Journal of System Simulation

Volume 36 | Issue 1 Article 18

1-20-2024

# Heterogeneous Multi-ant Colony Algorithm Combining Competitive Interaction Strategy and Eliminatingreconstructing Mechanism

Chen Feng
*Shanghai University of Engineering Science, Shanghai 201620, China*, 845346965@qq.com

Xiaoming You
*Shanghai University of Engineering Science, Shanghai 201620, China*, yxm6301@163.com

Sheng Liu
*Shanghai University of Engineering Science, Shanghai 201620, China*

# Heterogeneous Multi-ant Colony Algorithm Combining Competitive Interaction Strategy and Eliminatingreconstructing Mechanism


## Abstract

Abstract: The traditional ant colony algorithm has many problems in convergence and diversity when solving the traveling salesman problem (TSP). Therefore, this paper proposes a heterogeneous multi-ant colony algorithm that combines the competitive interaction strategy and the eliminating-reconstructing mechanism (CEACO) to overcome these shortcomings. Firstly, the algorithm uses a competitive interaction strategy, which adjusts the interaction period adaptively according to the Hamming distance of different groups in different periods. Competition coefficients are adopted to differentiate matching interaction objects for interaction. The matched objects interact with each other through the optimal solution and pheromone matrix. This mechanism achieves a balance between algorithm convergence speed and diversity. Secondly, the algorithm uses the eliminating-reconstructing mechanism, which periodically eliminates and reconstructs the poor-searching populations to improve the solution accuracy of the algorithm.Finally, several groups of simulation experiments prove that the algorithm outperforms other methods in improving the solution accuracy and convergence speed.



## Keywords

ant colony optimization(ACO), heterogeneous multiple population, competitive interaction, eliminating-reconstructing, traveling salesman problem

第 36 卷第 1 期 系统仿真学报© Vol. 36 No. 1
2024 年 1 月 Journal of System Simulation Jan. 2024

# 结合竞争交互策略和淘汰重组机制的异构多蚁群算法

冯晨，游晓明*，刘升
(上海工程技术大学，上海 201620)

**摘要：**针对传统的蚁群算法在解决旅行商问题时(traveling salesman problem, TSP)存在着收敛速度慢、容易陷入局部最优等问题，提出了一种结合竞争交互策略和淘汰重组机制的异构多蚁群算法。建立一个异构多种群系统，算法采用竞争交互策略，根据不同时期各种群的汉明距离来自适应的调节交互周期；并利用竞争系数来差异化匹配交互对象，经过匹配后的交互对象之间通过最优解和信息素矩阵进行交互，通过该机制实现了算法收敛速度和多样性的平衡。算法采用了淘汰重组机制，会定期对寻优能力差的种群进行淘汰与重组，以加快算法的求解精度。采用多组不同规模的TSP算例进行仿真实验，结果表明，该算法在提高求解精度和收敛速度方面表现更优。

**关键词：**蚁群算法；异构多种群；竞争交互；淘汰重组；旅行商问题
中图分类号：TP18 文献标志码：A 文章编号：1004-731X(2024)01-0232-17
DOI: 10.16182/j.issn1004731x.joss.22-1009
**引用格式:** 冯晨, 游晓明, 刘升. 结合竞争交互策略和淘汰重组机制的异构多蚁群算法[J]. 系统仿真学报, 2024, 36(1): 232-248.
**Reference format:** Feng Chen, You Xiaoming, Liu Sheng. Heterogeneous Multi-ant Colony Algorithm Combining Competitive Interaction Strategy and Eliminating-reconstructing Mechanism[J]. Journal of System Simulation, 2024, 36 (1): 232-248.

## Heterogeneous Multi-ant Colony Algorithm Combining Competitive Interaction Strategy and Eliminating-reconstructing Mechanism

*Feng Chen*, *You Xiaoming**, *Liu Sheng*
(Shanghai University of Engineering Science, Shanghai 201620, China)

**Abstract:** The traditional ant colony algorithm has many problems in convergence and diversity when solving the traveling salesman problem (TSP). Therefore, this paper proposes a heterogeneous multi-ant colony algorithm that combines the competitive interaction strategy and the eliminating-reconstructing mechanism (CEACO) to overcome these shortcomings. *Firstly, the algorithm uses a competitive interaction strategy, which adjusts the interaction period adaptively according to the Hamming distance of different groups in different periods. Competition coefficients are adopted to differentiate matching interaction objects for interaction. The matched objects interact with each other through the optimal solution and pheromone matrix. This mechanism achieves a balance between algorithm convergence speed and diversity. Secondly, the algorithm uses the eliminating-reconstructing mechanism, which periodically eliminates and reconstructs the poor-searching populations to improve the solution accuracy of the algorithm.* Finally, several groups of simulation experiments prove that the algorithm outperforms other methods in improving the solution accuracy and convergence speed.

收稿日期：2022-08-26 修回日期：2022-10-17
基金项目：国家自然科学基金(61673258，61075115)；上海市自然科学基金(19ZR1421600)
第一作者：冯晨(1999-)，男，硕士生，研究方向为智能算法、移动机器人路径规划。E-mail：845346965@qq.com
通讯作者：游晓明(1963-)，女，硕导，博士，研究方向为群智能系统、进化算法。E-mail：yxm6301@163.com

**Keywords:** ant colony optimization(ACO); heterogeneous multiple population; competitive interaction; eliminating-reconstructing; traveling salesman problem

## 0 引言

20 世纪 90 年代，意大利的 Dorigo，Maniezzo 等通过观察自然界中蚂蚁觅食的行为，提出了蚁群算法(ant colony optimization, ACO)[1-2]，主要用来解决旅行商问题。后期一些专家学者将该算法的应用扩展到车间调度[3]、数据挖掘[4]、网络路由[5]等领域中。

为了解决传统蚁群算法求解精度低，收敛速度慢的问题，Dorigo 于 1996 年在蚂蚁系统(ant system, AS)算法的基础上加以改进，提出了蚁群系统(ant colony system, ACS)[6]，该算法通过强化最优解对蚁群的指导作用，提升了算法的收敛速度，但也导致了算法容易陷入局部最优。文献[7]提出了最大最小蚂蚁系统(max-min ant system, MMAS)，通过给信息素设置上下界限制了信息素的挥发，在一定程度上提高了算法的搜索效率，改善了算法容易陷入局部最优的问题。

近些年来为了提升蚁群算法的收敛速度与求解精度，学者们提出了一些改进策略。文献[8]提出了一种基于进化经验引导信息素更新策略的多目标蚁群优化算法，该算法提出了一种历史经验引导的信息素更新方法，根据搜索历史自适应地选择合适的信息素更新方法，提升了算法的收敛速度。文献[9]提出了一种自适应参数调整的机制，在蚁群算法运行的过程中自适应的调整蚁群的控制参数，提高了算法针对不同规模问题的适应性。文献[10]通过增加较优的路径上的信息素浓度，从而减少了无用的搜索，提高了算法的收敛速度。文献[11]利用 K-means 聚类算法来生成城市关联矩阵，并加入了信息素二次更新的机制，通过这两种方法影响了蚂蚁选择城市的概率，从而提升了算法的多样性。上述单种群算法都是为了提升算法某一方面的性能而做出的改善，这种改善具有一定局限性，通常还会因此削弱算法其他方面的性能。例如，加快算法的收敛速度会降低求解的精度，而提高算法的求解精度又会导致算法收敛时间的加长。

为了能够平衡算法的收敛速度和多样性，有学者就提出了多策略结合的多蚁群算法。文献[12]提出了多蚁群算法的概念，采用两种蚁群算法来解决具有时间窗的车辆调度问题。文献[13]提出了一种多蚁群局部搜索算法，将改进的蚁群算法与 4 种局部搜索因子相结合，通过比较和交换每个群体的全局最优解来进行蚁群间的交流，提高了算法的求解精度和收敛速度。文献[14]提出了一种定向邻域扩展策略，提升了算法的搜索效率，同时结合双层精英蚁策略加大最佳路径的信息素含量，防止算法陷入局部最优。文献[15]将算法中的两个异构蚁群通过交换信息素信息来进行交流，提升了算法的多样性。文献[16]提出了一种基于相似度的自适应异类多种群蚁群算法，通过衡量各个子蚁群的相似度来选择最互补的蚁群进行信息交换，该算法增强了解的多样性，但该算法缺少提升收敛速度的策略，收敛速度还需要进一步提高。综上所述，目前虽然存在着一些多蚁群算法在寻求收敛速度和多样性之间的平衡，但随着问题规模的变化，求解的精度都有所下降；尤其在面临大规模问题时，算法的表现还有待改善。针对大规模 TSP 测试集，本文提出了一种结合竞争交互策略和淘汰重组机制的异构多蚁群算法(CEACO)。本文的主要贡献和创新点如下：

(1) 提出了一种竞争交互策略。该策略会根据不同时期各种群汉明距离的大小来自适应的调节交互周期；并通过衡量各种群的竞争系数，差异化的匹配交互对象；交互时会采用交换最优解和信息素矩阵两种交互方式，在前期通过最优解的交互来使算法达到快速收敛的目的，在后期通过




DOI: 10.16182/j.issn1004731x.joss.22-1009

交互信息素矩阵来达到提高系统多样性的目的。最终实现了算法收敛速度和多样性的平衡。

(2) 提出了淘汰重组机制。该机制会在交互周期的基础上确定淘汰周期，依据各种群在淘汰周期前后竞争潜力的差异变化确定出需要淘汰的种群；剩余种群会通过信息素矩阵的融合重组出一个新的子种群来代替淘汰种群；通过淘汰重组机制，有效的提高了算法的求解精度。仿真实验结果表明，本文提出的改进算法相比于传统蚁群算法及其他改进算法在求解TSP上能取得更好的实验结果，有效的平衡了算法的收敛速度与求解精度。

