\lceil\lambda n_1\rceil$个城市作为蚂蚁位于当前城市的待选邻居城市节点集合 $allowed_k$。

3) 比较随机数与延陷漂流因子的大小。蚂蚁选择城市前会生成一个 0~n 之间的随机数 e，若 $e\geqslant\mu$，则按照 GCAPOP 方法选择城市；相反，则从集合 $allowed_k$ 中随机选择下一跳节点。

4) 修改禁忌表。城市选择完成之后，将蚂蚁移动到新的城市，同时更新蚂蚁 k 的路径长度，并将该城市加入该蚂蚁的禁忌表中，$n_1=n_1-1$。此外，按照式(4)进行局部信息素的更新。

5) 判断遍历是否完成。禁忌表中是否包含所有城市，若未包含则转步骤 2)；否则转步骤 6)。

6) 若 m 只蚂蚁都构造完成各自的解，则转步骤 7)；否则转步骤 2)。

7) 迭代次数 N_c 加 1。若迭代次数为 5 的整数倍，则按式(6)更新全局信息素；反之，则按照 GSAPOP 方法即式(8)更新全局信息素。并用式(13)与式(14)控制信息素上下限。此外，按照式(1)更新值 μ。

8) 若全局最优解不变，则 $N_t=N_t+1$；否则 $N_t=0$。若 $\dfrac{N_t}{N_1}$ 为非零整数，则 $\lambda=\lambda+0.05$。