# 1 相关工作

## 1.1 蚁群算法(ACS)的工作原理

在蚁群算法中，蚂蚁遵循伪随机比例规则来选择下一个城市$j$：

$$j=\begin{cases}\arg\max\limits_{j\in N_k}\{\tau_{ij}\cdot\eta_{ij}^{\beta}\}, & q\leqslant q_0\\ J, & q>q_0\end{cases} \tag{1}$$

式中：$q$为在区间[0, 1]之间的一个随机数；$q_0$为在区间[0, 1]之间的一个可调节的参数。$J$按照式(2)状态转移公式寻找下一个城市：

$$P_{ij}^{k}=\begin{cases}\dfrac{\tau_{ij}^{\alpha}\cdot\eta_{ij}^{\beta}}{\sum\limits_{j\in N_k}\tau_{ij}^{\alpha}\cdot\eta_{ij}^{\beta}}, & j\in N_k\\ 0, & \text{else}\end{cases} \tag{2}$$

式中：$\tau_{ij}$表示城市$i$到城市$j$路径上的信息素；$\eta_{ij}$表示蚂蚁从城市$i$到城市$j$的期望程度，$\eta_{ij}=1/d_{ij}$，$d_{ij}$为城市$i$到城市$j$的距离；$\alpha$为信息素影响因子；$\beta$为启发式影响因子；$N_k$为蚂蚁$k$还未访问过的城市的集合。

当蚁群中的所有蚂蚁完成一次迭代之后，当前路径最优的蚂蚁将会进行一次全局信息素更新，全局信息素更新公式为

$$\tau_{ij}=(1-\rho)\cdot\tau_{ij}+\rho\cdot\Delta\tau_{ij}^{\text{bs}} \tag{3}$$

式中：$\rho$为信息素蒸发率；$\Delta\tau_{ij}^{\text{bs}}$为信息素增量，$\Delta\tau_{ij}^{\text{bs}}=1/L_{\text{gb}}$；$L_{\text{gb}}$为全局最优路径。通过全局更新，会加大最优路径上的信息素浓度，起到一个正反馈的作用，提高算法的收敛速度。

在蚁群中，除了每次迭代后的全局更新之外，当蚂蚁从城市$i$到城市$j$时，还要进行局部信息素更新，信息素更新公式为

$$\tau_{ij}=(1-\zeta)\cdot\tau_{ij}+\zeta\cdot\tau_0 \tag{4}$$

式中：$\zeta$为局部信息素蒸发率；$\tau_0$为信息素的初始值。通过信息素的局部更新，可以防止算法过早陷入局部最优，增加算法的多样性。

## 1.2 最大最小蚁群算法(MMAS)的工作原理

为了解决传统蚁群算法容易陷入局部最优的缺点，Stutzle在AS(ant system)的基础上提出了一个改进的最大最小蚁群算法。在MMAS运行的早期，通过全局最优路径和迭代最优路径交替更新信息素，信息素更新公式如式(3)所示。同时，这个算法的基本思想是将路径上的信息素设定在一个区间$[\tau_{\min},\tau_{\max}]$内，$\tau_{\max}$的限制使路径上的信息素不会过高，避免算法陷入局部最优；$\tau_{\min}$使每条路径上都始终存在信息素，因此每条路径都存在被探索的可能，提高了算法的多样性。其中，$\tau_{\min}$和$\tau_{\max}$的计算公式为

$$\tau_{\min}=\frac{\tau_{\max}}{2n} \tag{5}$$

$$\tau_{\max}=(1-\rho)\cdot\left(\frac{1}{L_{\text{gb}}}\right) \tag{6}$$

式中：$n$为城市数目；$\rho$为信息素蒸发率；$L_{\text{gb}}$为全局最优路径。如果$\tau_{ij}>\tau_{\max}$，则$\tau_{ij}=\tau_{\max}$；如果$\tau_{ij}<\tau_{\min}$，则$\tau_{ij}=\tau_{\min}$。这样就将信息素水平限制在了区间$[\tau_{\min},\tau_{\max}]$内。

## 1.3 汉明距离

在信号处理中，汉明距离是用来测试两串等长字符差异量的方法，它表示的两字符串中对应位置中不相同字符的个数；在本文中汉明距离用来度量种群中解的多样性。例如，现有两个字符串$a=[1,3,5,7,9,11]$和$b=[1,3,7,5,11,9]$，那么$a$和$b$之间的汉明距离为4，因为字符串$a$和$b$中的第3、



4、5、6位不相同。假设蚂蚁$m_1$和$m_2$之间的汉明距离为$HM(m_1,m_2)$。为了计算某个种群的总汉明距离，必须计算出种群中每个蚂蚁个体之间的汉明距离。假设种群中的蚂蚁数为$n$，城市规模为$citynum$，个体$m_i$和$m_j$的汉明距离为$HM(m_i,m_j)$，则可以计算第$k$个种群的平均汉明距离：

$$HM_k=\frac{2\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}HM(m_i,m_j)}{n(n-1)\cdot citynum} \tag{7}$$

# 2 改进算法

本文基于多种群蚁群算法，在搜索中主要采用两种蚁群系统：最大最小蚂蚁系统(MMAS)和蚁群系统(ACS)。两种算法的侧重点不同，最大最小蚂蚁系统在搜索过程中能够加快收敛速度，蚁群系统在搜索过程中可以较好的维持解的多样性。算法在运行过程中各种群会进行信息的交互和种群的淘汰重组，以此提升算法的性能。

## 2.1 竞争系数Q

在种群迭代的过程中，需要一个标准来衡量此时系统性能的好坏，本文综合解的质量和种群的多样性，提出了竞争系数$Q$，其计算公式为

$$Q_i=(1-\gamma)\cdot\frac{HM_i}{HM_{\mathrm{B}}}+\gamma\cdot\frac{L_B}{L_i} \tag{8}$$

式中：$\gamma$为综合判别系数；$HM_i$为该种群当前的汉明距离；$HM_{\mathrm{B}}$为整个蚁群系统运行到当前迭代次数时汉明距离的最大值；$L_i$为种该群当前的最优解；$L_B$为整个蚁群系统运行到当前迭代次数时的最优解。在算法运行的初期，路径上的信息素较为分散，各种群的多样性都维持在一个较高的水平，种群间的$HM_i$相差很小，此时竞争系数$Q_i$的大小主要取决于解的质量，较优解可以在搜索时起到引导作用，因此解的质量越好的种群竞争力也就越强。在算法运行的中后期，部分种群的解可能已经收敛，此时想要进一步提升解的质量就需要蚂蚁去探索其他解，此时竞争系数$Q_i$的大小主要取决于种群的多样性$HM_i$，多样性越好的种群越有可能探索出更优的解，其竞争力也就越强。因此通过竞争系数$Q_i$可以衡量出在迭代中某个种群相对于整个蚁群系统的综合性能指标。

## 2.2 竞争交互策略

在多种群系统运行的过程中，种群之间需要定期进行信息交互，这样才能保证搜索的效率。首先要确定的是交互周期，当交互周期比较大时，种群的搜索容易陷入停滞，交互的作用减弱；当交互周期比较小时，种群之间交互频繁，容易产生同质化现象，降低搜索效率。其次，交互对象的选取也很重要，选择合适的交流对象可以使得整个系统的性能都得到提升。最后还要确定交互的内容，可以选择交换最优解或者信息素矩阵，较优种群的信息可以引导较差种群进行搜索，提高解的质量，而较差种群的信息也可以提高较优种群的多样性，达到相互改善的目的。本章中提出了一个自适应的竞争交互策略。

### 2.2.1 自适应交互周期的确定

通过衡量种群当前的迭代次数与多样性的大小，来计算此时交流周期的大小：

$$\overline{HM}_{iter}=\sum_{i=1}^{m}HM_{iter}^{i} \tag{9}$$

$$\omega=D\cdot(1-\frac{\left|\overline{HM}_{iter}-HM_{\mathrm{b}}\right|}{2}+\frac{iter_{\max}-iter}{4\cdot iter_{\max}}) \tag{10}$$

式中：$\overline{HM}_{iter}$为迭代次数为$iter$时的平均汉明距离；$HM_{iter}^{i}$为第$i$个种群在迭代次数为$iter$时的汉明距离；$m$为种群个数；$D$为常数；$HM_{\mathrm{b}}$为当前汉明距离的最大值；$iter$为当前迭代次数；$iter_{\max}$为最大迭代次数。

在算法运行的前期，种群内部各路径上的信息素比较分散，各种群有较大的机会去探索新的解，应该适当延长交互周期；在系统运行的后期，路径上的信息素比较集中，此时很容易陷入局部最优，应该缩短交互周期，促进种群之间的交流；同时，汉明距离作为种群多样性的衡量指标，表征了当前路径的相似程度，当汉明距离比较小时，说明当前的路径相似度比较高，多个蚂蚁在重复




DOI: 10.16182/j.issn1004731x.joss.22-1009

的搜索相同的路径片段，此时算法可能已经陷入局部最优，应该缩短交互周期，加强与较优种群的交流，以平衡当前的信息素矩阵，降低对重复路径的搜索，提高种群的搜索效率。相反的，当汉明距离较大时，应该延长交互周期。因此本文综合迭代周期和多样性，给出了一个可以依据种群当前状态自适应变化的交互周期。

### 2.2.2 交互对象的选择

在种群间进行信息交流时，要选择一个合适的对象进行交流，本文中根据竞争系数 $Q$ 的大小来确定交互的对象。如图 1 所示，假设系统中有 $m$ 个种群，种群之间按照 $Q$ 值的大小进行排序，之后按大小顺序 $Q_1$ 和 $Q_m$ 进行交流，$Q_2$ 和 $Q_{m-1}$ 进行交流，依次进行下去，直至所有的种群都完成信息交换。

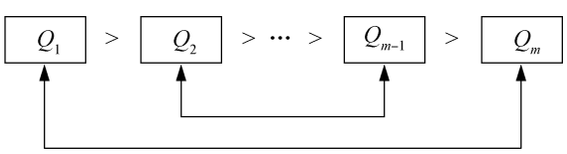



图 1 交互对象的选取
Fig. 1 Selection of interaction objects

若想改善系统整体的性能，对最差种群的改善就变得尤为重要。采用上述交互规则可以保证较优种群与较差种群之间进行交互，尤其是最优种群和最差种群间的交互；性能较差种群在性能较优的种群的引导下，可以迅速提升自身的性能，进而提升总体的性能。

### 2.2.3 竞争交互的内容

本文在信息交流时采用了两种交互方式：第一种是最优解的交互，其主要目的是引导较差种群加快收敛；第二种是交流信息素矩阵，其主要目的是增加种群解的多样性，使种群有更多的机会去探索新的解。在系统运行的前期，需要各个种群的解快速收敛到较优解附近，以加快后期的搜索效率。因此本文在第一次淘汰操作之前采取最优解交互的策略：

$$Lbest_i \rightleftharpoons Lbest_j \tag{11}$$

式中：$Lbest_i$ 为第 $i$ 个种群的最优解；$Lbest_j$ 为第 $j$ 个种群的最优解。在交互完成之后，种群在更新信息素时会根据交互得到的解进行更新。此时较优种群的解会对较差种群的搜索起到一个引导作用，加快了较差种群的收敛速度；而此时较差种群的解虽然对较优种群的收敛速度没有改善作用，但却拓宽了较优种群的搜索范围，增加了较优种群中除最优解之外其他路径上的信息素，提高了较优种群的多样性。

种群中的蚂蚁通过在路径上遗留的信息素进行交流，种群的信息素矩阵也就反映了当前种群的状态。在经过第一次淘汰操作之后，各种群的解基本上都收敛到较优解附近，此时采用信息素矩阵交互的策略。本文采用了一种信息素融合机制，将交流双方的信息素矩阵经过加权后融合在一起，这样不仅可以提升种群的多样性，而且也解决了单纯交换信息素矩阵容易引起种群之间的同质化的现象，其公式为

$$\boldsymbol{Ph}_i \leftarrow \boldsymbol{Ph}_i + \frac{Q_j}{Q_i + Q_j} \cdot \boldsymbol{Ph}_j \tag{12}$$

式中：$Q_i$ 为第 $i$ 个种群的竞争系数；$Q_j$ 为第 $j$ 个种群的竞争系数；$\boldsymbol{Ph}_i$ 为第 $i$ 个种群的信息素矩阵；$\boldsymbol{Ph}_j$ 为第 $j$ 个种群的信息素矩阵。

迭代初期，路径上的信息素分布较为平均，但是随着迭代的进行，路径上的信息素逐渐累积在几条较优的路径上，此时蚁群很容易在信息素的指导下陷入局部最优；此时根据竞争系数 $Q$ 对各种群进行排序，性能较好的种群会引导性能较差的种群进行搜索；其中交互的周期由算法根据各种群当前的状态自适应求解得出，在前期通过交互最优解提升了算法的收敛速度，而在后期通过交互信息素矩阵下提升了算法的多样性，最终实现了算法收敛速度和多样性之间的平衡。

## 2.3 淘汰重组机制

随着迭代的进行，算法的收敛速度与搜索能

力会逐渐降低甚至趋于停滞，造成这个现象的原因主要有两个：第一个原因是该种群的解和多样性的综合表现比较差，经过一段时间的迭代之后解的精度和多样性都很难提高甚至有所下降；第二个原因是该种群陷入局部最优，局部最优会导致该种群的蚂蚁长时间搜索同一条线路，这就会造成多样性的下降以及解的精度难以提升。由于最优解长时间得不到更新或者多样性的逐渐降低，会导致竞争系数 $Q$ 长时间保持同一个值甚至出现降低。算法的运行效率就会大幅度降低，此时可以利用淘汰重组机制来帮助算法提高运行效率，跳出局部最优。

### 2.3.1 竞争潜力

为了观察种群在迭代过程中的状态变化，需要定义一个变量来确定该种群在过去一段迭代周期中的表现。这里利用种群淘汰周期前后竞争系数的差值来表征种群的竞争潜力：

$$\Delta Q_i = Q_i^m - Q_i^n \tag{13}$$

式中：$\Delta Q_i$ 为第 $i$ 个种群的竞争潜力；$Q_i^m$ 为第 $i$ 个种群在第 $m$ 次迭代时的竞争系数；$Q_i^n$ 为第 $i$ 个种群在第 $n$ 次迭代时的竞争系数；$m-n$ 为当前的淘汰周期。竞争潜力表示在经历过一段时期的迭代后，该种群整体性能的变化。当竞争潜力为 0 或者出现负值时，表明在过去的这段迭代周期内种群性能没有提升甚至有所降低。当竞争潜力为正值时表明种群的性能相比于上一个迭代周期有所提升。

### 2.3.2 淘汰周期

淘汰的目的是为了保持种群的搜索效率，因此需要确定一个合适的淘汰周期。淘汰周期过大时，系统可能早已陷入局部最优或者长时间的在进行低效率的搜索，导致整体性能的降低；淘汰周期过小时，信息交互的作用就难以发挥，可能会导致竞争交互策略的优势没有体现出来种群就被淘汰，因此要确定一个合适的淘汰周期。淘汰周期是在前文得出的交互周期的基础上求得的，如图 2 所示，其中 $iter_1, iter_2, \cdots, iter_n$ 表示当前的迭代次数，$W_1, W_2, \cdots, W_n$ 表示交流周期，淘汰周期的计算公式为

$$EW_x = \sum_{i=p\cdot x-p+1}^{n\cdot x} W_i,\ i = 1, 2, \cdots, n \tag{14}$$

式中：$EW_x$ 为第 $x$ 个淘汰周期；$p$ 为一个可以调节的常数。

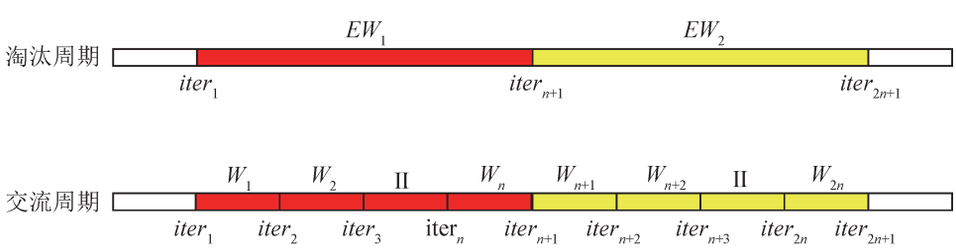



图 2 淘汰周期的确定

Fig. 2 Determination of elimination cycle

### 2.3.3 淘汰规则

为了提升算法的收敛速度和多样性，引入竞争交互策略，种群之间自适应的进行信息交互，在进行信息交互时会将表现最差的种群优先与表现最优种群进行交互，目的就是提升最差种群的搜索效率，使整个系统保持一个较高的搜索效率，因此最差种群经过交互后性能是否有提升，是关注的重点。在某一次淘汰周期中，首先计算每个种群的竞争潜力 $\Delta Q_i$，并得到整个系统的平均竞争潜力 $\overline{\Delta Q}$，之后记录下每次淘汰周期前竞争系数最小的种群的竞争潜力 $\Delta Q_w$。

淘汰规则如图 3 所示，当存在种群的 $\Delta Q < 0$ 时，说明该种群的性能在交互过后有所降低，因此淘汰 $\Delta Q$ 值最小的种群。当所有种群的 $\Delta Q > 0$



https://dc-china-simulation.researchcommons.org/journal/vol36/iss1/18
DOI: 10.16182/j.issn1004731x.joss.22-1009

时，说明所有种群的性能相比于淘汰周期之前都有所改善；此时要判断最差种群的性能改善情况，利用前文中求得的$\Delta Q_w$和$\overline{\Delta Q}$作比较；由于在交流时采用的是互补交流策略，表现最差的种群会和表现最优的种群进行信息交流，最差种群的性能也就有很大的提升空间；所以当$\Delta Q_w < \overline{\Delta Q}$时，说明最差种群的性能提升程度还没有达到各种群提升的平均值，提升程度很有限，所以淘汰最差种群。当$\Delta Q_w \geqslant \overline{\Delta Q}$时，此时表明最差种群的性能有了较大的提升，因此淘汰$\Delta Q$值最小的种群，也就是淘汰掉提升最小的种群。通过上述淘汰规则，可以保证整个系统一直以一个较高的搜索效率运行，提升了算法的求解精度。