9) 判断 N_c 是否小于 $N_{\max}$，如果达到最大迭代次数 $N_{\max}$，则结束循环，输出结果；否则，清空禁忌表并跳转到步骤 2)。

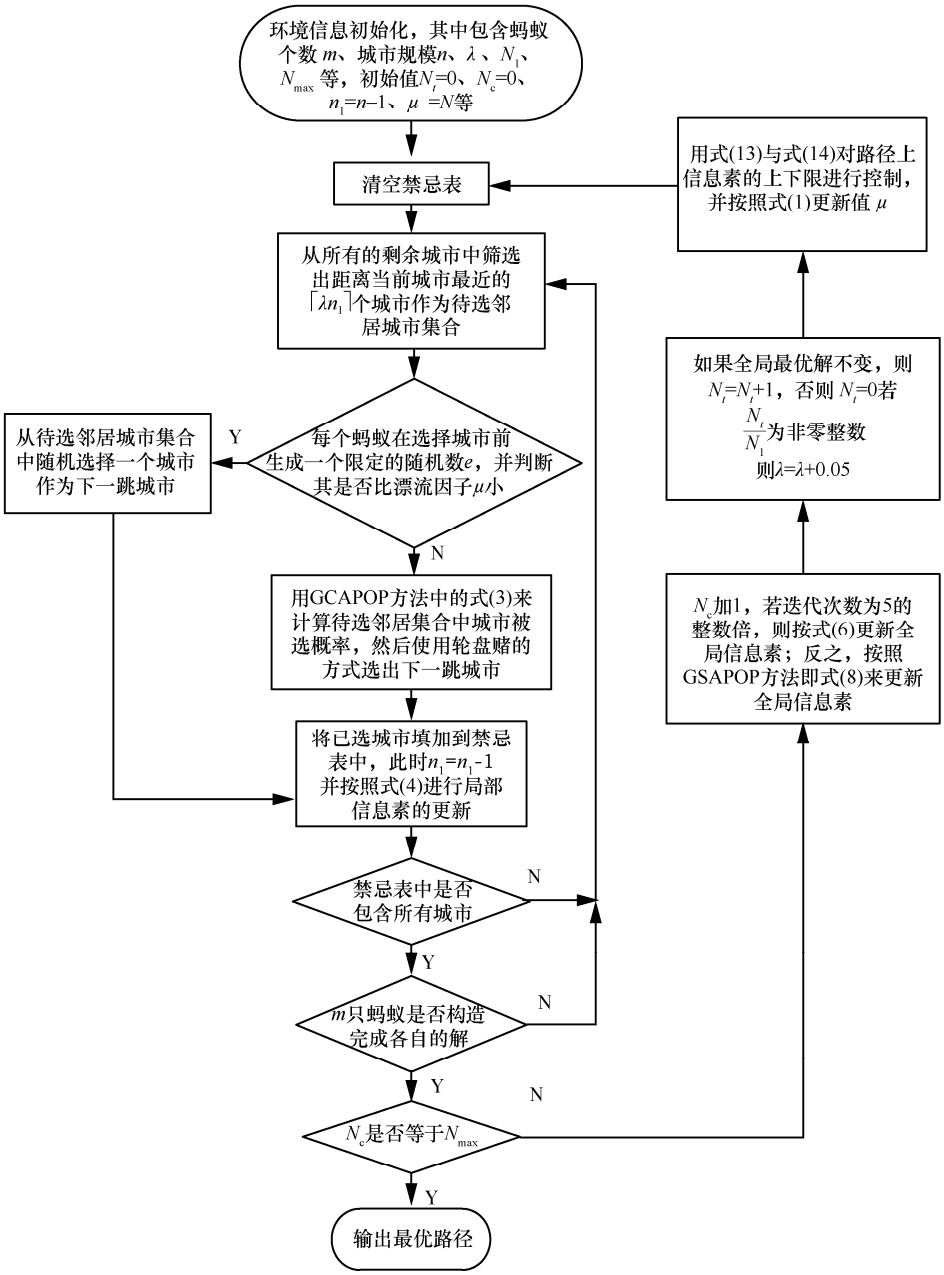


图 3　ACADCG 算法求解 TSP 流程

3.5　ACADCG 算法复杂度

ACADCG 算法复杂度包括时间复杂度与空间复杂度。

3.5.1　ACADCG 算法时间复杂度

本文蚁群算法中 m 只蚂蚁行走 n 个城市节点，经过 N_c 迭代后，由表 1 可知，其具体的时间复杂度为 $T_1(n) \approx O(N_c(\lambda n)^2 m)$（$\lambda$ 为蚂蚁选择范围参数）。但由于本文涉及凸包的构建，还需将凸包有关的时间复杂度 $T_2(n)$ 进行分析与确定。

表 1　本文蚁群算法的时间复杂度分析

步骤	内容	时间复杂度
1)	初始化参数	$O(n^2+m)$
2)	设置算法禁忌表	$O(m)$
3)	蚂蚁遍历解	$O(\lceil \lambda n \rceil^2 m)$
4)	解的比较和路径更新量的计算	$O(\lceil \lambda n \rceil^2 m)$
5)	路径上信息素浓度的更新	$O(\lceil \lambda n \rceil^2)$
6)	判断算法是否达到停止条件若没有则转到步骤 2),	$O(\lceil \lambda n \rceil m)$
7)	输出执行结果	$O(1)$

本文凸包采用 Graham-Scan 算法进行构建，复杂度仅为 $O(t\log t)$（t 为参与凸包构建的节点数量)。ACADCG 算法中参与凸包构建的城市节点数为 $\lceil \lambda n_1 \rceil$（$n_1$ 为当前所有剩余城市节点数），因此构建一次凸包的复杂度为 $O(\lceil \lambda n_1 \rceil \log \lceil \lambda n_1 \rceil)$。一只蚂蚁在遍历 n 个城市节点时需构建 $n-1$ 次凸包，当蚂蚁遍历结束后，总的复杂度为 $O((n-1)\lceil \lambda n_1 \rceil \log \lceil \lambda n_1 \rceil)$。从而 ACADCG 中 m 只蚂蚁行走 n 个节点，经过 N_c 次迭代后凸包的复杂度为 $T_2(n) \approx O(N_c m\lambda n^2 \log \lceil \lambda n_1 \rceil)$。

从上述分析可知，ACADCG 算法总的时间复杂度为 $T(n)=T_1(n)+T_2(n)\approx O(N_c(\lambda n)^2 m)+ O(mN_c\lambda n^2 \cdot \log \lceil \lambda n_1 \rceil)$。

3.5.2 ACADCG 算法空间复杂度

假设算法所求解问题的规模为 n，则需一个 n 阶二维距离矩阵描述问题本身特征；为了表示有向图上的信息量，需用另外一个 n 阶二维距离矩阵来表示图上的信息素轨迹浓度；同时，在算法求解中，为保证 TSP 中城市不重复，需为每只蚂蚁设定一个 n 阶一维数组的禁忌表；此外，本文还对蚂蚁待选邻居节点进行了筛选，需额外设定一个 λn 阶一维数组的可选表 $allowed_k$；为了保存蚂蚁寻到的解，需为每只蚂蚁设立一个数组；为了便于更新轨迹，需设置每条边上的信息素更新量，其为一个二维数组；为了评价解的优劣，需定义中间变量等。通过对本文蚁群算法各步骤的综合分析，可得到其整个空间复杂度为 $S_1(n)=O(n^2)+O((1+\lambda)nm)$（$n$ 为城市节点数量，m 为蚂蚁数量）。由 $S_1(n)$ 可知，在数据存储上本文蚁群算法的空间复杂度非常简单。