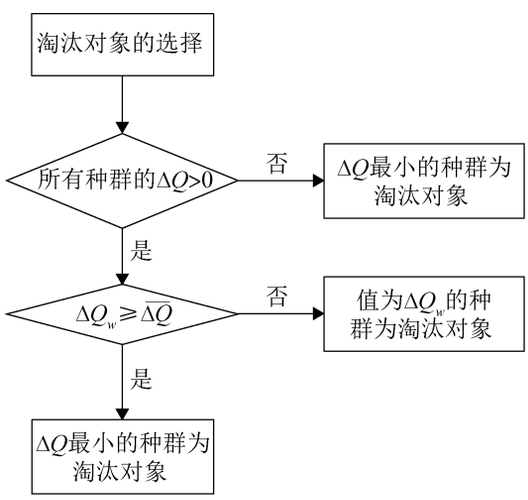



图3 淘汰规则流程图
Fig. 3 Flow of elimination rule

### 2.3.4 重组规则

当蚁群执行完淘汰操作之后，为了维持系统整体的性能，此时就会执行种群重组操作。系统中剩余的种群将信息素矩阵按比例融合重组成一个新的信息素矩阵，并将该矩阵作为新种群初始信息素矩阵，重组产生的新种群在该信息素矩阵的指导下继续进行搜索，重组的信息素矩阵的公式为

$$\boldsymbol{Ph}_{\text{new}} = \sum_{i=1}^{m-1} \frac{\boldsymbol{Q}_i}{\sum_{j=1}^{m-1} \boldsymbol{Q}_j} \cdot \boldsymbol{Ph}_i \tag{15}$$

式中：$m$为种群个数；$\boldsymbol{Q}_i$为第$i$个种群的竞争系数；$\boldsymbol{Ph}_i$为$i$种群的信息素矩阵。

通过淘汰操作，系统会定期淘汰掉性能表现不好或者陷入局部最优的种群；同时，在经历过淘汰操作之后系统会进行重组操作，系统会将剩余种群的信息素矩阵进行融合，重组产生的新种群会替代淘汰种群继续进行搜索。通过淘汰重组操作，算法可以对陷入停滞和效率较低的种群进行淘汰重组，有效提升了算法的搜索效率和求解精度。

## 2.4 CEACO 算法流程

本文提出的改进算法伪代码如下所示。

Algorithm 1： CEACO Algorithm for TSP

1 Initialize the pheromone and the parameters

2 Calculate the distance between cities

3 While termination condition is not satisfied do

4 Update pheromone for MMAS, ACS with Eq. (3), Eq.(4), Eq.(5), Eq.(6)

5 Calculate Hamming distance with Eq.(7)

6 Calculate competition coefficient with Eq.(8)

7 Calculate interaction period and eliminate period with Eq.(10), Eq.(14)

8 If NC = = eliminate period

9 Implement the elimination and reorganization strategy

10 End-If

11 If NC = = interaction period

12 Implement a population interact strategy with Eq.(11), Eq.(12)

13 End-If

14 NC = NC+1

15 End-While



### 2.5 算法的时间复杂度分析

表1展示了改进算法中各步骤的时间复杂度。在本算法中，迭代次数为$NC$，种群数目为$H$,蚂蚁数目为$m$，城市数目为$n$，所有子种群在计算机中是并行运行的。通过对上对上表各步骤时间复杂度的分析可以得出CEACO算法最大的时间复杂度为$O(NC{\cdot}m{\cdot}n^2)$。而传统算法ACS和MMAS的最大时间复杂度也为$O(NC{\cdot}m{\cdot}n^2)$，因此CEACO没有增加整体算法的时间复杂度。

表1 各步骤的时间复杂度
Table 1 Time complexity of each step

| 序号 | 内容 | 时间复杂度 |
|---|---|---|
| 1 | 初始化参数 | $O(n^2+m)$ |
| 2 | 蚂蚁构建路径 | $O(m{\cdot}n)$ |
| 3 | 信息素更新 | $O(n^2)$ |
| 4 | 计算汉明距离 | $O(n)$ |
| 5 | 计算竞争系数 | $O(m{\cdot}n)$ |
| 6 | 执行竞争交互策略 | $O(n^2)$ |
| 7 | 执行淘汰重组操作 | $O(n^2)$ |
| 8 | 输出最优路径 | $O(1)$ |

## 3 仿真实验

本文通过Matlab2019a的仿真环境对本算法进行性能测试，为了验证CEACO的有效性，通过TSPLIB中不同规模的城市库来对本算法进行验证。

### 3.1 参数设置

#### 3.1.1 基础参数设置

本文在ACS和MMAS的基础上，对算法加以改进，并进行了多次试验，分析每个参数设置和改变对算法产生的影响。经过对比分析发现，公共参数设置如表2所示，实验所得结果较好，并且使用相同参数与ACS算法和MMAS算法进行对比实验时，所得结果也更为合理。

表2中，$\alpha$ 为信息素影响因子，$\alpha$的值越大，信息素对蚂蚁搜索的导向作用越强，但$\alpha$值过大时容易使算法陷入局部最优；$\beta$为期望启发因子，$\beta$ 的值越大，蚂蚁在选择路径时越倾向于选择长度较短的路径，$\beta$值过大时会使算法的多样性降低；$\rho$ 和 $\xi$ 分别为全局信息素挥发速率和局部信息素挥发速率，其大小影响了算法的收敛速度；$T_{\max}$为最大迭代次数；$m$为蚂蚁数；$D$为计算自适应交互周期时的一个常数；$H$为种群的个数。

$H$的个数确定与配比实验结果如表3所示，实验选取kroB200作为试验集。通过表中的数据可以发现当ACS与MMAS种群的比例为1:1且种群数$H=4$时，所求得的解最好。这是由于两个种群在求解时的侧重点不同，ACS种群在求解时收敛性更好，而MMAS种群的多样性更好，当二者数量比达到1:1时，可以达到收敛性和多样性的平衡。而且通过实验发现，随着种群数目的提升，精度并没有得到提高，反而增加了算法的运行时间，因此在本文中种群数目选择$H=4$时最为合适。

表2 CEACO算法中的基础参数设置
Table 2 Basic parameter setting in CEACO algorithm

| 参数 | ACS参数值 | MMAS参数值 |
|---|---|---|
| $\alpha$ | 1 | 1 |
| $\beta$ | 4 | 4 |
| $\rho$ | 0.3 | 0.2 |
| $\xi$ | 0.1 | |
| $q_0$ | 0.8 | |
| $T_{\max}$ | 2 000 | 2 000 |
| $m$ | 30 | 30 |
| $D$ | 50 | 50 |
| $H$ | 2 | 2 |

表3 种群个数的确定与配比
Table 3 Determination and matching of population number

| 种群数量 | ACS与MMAS种群不同比例的最优解 | | |
|---|---|---|---|
| | 1:1 | 1:2 | 2:1 |
| 2 | 29 583 | | |
| 4 | **29 459** | | |
| 6 | 29 482 | 29 631 | 29 529 |
| 9 | | 29 660 | 29 586 |

#### 3.1.2 综合判别系数$\gamma$的确定

在前文中通过竞争系数$Q$来判定种群的综合性能，由式(8)可知，综合判别系数$\gamma$的大小决定了多样性和解的质量在竞争系数中所占的比重，而$\gamma$取值的大小最终会影响到算法的收敛速度和求解的精度。因此本节选取城市规模不同的3个测

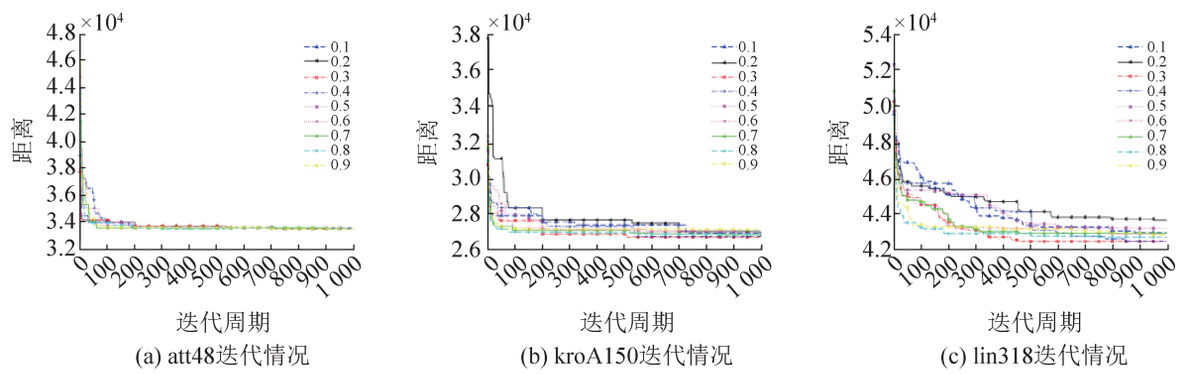



DOI: 10.16182/j.issn1004731x.joss.22-1009

第 36 卷第 1 期 系统仿真学报 Vol. 36 No. 1
2024 年 1 月 Journal of System Simulation Jan. 2024

试集att48、kroa150和lin318来进行测试，针对不同的 $\gamma$ 值分别进行15次实验，通过对比实验结果中的收敛速度和求解的质量来确定 $\gamma$ 的取值。由于算法运行到1 000代之后图像基本收敛成一条直线，所以为了能使图像更清晰的反映最优解随迭代周期的变化，只取前1 000次迭代图像。

由图4和图5可以看出，$\gamma$取值越大，解的质量在竞争系数中所占的比重越大，算法越倾向于将解的质量作为判断种群性能好坏的标准，此时算法的收敛速度较快，但是也容易陷入局部最优；$\gamma$取值越小，多样性在竞争系数中所占的比重越大，算法越倾向于将种群的多样性指标作为判断种群性能好坏的标准，此时虽然收敛速度放缓，但是种群搜索解的范围扩大，所得解的精度更高。经过实验对比发现，在这3种不同规模的城市中，当$\gamma=0.3$时，实验所获得的解和收敛速度最好。综上所述，综合判别系数$\gamma$设为0.3最为合适。




(a) att48迭代情况 (b) kroA150迭代情况 (c) lin318迭代情况

图4 不同$\gamma$值的迭代结果对比图

Fig. 4 Comparison of iteration results with different $\gamma$ values

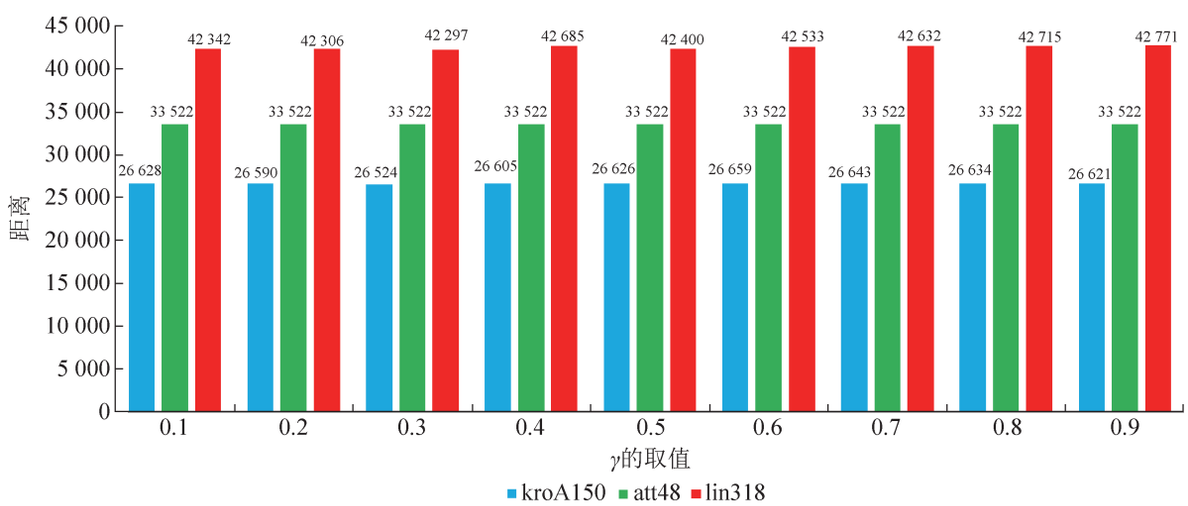



图5 $\gamma$ 取不同值时的最优解

Fig. 5 Optimal solutions for different $\gamma$ values

### 3.1.3 淘汰周期中 $p$ 的确定

由式(14)可知，本文中的淘汰周期是在交流周期的基础上得到的，为了能够及时对种群进行淘汰重组，需要设置一个合适的 $p$ 的值。由前文的淘汰周期求解公式可以得出，$p$的取值和种群规模的大小没有关系，因此本文选取kroA150作为测试集，针对不同 $p$ 值分别进行15次实验，根据种群所得到的平均解来确定 $p$ 的取值。

通过图6可以发现，当 $p=1$时，此时由于 $p$ 取值过小，淘汰周期和交互周期基本重合，种群



间竞争交互策略的作用还没有体现时，该种群可能就已经被淘汰，这会使种群间的交互失去效果。在本文中，考虑到算法的时间成本，将迭代周期设置为2 000代，当 $p=5$ 时，此时 $p$ 的取值较大，种群间的淘汰周期过长，经过2 000次迭代之后只会经历一次淘汰操作甚至没有淘汰操作，此时种群很容易陷入局部最优，淘汰重组的效果就难以体现。因此要选取一个合适的 $p$ 值，使种群间的交互和淘汰重组都能发挥出最大的作用。由实验结果分析可得，$p$ 取3时最为合适。

## 3.2 改进策略的有效性分析

本节为了验证CEACO算法中各个策略的作用，将不同的策略组合起来，组成了3组不同的优化方案，优化方案如表4所示。分别选取eil51、eil76、kroA100、kroA200、a280和lin318六组不同规模的城市集进行实验，每个城市集进行15次实验，统计各优化方案所求得的最优解、最差解、平均解、误差率以及迭代次数，并结合绘制出的收敛函数曲线来对各策略的性能进行分析，测试结果如表5和图7所示。

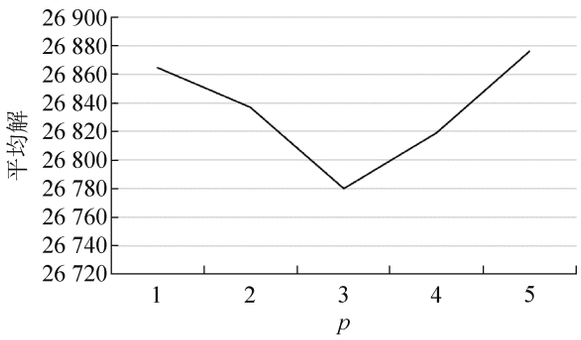



图6 $p$取值不同时的平均解

Fig. 6 Average solutions with different $p$ values

表4 优化方案表

Table 4 Optimization plan

| 方案 | 策略组合 |
|---|---|
| A | 多种群 |
| B | 多种群+竞争交互机制 |
| C | 多种群+竞争交互机制+淘汰重组机制 |

表5 优化方案性能对比

Table 5 Performance comparison of optimization schemes

| TSP | 标准最优解 | 算法模型 | 最优解 | 最差解 | 平均解 | 误差率 | 迭代次数 |
|---|---|---|---|---|---|---|---|
| eil51 | 426 | A | 426 | 428 | 426.8 | 0 | 220 |
| | | B | 426 | 427 | 426.5 | 0 | 125 |
| | | C | 426 | 427 | 426.3 | 0 | 60 |
| eil76 | 538 | A | 538 | 543 | 539.6 | 0 | 642 |
| | | B | 538 | 543 | 538.7 | 0 | 519 |
| | | C | 538 | 542 | 539.2 | 0 | 210 |
| kroA100 | 21 282 | A | 21 282 | 21 379 | 21 310.9 | 0 | 1 210 |
| | | B | 21 282 | 21 379 | 21 306.8 | 0 | 679 |
| | | C | 21 282 | 21 379 | 21 302.8 | 0 | 523 |
| kroA200 | 29 368 | A | 29 401 | 30 005 | 29 547.2 | 0.11 | 1 450 |
| | | B | 29 382 | 29 777 | 29 518.9 | 0.05 | 834 |
| | | C | 29 368 | 29 722 | 29 489.5 | 0 | 520 |
| a280 | 2 579 | A | 2 597 | 2 668 | 2 624.6 | 0.69 | 1 540 |
| | | B | 2 584 | 2 650 | 2 610 | 0.19 | 1 220 |
| | | C | 2 579 | 2 634 | 2 604.4 | 0 | 1 134 |
| lin318 | 42 029 | A | 42 608 | 43 577 | 43 180 | 1.38 | 1 409 |
| | | B | 42 475 | 43 464 | 43 053.4 | 1.06 | 1 091 |
| | | C | 42 297 | 43 524 | 42 803.8 | 0.64 | 775 |

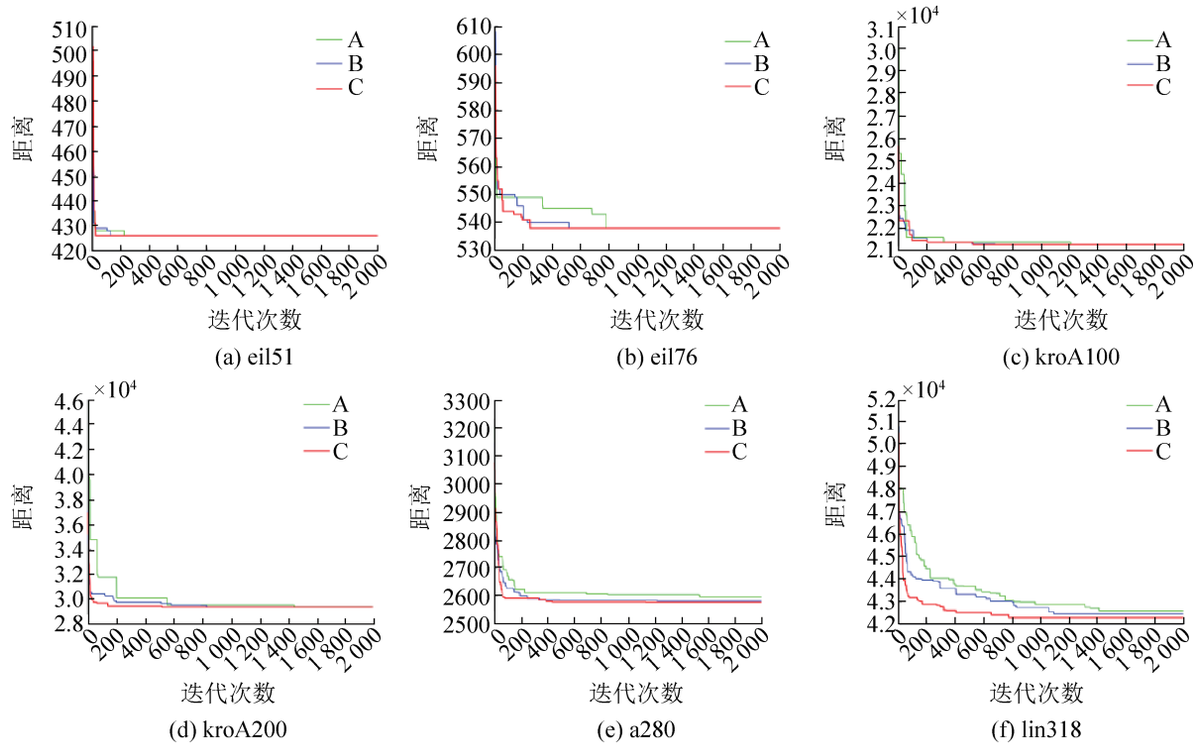



图 7　各测试集收敛情况

Fig. 7　Convergence of each test set

由上述实验结果可以看出，在小规模城市集中，3 种优化方案都可以求得标准最优解。面对中大规模的城市集，A 方案的求解精度和收敛速度相比于其他 2 个方案都不理想。在加入竞争交互机制之后，B 方案的收敛速度有了明显的提高，求解的精度也有改善，当城市数小于 200 时，B 方案都能求得标准最优解。这表明竞争交互机制在提升算法收敛速度的同时，也能提高算法的多样性，改善算法的求解精度。