同时，ACADCG 算法中存在凸包，还需对凸包的空间复杂度 $S_2(n)$ 进行分析与确定。本文的凸包主要涉及 2 个方面：城市选择策略和信息素更新策略。

在城市选择策略方面，利用每个当前城市节点与待选邻居节点构建的凸包区域范围之间的关系来影响蚂蚁节点的选择。在此过程中，要对形成交集的 2 个凸包的构建节点及凸包区域范围交集中的城市节点进行存储，存储的这些节点数量最多不会超过 λn，因此需建立 2 个 λn 阶一维数组（实际需建立 3 个，其中一个已在分析 $S_1(n)$ 时建立），则 m 只蚂蚁的空间复杂度为 $O(2\lambda nm)$；在信息素更新策略方面，需使用凸包角来对信息素进行调控，这需要在城市选择策略期间对已构建凸包的凸包角进行存储，需建立一个 n 阶的一维数组来存储凸包角信息，则 m 只蚂蚁的空间复杂度为 $O(nm)$。此外，还需建立一个 n 阶的一维数组，对与每个城市节点相距最近的城市距离进行存储，空间复杂度为 $O(n)$。综上，$S_2(n)=O((2\lambda m+m+1)n)$。

从上述分析可知，ACADCG 算法总的空间复杂度为 $S(n)=S_1(n)+S_2(n)=O(n^2)+O((3\lambda m+2m+1)n)$。

4 仿真实验结果与分析

美国 RAND 公司将城镇的实际地理位置抽象成坐标节点，从而来构建经典的 TSP 标准数据库。众多专家和学者基本都以该库作为基础来测试所研究的近似求解算法，因此本文将 ACADCG 算法应用于标准数据库里的不同经典 TSP 模型（Oliver30、Eil51、St70、Eil76）中求解，来验证本文信息素最大最小值限制策略的有效性以及算法的精确性、收敛的快速性与适用性。算法开发环境为：在配有 Intel 2.3 GHz，4 GB RAM 的 Windows 7 PC 机上采用 Java 语言和 Matlab 开发了 ACADCG 算法，并对此算法进行了仿真实验测试。

4.1 算法参数

参数的不同选择对算法性能有较大影响，但目前参数选取的方法和原则没有具体的理论依据，通常都是结合实际问题，根据领域经验或实验而定。本文算法结合文献[11,13,19,26]，针对 4 种 TSP 模型经过多次仿真实验，确定值相同的算法参数 α、β、ρ_1、P、λ、N_1 和 $N_{\max}$，如表 2 所示，值不同的算法参数 ρ、μ、m、λ_1、λ_2 和 λ_3，如表 3 所示。其中，若当前所有剩余城市多于 2 个，则 $\lceil \lambda n_1 \rceil \geqslant 2$，否则 $\lceil \lambda n_1 \rceil$ 取值为当前所有剩余城市数目，$\lambda \in (0, 0.25]$。此外，参数 λ 动态增量、ω 以及本文中已设定的其他相关常数也都是经过分析和多次仿真实验最终确定的。

参数确定的具体过程和筛选原则如下。

1) 依据文献[26]可知，α 和 β 相关性较大，因此把 α 和 β 联合进行实验调控并结合文献[26]来确定其值。调控准则为：先保持其他参数值不变，在文献[26]已给的范围内来调控 α 值，可得出多个较优值，大幅度缩小 α 取值范围，此时再将 α 值固定，来调控 β 值，同理可得出多个 β 较优值，此时再把 α 和 β 进行联合调控，最终确定 α 和 β 参数值。

表 2　本文算法在 4 种 TSP 模型中相同参数值设置

参数名称	参数值
信息素浓度权重值 α	1
启发式函数权重值 β	5
局部信息素挥发系数 ρ_1	0.04
概率增量 P	0.07
蚂蚁选择范围参数 λ 的初始值 N	0.1
全局最优解连续 N_1 次迭代值	30
最大迭代次数 $N_{\max}$	1 000

表 3　本文算法在 4 种 TSP 模型中不同参数值设置

TSP 算例	参数名称	参数值
Oliver30	全局信息素挥发系数 ρ	0.7
	蚂蚁个数 m	30
	延陷漂流因子 μ 初始值 N	1.4n
	权重因子 λ_1 、 λ_2 、 λ_3	1.1、1.0、0.8
Eil51	全局信息素挥发系数 ρ	0.8
	蚂蚁个数 m	51
	延陷漂流因子 μ 初始值 N	1.3n
	权重因子 λ_1 、 λ_2 、 λ_3	1.2、1.1、0.9
St70	全局信息素挥发系数 ρ	0.8
	蚂蚁个数 m	70
	延陷漂流因子 μ 初始值 N	1.2n
	权重因子 λ_1 、 λ_2 、 λ_3	1.4、1.3、1.0
Eil76	全局信息素挥发系数 ρ	0.8
	蚂蚁个数 m	76
	延陷漂流因子 μ 初始值 N	1.1n
	权重因子 λ_1 、 λ_2 、 λ_3	1.5、1.4、1.1

2) 根据文献[26]可知道蚂蚁数量 m 的选取影响算法迭代次数，因此把 m、$N_{\max}$、N_1、μ、ω 联合进行实验调控并结合文献[11,19,26]来确定其值，其中，N_1 和 μ 是本文所定义的。N_1 选值原则：N_1 是根据已定义的最大迭代次数，并统计算法实际迭代过程中所求的最优解有多少次连续迭代不发生变化，依据统计数据，同时结合 1)中调控准则，再进行多次仿真实验调整来最终确定 N_1 值。μ 值选取原则：文献[11]已给出相关参数值的具体筛选原则与过程，本文依据文献[11]中的筛选过程，针对不同 TSP 模型进行联合调控，来确定初始值。

3) 根据本文算法思想，P 和 λ 相关性较大，因此把 P 和 λ 联合进行实验调控并结合文献[13]来确定其值，其中，λ 是本文定义的。λ 的相关值选取原则：依据城市规模，并结合已形成的最优路径中某一城市相连的前后节点与该城市的距离值在所有可选邻居城市节点中的排行，同时考虑凸包的构建，此外，进行多次仿真实验调控来确定一个相对较佳的初始值与增量值。

4) 在信息素策略里面，ρ_1、ρ 与 λ_1、λ_2、λ_3 相关度很大，因此把它们联合进行实验调控并结合文献[11,19,26]来确定其值，其中，λ_1、λ_2、λ_3 是本文定义的。λ_1、λ_2、λ_3 选值原则为：首先对无权重因子的式(10)~式(12)进行计算，其中，参与运算的信息素值是不包含式(10)~式(12)算法运行后的值；接着根据运算值并结合路径边上的 $(1-\rho)\tau_{ij}(t)$ 值及 $\Delta\tau_{ij}^{\text{best}}$ 值对 $\Delta\tau$ 进行调控，从而大概确定 λ_1、λ_2、λ_3 这 3 个初始值的取值范围；然后结合随着蚂蚁访问城市增加，非优路径出现概率越大的情况，对所处不同时期的权重因子值进行不同程度的调整；最后针对不同的 TSP 模型进行多次仿真实验，来最终确定较为合适的权重因子值。

4.2　信息素最大最小值限制区间对算法的影响

选取 Eil51 作为仿真对象，对具有不同信息素最大最小值限制策略的 ACADCG 算法进行收敛速度的比较，结果如图 4 所示。图 4 中 3 条曲线分别为含有自身信息素最大最小值限制式的 ACADCG 算法、含有传统信息素最大最小值限制式的 ACADCG 算法（简称 T-ACADCG 算法）、含有固定信息素最大最小值限制的 ACADCG 算法（简称 F-ACADCG 算法）。其中，根据本文信息素的初始值以及信息素更新规则，设置固定信息素最大最小值限制范围为 $[0.000\,5, 0.1]$。

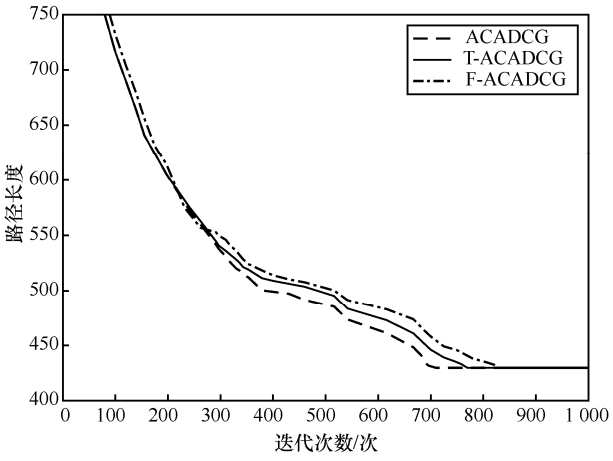


图 4　3 种不同信息素最大最小值限制区间算法收敛速度的比较

由图 4 所知，T-ACADCG 算法的总体收敛速度比 F-ACADCG 算法快，而 ACADCG 算法相比 T-ACADCG 算法其收敛性又进一步提升。F-ACADCG 算法和 T-ACADCG 算法分别在 830 多次和 760 多次收敛到最优解，而 ACADCG 算法在 700 次左右就已经收敛到最优解了。