但在大规模城市集中，B 方案在求解的精度表现的不是很理想。由于城市规模的扩大，算法此时很容易陷入局部最优，此时如果没有改进的策略，就会造成精度的降低。从实验数据可以看出，当 C 方案加入淘汰重组策略之后，针对陷入局部最优的种群或者精度较差的种群会定期进行淘汰和重组操作。在测试的前 5 个城市集中，C 方案都能求得标准最优解，在 lin318 城市集的实验中也可以将误差率降低到 0.64%。说明通过淘汰重组策略可以帮助算法跳出局部最优，提高算法的求解精度。

### 3.3　与经典蚁群算法的对比

在本节中，将改进算法 CEACO 与传统算法 ACS 以及 MMAS 算法的实验结果进行对比分析。采用 eil51、kroA100、kroA200、a280、lin318、pr439 等 15 组规模不同的城市集合进行实验，每个城市集运行 15 次，每轮进行 2000 次迭代，实验结果如表 6 所示。表中误差率计算公式为

$$E=\frac{L_{\text{best}}-L_{\min}}{L_{\min}}\times 100\% \tag{16}$$

表 6　与传统算法的性能对比

Table 6　Performance comparison with traditional algorithms

| TSP | 标准最优解 | 算法 | 最优解 | 最差解 | 平均解 | 误差率/% | 迭代次数 |
|---|---|---|---|---|---|---|---|
| eil51 | 426 | ACS | 427 | 436 | 428.0 | 0 | 306 |
| | | MMAS | 426 | 432 | 428.0 | 0 | 139 |

续表

| TSP | 标准最优解 | 算法 | 最优解 | 最差解 | 平均解 | 误差率/% | 迭代次数 |
|---|---|---|---|---|---|---|---|
| | | CEACO | 426 | 427 | 426.3 | 0 | 60 |
| | | ACS | 538 | 549 | 543.5 | 0.00 | 410 |
| eil76 | 538 | MMAS | 541 | 552 | 546.6 | 0.56 | 580 |
| | | CEACO | 538 | 542 | 539.2 | 0 | 210 |
| | | ACS | 21 282 | 21 673 | 21 392.6 | 0 | 656 |
| korA100 | 21 282 | MMAS | 21 292 | 21 841 | 21 467.6 | 0.04 | 1 226 |
| | | CEACO | 21 282 | 21 379 | 21 302.8 | 0 | 523 |
| | | ACS | 22 199 | 22 885 | 22 315.5 | 0.26 | 500 |
| kroB100 | 22 141 | MMAS | 22 220 | 22 649 | 22 420.8 | 0.36 | 638 |
| | | CEACO | 22 141 | 22 320 | 22 235.6 | 0 | 355 |
| | | ACS | 6 150 | 6 364 | 6 220.6 | 0.65 | 1 190 |
| ch130 | 6 110 | MMAS | 6 173 | 6 313 | 6 221.5 | 1.03 | 1 162 |
| | | CEACO | 6 110 | 6 173 | 6 148.8 | 0 | 517 |
| | | ACS | 6 553 | 6 650 | 6 586.6 | 0.38 | 233 |
| ch150 | 6 528 | MMAS | 6 548 | 6 620 | 6 580.0 | 0.31 | 607 |
| | | CEACO | 6 528 | 6 580 | 6 558.9 | 0 | 756 |
| | | ACS | 26 640 | 27 853 | 27 179.8 | 0.44 | 1 252 |
| kroA150 | 26 524 | MMAS | 26 677 | 27 130 | 26 962.1 | 0.58 | 1 878 |
| | | CEACO | 26 524 | 27 066 | 26 843.5 | 0 | 1 643 |
| | | ACS | 26 141 | 27 104 | 26 578.7 | 0.04 | 1 654 |
| kroB150 | 26 130 | MMAS | 26 135 | 26 691 | 26 404.0 | 0.02 | 1 744 |
| | | CEACO | 26 130 | 26 502 | 26 234.4 | 0.00 | 310 |
| | | ACS | 29 503 | 30 275 | 29 848.0 | 0.46 | 1 942 |
| kroA200 | 29 368 | MMAS | 29 401 | 29 920 | 29 634.9 | 0.11 | 1 714 |
| | | CEACO | 29 368 | 29 722 | 29 489.5 | 0 | 520 |
| | | ACS | 29 660 | 30 466 | 30 106.9 | 0.75 | 1 597 |
| kroB200 | 29 437 | MMAS | 29 586 | 30 559 | 30 204.4 | 0.51 | 1 920 |
| | | CEACO | 29 459 | 30 052 | 29 811.7 | 0.07 | 634 |
| | | ACS | 3 929 | 4 038 | 3 984.5 | 0.33 | 653 |
| tsp225 | 3 916 | MMAS | 3 935 | 4 078 | 3 996.4 | 0.49 | 1058 |
| | | CEACO | 3 920 | 4 017 | 3 958.5 | 0.10 | 845 |
| | | ACS | 2 584 | 2 727 | 2 634.2 | 0.19 | 1 297 |
| a280 | 2 579 | MMAS | 2 601 | 2 699 | 2 631.3 | 0.85 | 1 581 |
| | | CEACO | 2 579 | 2 634 | 2 604.4 | 0.00 | 1 134 |
| | | ACS | 42 741 | 44 851 | 43 385.3 | 1.69 | 1 817 |
| lin318 | 42 029 | MMAS | 43 074 | 44 695 | 43 611.5 | 2.49 | 1 887 |
| | | CEACO | 42 297 | 43 524 | 42 803.8 | 0.64 | 775 |
| | | ACS | 12 039 | 12 346 | 12 153.4 | 1.50 | 1 910 |
| f417 | 11 861 | MMAS | 12 050 | 12 506 | 12 259.5 | 1.60 | 1 990 |
| | | CEACO | 11 971 | 12 238 | 12 078.0 | 0.93 | 959 |
| | | ACS | 108 625 | 114 208 | 110 774.5 | 1.31 | 1 831 |
| pr439 | 107 217 | MMAS | 108 322 | 118 864 | 110 815.9 | 1.03 | 1 896 |
| | | CEACO | 107 818 | 113 112 | 110 112.3 | 0.56 | 1 045 |

由表 6 可以看出，对于中小规模的城市如 eil51、eil76、kroA100、kroB100 及 ch130 城市集，改进后的算法都能找到标准最优解，并且相对于传统的蚁群算法来说，寻找到最优路径所需的迭



https://dc-china-simulation.researchcommons.org/journal/vol36/iss1/18
DOI: 10.16182/j.issn1004731x.joss.22-1009

代次数更少，均在600代以内都搜索到了标准最优解。并且在ch130城市集中，CEACO所得到的平均解比两个传统算法得到的最优解都要好。这是由于改进算法采用了异构多种群算法，多个不同的蚁群算法并行搜索，提升了算法的全局寻优能力。

对于中大规模城市，本文选取了kroA150、kroB200、tsp225、a280等7个城市集。其中，除kroB200和tsp225两个城市集未收敛到标准最优解，其他城市集都能收敛到标准最优解，而kroB200和tsp225两个城市集的误差率也控制到了0.1%以内。并且在这6个城市集的实验结果中，无论是平均解还是最差解，CEACO所得到的结果相较于两种传统算法，均有较大的提升。

对于大规模的城市，本文选取了lin318、f417和pr439三个城市集，传统的ACS和MMAS算法在大规模城市中已经很难进行解的优化，算法的收敛速度和解的精度都有很大的改善空间。而CEACO算法在求解时都能将精度控制1%以内。这表明改进的CEACO算法可以较好的提升算法的求解精度。部分最优路径图如图8所示。