T-ACADCG 算法中的信息素最大最小值限制范围的上下限会随着解的改变而变化，相比于固定上下限的 F-ACADCG 算法收敛速度要快。但 T-ACADCG 算法中的上下限有时会减慢算法向最优解逼近，而本文中信息素最大最小值限制范围的上下限相比于 T-ACADCG 算法中的上下限会在算法迭代中期，随着迭代的推移平稳增加，能适当减少较优路径上信息素递增幅度的限制，增加算法逼近于最优解的概率；在算法后期，相比于 T-ACADCG 算法中的上下限其倍数的增量非常缓慢直至不变，本文中信息素最大最小值限制范围的上下限随着算法的迭代，避免不同路径上信息素差别过大；但在算法前期，本文中信息素最大最小值限制范围的上下限与 T-ACADCG 算法中的上下限变化相似，以抑制算法前期解的多样性。将上述不同策略的 ACADCG 算法拓展到 4.4 节中的 Oliver30、St70、Eil76 中进行仿真实验，变化趋势与图 4 相似，在此不再重复绘图与分析。

4.3　ACADCG 算法对收敛速度的影响

仍以 Eil51 作为详细分析对象来测试本文算法的收敛速度。将蚁群系统（ACS, ant colony system）、最大最小蚂蚁系统（MMAS, max-min ant system）、基于方向信息素协调的蚁群算法[11]（AADC, ant algorithm based on direction-coordinating）和本文 ACADCG 算法进行寻优结果比较，实验结果如图 5 所示。

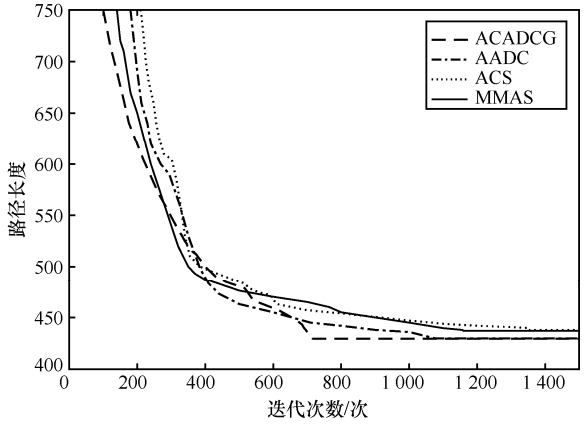


图 5　4 种算法收敛速度的比较

由图 5 可知，ACS 算法、MMAS 算法、AADC 算法在迭代 400 次之后开始逐步收敛，分别在迭代 1 400 次左右、1 200 次左右、1 000 次左右才能最终收敛；ACADCG 算法在迭代 700 多次，就收敛到 429.18，逼近于理论最优解 426。因此，ACADCG 算法收敛速度有显著提高，求解精度也有进一步的改进。将 4.4 节中的 Oliver30、St70、Eil76 作为仿真对象测试上述不同算法，变化趋势与图 5 相似，在此也不再重复绘图与分析。

针对收敛速度的问题，本文以 Oliver30 和 Eil51 为例，将 ACADCG 算法与 ACS 算法、MMAS 算法、AADC 算法进行仿真实验比较，结果如表 4 所示。表 4 中结果均为 30 次实验所得。为避免赘述，St70、Eil76 的仿真实验结果将在 4.4 节中详细列出。

表 4　ACADCG 算法与其他算法收敛速度上下限对比

TSP 问题		Oliver30	Eil51
ACS 算法	迭代最小值	612	897
	迭代最大值	1502	1712
	迭代平均值	1011	1397
	平均解	429.16	441.23
MMAS 算法	迭代最小值	407	781
	迭代最大值	1124	1501
	迭代平均值	712	1238
	平均解	427.18	439.81
AADC 算法	迭代最小值	191	450
	迭代最大值	531	1703
	迭代平均值	306	1078
	平均解	424.64	437.20
ACADCG 算法	迭代最小值	187	446
	迭代最大值	473	923
	迭代平均值	274	716
	平均解	423.89	437.03