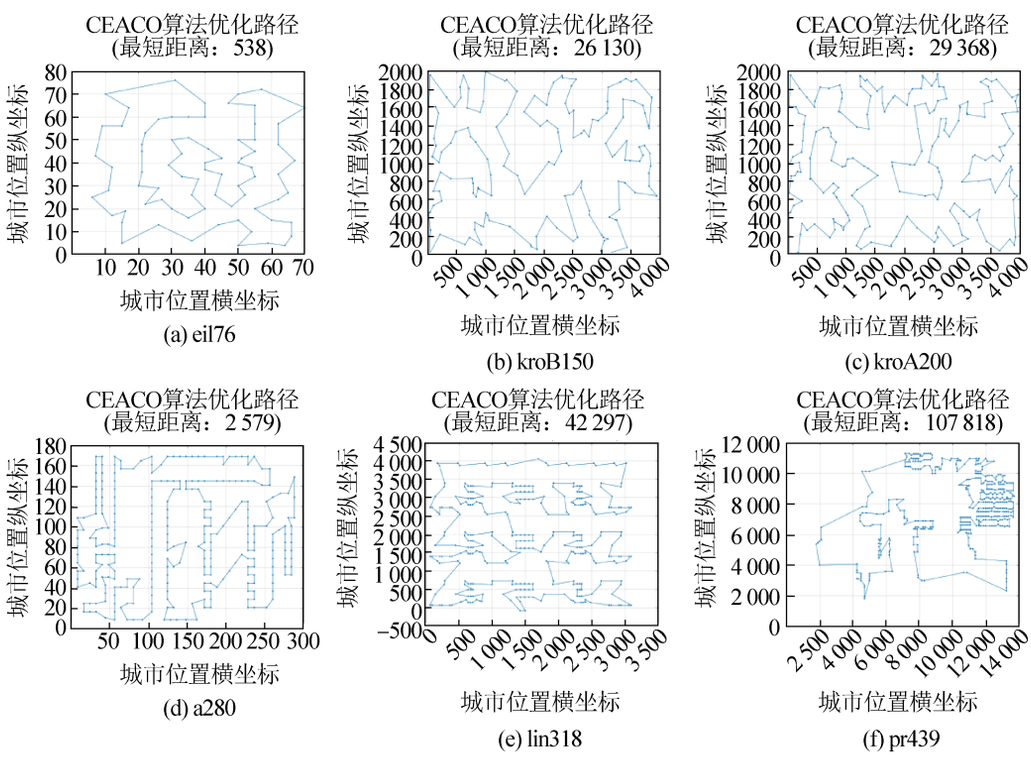



图8 部分最优路径图
Fig. 8 Partial optimal path

### 3.3.1 CEACO算法收敛性与多样性的分析

为了更好的验证CEACO算法的性能，本节将常规的多种群算法(ACS与MMAS相结合的多种群算法)与CEACO算法在多样性和收敛性两方面进行对比分析。本节选取了eil76、kroA200以及lin318三个不同规模的测试集进行试验，绘制出各自的多样性折线图和收敛曲线图。

在图9~11中，(a)表示常规的多种群算法的多样性图，(b)表示CEACO算法的多样性图。从图中可以观察到随着城市规模的扩大，算法的多样性都有所提升，并且(b)图中的多样性的值明显要高于(a)图中的值。在kroA200城市集中，常规多种群算法的多样性维持在(0.4, 0.8)这个区间内，而CEACO算法的多样性则基本维持在(0.97, 0.99)这



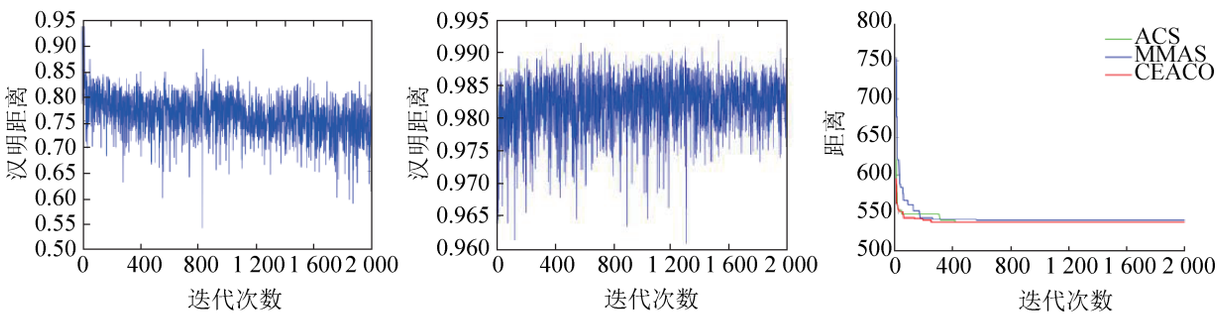
 

个区间内，这是由于种群间的竞争交互策略，促使算法在运行中加强种群间的交流，将算法的多样性一直维持在一个较高的区间内，从而丰富了种群的多样性。

图9~11中的(c)图表示算法的收敛曲线图，其中包含了常规多种群算法中的ACS和MMAS算法的收敛曲线以及CEACO算法的收敛曲线。从图(c)中可以明显的看出，CEACO算法在不同规模的测试集中的收敛速度与求解精度都要优于传统算法。这是由于在算法前期，由于竞争交互策略，各种群会交换最优解或者信息素矩阵，在保证一个较好的多样性的前提下促进了算法的收敛；同时，在后期种群搜索陷入停滞后，算法的淘汰重组机制对信息素矩阵进行了重制，提高了算法跳出局部最优的能力。

### 3.3.2 CEACO算法稳定性的分析

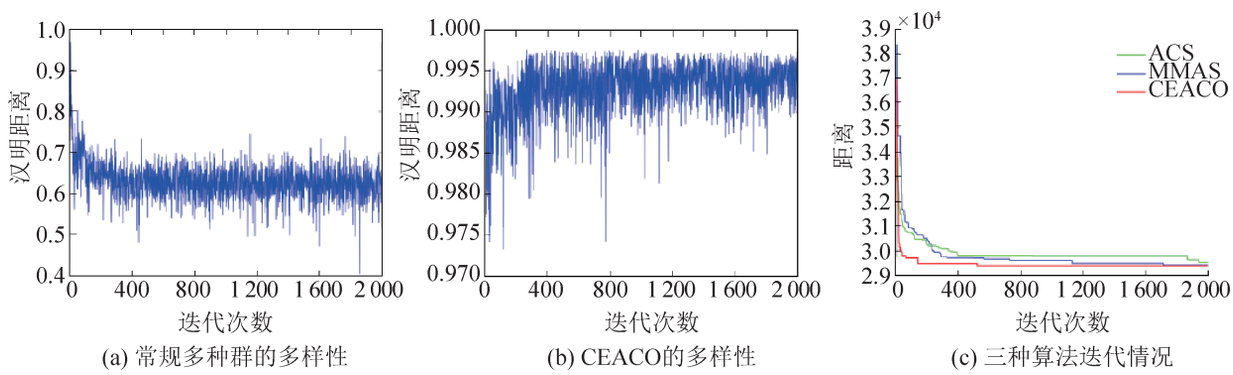
图12是3种算法在进行实验的15个城市集中的稳定性分析图，平均误差率是指平均解的相较于标准最优解的误差率。算法的平均误差率越小，表明该算法的平均解就越靠近标准最优解，算法的稳定性也就越强。由实验数据可以看出，两种传统算法在求解不同规模的TSP时，平均误差率各有高低。改进算法CEACO的平均误差率一直低于2种传统算法，在城市数小于300时，可以将平均误差率稳定在1.3%以内，大规模城市也可以将误差率稳定在3%以内。这表明CEACO算法有效的改善了平均解的质量，可以增强算法的稳定性。综上表明，改进的CEACO算法在各方面的性能都要优与传统的蚁群算法。




(a) 常规多种群的多样性 (b) CEACO的多样性 (c) 三种算法迭代情况

图9 eil76测试集

Fig. 9 Comparison of eil76 test set



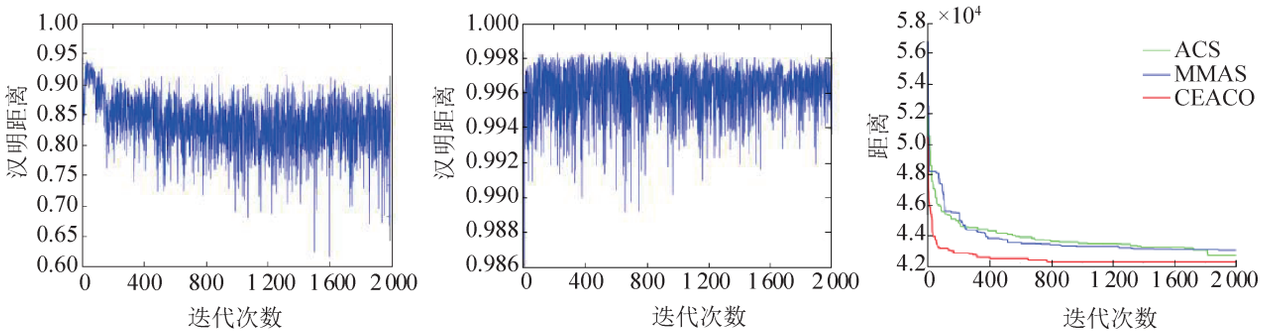

(a) 常规多种群的多样性 (b) CEACO的多样性 (c) 三种算法迭代情况

图10 kroA200测试集

Fig. 10 Comparison of kroA200 test set




DOI: 10.16182/j.issn1004731x.joss.22-1009




(a) 常规多种群的多样性 (b) CEACO的多多样性 (c) 3种算法迭代情况

图11 lin318测试集

Fig. 11 Comparison of lin318 test set

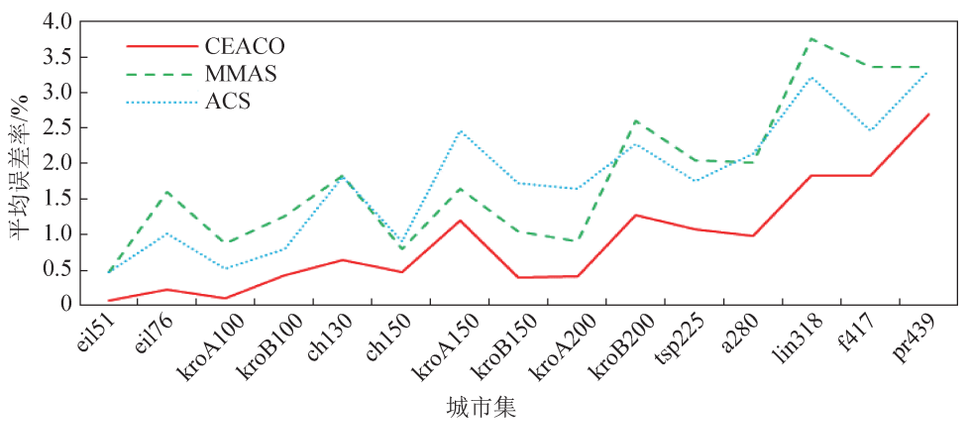



图12 算法稳定性对比

Fig. 12 Comparison of algorithm stability

### 3.4 与其他改进算法的对比

为了进一步验证本算法的性能，本文改进算法(CEACO)还与其他的改进算法进行了对比，对比结果如表7所示。其中，基于SAC模型的改进遗传算法(GA-SAC)来自于文献[17]，基于参数优化的改进蚁群算法(SOS-ACO)来自于文献[18]，结合价格波动策略与动态回溯机制的蚁群算法(PBACO)来自于文献[19]。由表7中的实验数据可以看出，对于小规模的测试集，各种改进算法基本都能求得标准最优解，在中规模的城市集中CEACO也能求得最优解，相比于其他算法表现更优；在大规模城市集中，虽然PBACO算法与SOS-ACO算法在lin318和pr439测试集中的平均解要优于本算法，但本算法在两个测试集中得到的最优解更优。综上所述，CEACO算法所得解的质量要明显优于对比算法。