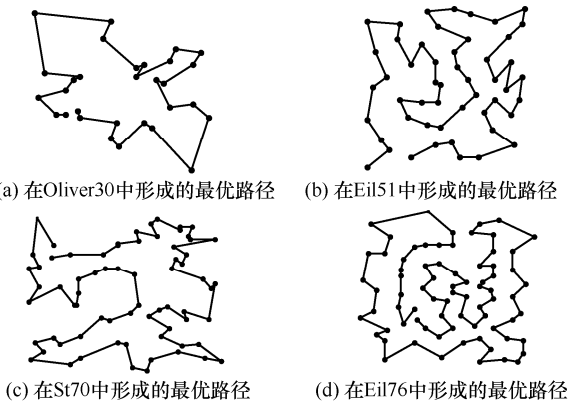
由表 4 可知，ACADCG 算法在 Oliver30 问题中，其迭代次数和所求解的平均值要优于 ACS 算法和 MMAS 算法，相比于 AADC 算法也有进一步提升；在 Eil51 问题中，ACADCG 算法的迭代平均值为 716，平均解为 437.03，迭代平均值分别约为 ACS 算法的 $\frac{1}{2}$、MMAS 算法的 $\frac{7}{12}$、AADC 算法的 $\frac{7}{10}$，而所求平均解的误差比 ACS 算法减少 1.1%，比 MMAS 算法减少 0.7%，与 AADC 算法接近。因

此，ACADCG 算法除能得到较优解外，收敛速度也有显著提高，从而实现 ACADCG 算法对 TSP 问题的快速求解。

求解快速性的主要原因是：ACADCG 算法中的蚂蚁是从当前所有剩余城市中筛选出若干个城市作为待选邻居城市集合，缩小了搜索空间。此外，待选邻居城市集合的动态扩充有助算法在陷入局部最优时跳出局部最优，加快算法的收敛速度；当蚂蚁进行城市选择时，GCAPOP 方法增加了凸包区域范围交集中城市被选概率，使蚂蚁优先访问最近已遍历城市附近未选择的城市节点，增加蚂蚁向无交叉路径解逼近的概率，推动 ACADCG 算法向全局最优解逼近；当所有蚂蚁完成遍历后，含有 GSAPOP 方法的信息素更新策略能削弱非优路径及其周边路径上的信息素，提高后继蚂蚁向较优路径集合逼近的概率，加快算法寻优过程中的收敛速度；同时，本文的信息素最大最小值限制策略能对算法进行适当的干预，进一步减少了算法的迭代次数。

4.4　ACADCG 算法在多种模型中综合性能分析

将 ACADCG 算法与其他蚁群算法分别应用于 4 个典型 TSP 模型中进行误差率、收敛速度、适用性等综合性能仿真实验，结果如表 5 所示。此外，ACADCG 算法在 4 个模型上求解获得的最优路径如图 6 所示。



(a) 在Oliver30中形成的最优路径　(b) 在Eil51中形成的最优路径

(c) 在St70中形成的最优路径　(d) 在Eil76中形成的最优路径

图 6　ACADCG 算法最优路径遍历结果

由表 5 可知，ACADCG 算法在 Oliver30 问题规模中，最优路径长度为 423.14，与 ACS 算法的 425.52、MMAS 算法的 424.86 相比，误差分别减小了 0.5%、0.4%，同时，ACADCG 算法的迭代次数仅为 ACS 算法的$\frac{1}{4}$、MMAS 算法的$\frac{1}{3}$；而在 Eil51 问题中，ACADCG 算法所求的最优路径长度误差分别比 ACS 算法减小 2.2%、比 MMAS 算法减少 1.7%，平均迭代次数也分别减少$\frac{1}{2}$、$\frac{1}{3}$。另一方面，ACADCG 算法在 Oliver30 和 Eil51 问题中，误差与 AADC 算法接近，但迭代次数分别减少了三十多次、三百多次。此外随着城市规模的增

表 5　　**典型 TSP 问题求解对比模型**

TSP 算例	算法名称	最优路径长度	误差	最差路径长度	平均路径长度	平均迭代次数/次
Oliver30	ACS 算法	425.52	1.3%	446.87	429.16	1 011
	MMAS 算法	424.86	1.2%	442.34	427.18	712
	AADC 算法	423.91	0.9%	425.82	424.64	306
	ACADCG 算法	423.14	0.8%	425.26	423.89	274
Eil51	ACS 算法	438.74	3.0%	455.17	441.23	1 397
	MMAS 算法	436.63	2.5%	453.12	439.81	1 238
	AADC 算法	429.74	0.9%	449.82	437.20	1 078
	ACADCG 算法	429.18	0.8%	450.39	437.03	716
St70	ACS 算法	687.46	1.8%	700.95	697.37	1 276
	MMAS 算法	685.13	1.5%	698.89	694.74	1 112
	AADC 算法	682.31	1.1%	699.12	689.92	932
	ACADCG 算法	681.25	0.9%	698.27	689.16	619
Eil76	ACS 算法	555.61	3.3%	559.55	557.18	1 297
	MMAS 算法	552.26	2.7%	558.11	555.84	1 142
	AADC 算法	544.43	1.2%	557.43	550.21	970
	ACADCG 算法	543.41	1.0%	557.52	549.39	658

加，ACADCG 算法的误差不但比 ACS 算法和 MMAS 算法小，而且相比于 AADC 算法也有所减少，并且其迭代次数相比于 AADC 算法进一步减少了大约$\frac{1}{3}$。

ACADCG 算法引入了二维凸包，而二维凸包的构建与城市节点的位置相关，因此具有不同城市节点位置和城市数量的 TSP 模型，其求解误差可能不一样，但 ACADCG 算法的误差控制能力优于与其对比的算法且收敛速度较快，主要原因是：本文算法能有效避免蚂蚁在 TSP 模型中遍历时出现交叉路径，提高了算法的整体求解精度和收敛速度。同时，本节 4 个 TSP 模型上的仿真结果也体现了 ACADCG 算法具有较好的适用性。

5 结束语

针对蚁群算法搜索空间大、求解精度差、迭代轮数多的缺陷，本文基于二维凸包对蚁群算法进行了改进。通过仿真对比分析可知，改进后的蚁群算法全局搜索能力得到提高，收敛速度进一步加快，而且在此过程中，能够较好地克服算法过早收敛到局部解的缺点，同时能够得到更精确解，且具有较好的适用性。实验结果表明，本文所提出的改进是切实可行的。

然而本文算法还存在一些不足之处，首先蚁群算法的随机性和凸包构建的动态性导致了算法复杂度的不确定性；其次凸包的构建有一定的代价，但本文不把凸包构建作为影响算法的性能指标，因凸包构建的代价不影响算法的迭代次数。未来，将深入分析和探讨凸包构建，以便进一步提高算法的性能，并将算法应用于数据规模更大的问题上来进行求解。

参考文献：

[1] DORIGO M, GAMBARDELL L M. Ant colonies for the travelling salesman problem[J]. Bio Systems, 1997, 43(2): 73-81.

[2] MAEKAWA K, TAMAKI H, KITA H, et al. A method for the traveling salesman problem based on the genetic algorithm[J]. Transactions of the Society of Instrument & Control Engineers, 1995, 31(5): 598-605.

[3] WANG J, ERSOY O K, HE M, et al. Multi-offspring genetic algorithm and its application to the traveling salesman problem[J]. Applied Soft Computing, 2016, 43(3): 415-423.

[4] MURRAY A T, CHURCH R L. Applying simulated annealing to location-planning models[J]. Journal of Heuristics, 1996, 2(1): 31-53.

[5] YU D, JIA J. A neural network algorithm with optimized parameters and used to solve the TSP[J]. Acta Electronica Sinica, 1993, 21(7): 16-22.

[6] LIN Y, BIAZ Z, LIU X. Developing a dynamic neighborhood structure for an adaptive hybrid simulated annealing – tabu search algorithm to solve the symmetrical traveling salesman problem[J]. Applied Soft Computing, 2016, 49(9):937-952.

[7] LIU Z, LI X, WANG H. Improvement of self-adaptive ant colony algorithm based on cloud model[J]. Computer Engineering and Applications, 2016, 52(19): 68-71.

[8] PAN G, LI K, OUYANG A, et al. Hybrid immune algorithm based on greedy algorithm and delete-cross operator for solving TSP[J]. Soft Computing, 2016, 20(2):555-566.

[9] DORIGO M, MANIEZZO V, COLORNI A, et al. Ant system: optimization by a colony of cooperating agents[J]. IEEE Transactions on Systems Man and Cybernetics-Part B, 1996, 26(1): 29-41.

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

[11] 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(5): 782-786.

[12] 吴华锋, 陈信强, 毛奇凰, 等. 基于自然选择策略的蚁群算法求解 TSP 问题[J]. 通信学报, 2013, 34(4): 165-170.
WU H F, CHEN X Q, MAO Q F, et al. Improved ant colony algorithm based on natural selection strategy for solving TSP problem[J]. Journal on Communications, 2013(4): 165-170.