表7 与其他改进算法对比

Table 7 Comparison with other improved algorithms

| TSP | 算法 | 最优解 | 平均解 | 误差率/% | TSP | 算法 | 最优解 | 平均解 | 误差率/% |
|---|---|---|---|---|---|---|---|---|---|
| eil51 | **CEACO** | **426** | **426.3** | **0** | kroA200 | **CEACO** | **29 368** | **29489.5** | **0** |
| | GA-SAC | 426 | 427.4 | 0 | | GA-SAC | — | — | — |
| | SOS-ACO | 426 | 428.1 | 0 | | SOS-ACO | 29 413 | 29 520.2 | 0.15 |
| | PBACO | 426 | 427 | 0 | | PBACO | 29 383 | 29 768 | 0.05 |

续表

| TSP | 算法 | 最优解 | 平均解 | 误差率/% | TSP | 算法 | 最优解 | 平均解 | 误差率/% |
|---|---|---|---|---|---|---|---|---|---|
| eil76 | **CEACO** | **538** | **539.2** | **0** | a280 | **CEACO** | **2 579** | **2 604.4** | **0** |
| | GA-SAC | 538 | 539.3 | 0 | | GA-SAC | 2 609 | 2 640.8 | 1.16 |
| | SOS-ACO | 538 | 541.7 | 0 | | SOS-ACO | — | — | — |
| | PBACO | 538 | 540 | 0 | | PBACO | 2 590 | 2 617 | 0.42 |
| kroA100 | **CEACO** | **21 282** | 21 302.8 | **0** | lin318 | **CEACO** | **42 297** | 42 803.8 | **0.64** |
| | GA-SAC | 21 282 | 21 300.1 | 0 | | GA-SAC | 42 329 | 42 902.4 | 0.71 |
| | SOS-ACO | 21 282 | **21 290.1** | 0 | | SOS-ACO | 42 473 | 42 762.7 | 1.05 |
| | PBACO | 21 282 | 21 345 | 0 | | PBACO | 42 384 | **42 432** | 0.84 |
| ch150 | **CEACO** | **6 528** | 6 558.9 | **0** | pr439 | **CEACO** | **107 818** | 110 112.3 | **0.56** |
| | GA-SAC | — | — | — | | GA-SAC | — | — | — |
| | SOS-ACO | 6 558 | 6 571.2 | 0.46 | | SOS-ACO | 107 978 | **108 873.8** | 0.71 |
| | PBACO | 6 533 | **6 551** | 0.07 | | PBACO | — | — | — |

## 4 结论

本文针对传统蚁群算法在求解TSP时容易陷入局部最优，收敛速度慢的问题，提出了一种结合竞争交互策略和淘汰重组机制的异构多蚁群算法(CEACO)。通过衡量各种群的竞争系数的差距对交互种群进行匹配，种群之间通过交互信息素矩阵和最优解来实现解的质量和多样性的平衡。同时，由于交流周期是自适应变化的，保证了对于不同种群和不同规模问题的适应性。算法引入的淘汰重组机制，可以依据各种群在淘汰周期前后的竞争潜力的差异变化确定出需要淘汰的种群；并利用重组机制来建立一个新的种群来维持系统的性能。通过该机制有效的提高了算法的求解精度。仿真实验结果表明，改进算法的收敛速度、求解精度以及稳定性等方面均有明显的提升。针对中小规模的TSP时，改进算法可以在较少的迭代周期内搜寻到标准最优解；针对中大规模的问题也能将误差率控制在较低的范围内，并且算法的平均解的质量也有所改善；但面对超大规模的TSP时，算法求解的精度还有待提高。

## 参考文献：

[1] Dorigo M, Thomas Stützle. Ant Colony Optimization [M]. Cambridge: MIT Press, 2004.

[2] Dorigo M, Maniezzo V, Colorni A. Ant System: Optimization by a Colony of Cooperating Agents[J]. IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics), 1996, 26(1): 29-41.

[3] Rodammer F A, White K P. A Recent Survey of Production Scheduling[J]. IEEE Transactions on Systems, Man and Cybernetics, 1988, 18(6): 841-851.

[4] Admane L, Benatchba K, Koudil M, et al. Using Ant Colonies to Solve Data-mining Problems[C]//2004 IEEE International Conference on Systems, Man and Cybernetics. Piscataway, NJ, USA: IEEE, 2004: 3151-3157.

[5] Chu C H, Gu Junhua, Hou Xiangdan, et al. A Heuristic Ant Algorithm for Solving QoS Multicast Routing Problem[C]//Proceedings of the 2002 Congress on Evolutionary Computation. Piscataway, NJ, USA: IEEE, 2002: 1630-1635.

[6] 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.

[7] Thomas Stützle, Holger H Hoos. MAX-MIN Ant System [J]. Future Generation Computer Systems, 2000, 16(8): 889-914.

[8] Zhao Haitong, Zhang Changsheng. An Ant Colony Optimization Algorithm with Evolutionary Experience-guided Pheromone Updating Strategies for Multi-objective Optimization[J]. Expert Systems with Applications, 2022, 201: 117151.

[9] Tuani A F, Keedwell E, Collett M. Heterogenous Adaptive Ant Colony Optimization with 3-opt Local Search for the Travelling Salesman Problem[J]. Applied Soft Computing, 2020, 97, Part B: 106720.

[10] Sangeetha V, Krishankumar R, Ravichandran K S, et al.




DOI: 10.16182/j.issn1004731x.joss.22-1009

Energy-efficient Green Ant Colony Optimization for Path Planning in Dynamic 3D Environments[J]. Soft Computing, 2021, 25(6): 4749-4769.

[11] 王铁, 胡泓. 基于K-means信息挥发速率动态调整的改进蚁群算法[J]. 机械与电子, 2020, 38(2): 25-29.
Wang Tie, Hu Hong. An Improved Ant Colony Algorithm Based on K-means and Dynamic Volatility Rate Adjustment Strategy[J]. Machinery & Electronics, 2020, 38(2): 25-29.

[12] Gambardella L M, Taillard E D, Agazzi G. A Multiple Ant Colony System for Vehicle Routing Problems with Time Windows[EB/OL]. (2016-04-28) [2022-08-17]. https://repository.supsi.ch/6353/.

[13] Wang Yuan, Wang Ling, Peng Zhiping, et al. A Multi Ant System Based Hybrid Heuristic Algorithm for Vehicle Routing Problem with Service Time Customization[J]. Swarm and Evolutionary Computation, 2019, 50: 100563.

[14] 刘双双, 黄宜庆. 多策略蚁群算法在机器人路径规划中的应用[J]. 计算机工程与应用, 2022, 58(6): 278-286.
Liu Shuangshuang, Huang Yiqing. Application of Multi-strategy Ant Colony Algorithm in Robot Path Planning [J]. Computer Engineering and Applications, 2022, 58 (6): 278-286.

[15] Zhang Dehui, You Xiaoming, Liu Sheng, et al. Multi-colony Ant Colony Optimization Based on Generalized Jaccard Similarity Recommendation Strategy[J]. IEEE Access, 2019, 7: 157303-157317.

[16] 张鹏, 薛宏全, 原欣伟. 基于相似度的自适应异类多种群蚁群算法[J]. 计算机工程与应用, 2014, 50(19): 37-41.
Zhang Peng, Xue Hongquan, Yuan Xinwei. Adaptive Heterogeneous Multiple Ant Colonies Algorithm Based on Similarity[J]. Computer Engineering and Applications, 2014, 50(19): 37-41.

[17] 陈斌, 刘卫国. 基于SAC模型的改进遗传算法求解TSP问题[J]. 计算机科学与探索, 2021, 15(9): 1680-1693.
Chen Bin, Liu Weiguo. SAC Model Based Improved Genetic Algorithm for Solving TSP[J]. Journal of Frontiers of Computer Science & Technology, 2021, 15 (9): 1680-1693.

[18] Wang Yong, Han Zunpu. Ant Colony Optimization for Traveling Salesman Problem Based on Parameters Optimization[J]. Applied Soft Computing, 2021, 107: 107439.

[19]赵家波, 游晓明, 刘升. 结合价格波动策略与动态回溯机制的蚁群算法[J]. 计算机科学与探索, 2022, 16(6): 1390-1404.
Zhao Jiabo, You Xiaoming, Liu Sheng. Ant Colony Algorithm Based on Price Fluctuation Strategy and Dynamic Back-tracking Mechanism[J]. Journal of Frontiers of Computer Science & Technology, 2022, 16 (6): 1390-1404.