[13] 林建兵, 陈智雄, 姚国祥. 一种基于域密度的蚁群系统(AS)改进算法及结果解析[J]. 武汉大学学报(工学版), 2016, 49(4): 627-634.
LIN J B, CHEN Z X, YAO G X. An improved AS algorithm and result analyzing based on domain and its density[J]. Engineering Journal of Wuhan University, 2016, 49(4): 627-634.

[14] 马学森, 曹政, 韩江洪, 等. 改进蚁群算法的无线传感器网络路由优化与路径恢复算法[J]. 电子测量与仪器学报, 2015, 29(9): 1320-1327.
MA X S, CAO Z, HAN J H, et al. Routing optimization and path recovery algorithm in wireless sensor network based on improved ant colony algorithm[J]. Journal of Electronic Measurement and Instrument, 2015, 29(9): 1320-1327.

[15] 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.
SUN L J, WANG L J, WANG R C, Research of using an improved ant colony algorithm to solve TSP[J]. Journal on Communications, 2004, 25(10): 111-116.

[16] BU Y, LI T Q, ZHANG Q. A weighted max-min ant colony algorithm for TSP instances[J]. IEICE Transactions on Fundamentals of Electronics Communications & Computer Sciences, 2015(3): 894-897.

[17] 耿志强, 邱大洪, 韩永明. 基于混沌的 MMAS 算法及其在旅行商问题中的应用[J]. 计算机工程, 2016, 42(3): 192-197.
GENG Z Q, QIU D H, HAN Y M. Max-min ant system algorithm based on chaos and its application in TSP[J]. Computer Engineering, 2016, 42(3): 192-197.

[18] WANG Y. Hybrid max-min ant system with four vertices and three lines inequality for traveling salesman problem[J]. Soft Computing, 2015, 19(3): 585-596.

[19] 张层. 基于二维凸包的改进蚁群算法求解 TSP 问题[D]. 广州: 华南理工大学, 2013.
ZHANG C. An improved ant colony algorithm based on two-dimensional convex hull for TSP[D]. Guangzhou: South China University of Technology, 2013.

[20] 党小超, 李小艳. 基于图论的 WSN 节点定位路径规划[J]. 计算机工程, 2012, 38(11): 100-103.
DANG X C, LI X Y. WSN node localization path planning based on graph theory[J]. Computer Engineering, 2012, 38(11): 100-103.

[21] PRAGYA, DUTTA M, PRATYUSH. TSP solution using dimensional ant colony optimization[C]//International Conference on Advanced Computing & Communication Technologies. 2015: 506-512.

[22] QIN H, ZHOU S, HUO L, et al. A new ant colony algorithm based on dynamic local search for TSP[C]//International Conference on Communication Systems and Network Technologies. 2015: 913-917.

[23] GU S, ZHANG X. An improved ant colony algorithm with soldier ants[C]//2015 11th International Conference on Natural Computation (ICNC). 2016: 205-209.

[24] LUO W, LIN D, FENG X X. An improved ant colony optimization and its application on TSP problem[C]//2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData). 2017: 136-141.

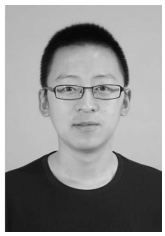
[25] 盛国军, 温涛, 郭权, 等. 基于改进蚁群算法的可信服务发现[J]. 通信学报, 2013, 34(10): 37-48.
SHENG G J, WEN T, GUO Q, et al. Trustworthy service discovery based on a modified ant colony algorithm[J]. Journal on Communications, 2013, 34(10): 37-48.

[26] 詹士昌, 徐婕, 吴俊. 蚁群算法中有关算法参数的最优选择[J]. 科技通报, 2003, 19(5): 381-386.
ZHAN S C, XU J, WU J. The optimal selection on the parameters of the ant colony algorithm[J]. Bulletin of Science and Technology, 2003, 19(5): 381-386.

[作者简介]

马学森（1976−），男，安徽庐江人，合肥工业大学副教授、硕士生导师，主要研究方向为无线传感器网络、嵌入式系统、大数据处理、网络与信息安全。

宫帅（1993−），男，安徽滁州人，合肥工业大学硕士生，主要研究方向为无线传感器网络、物联网、网络与信息安全。



朱建（1992−），男，安徽五河人，合肥工业大学硕士生，主要研究方向为高可靠性嵌入式系统、无线传感器网络、物联网。

唐昊（1972−），男，安徽庐江人，合肥工业大学教授、博士生导师，主要研究方向为离散事件动态系统、随机决策与优化理论、强化学习等智能优化与控制方法、自动化生产线、物联网和微网等系统优化设计与控制技术。