基于改进排挤遗传算法的旅行商问题研究

刘文涛[1]
LIU Wentao

摘 要 遗传算法在求解旅行商问题时，特别对于存在多个解的旅行商问题，容易陷入局部最优解，为了防止提前收敛，文章提出了一种基于改进排挤遗传的算法。算法使用小生境排挤遗传，使种群在遗传过程中，保持种群多样性，并能在排挤过程中逐渐聚焦到不同的最优解，可以一次性求出多个最优解路径。同时在遗传过程中，利用路径价值进行预算，剔除弱价值路径，用强价值路径进化，提高了进化速度。通过对多个案例的实验和测试，结果表明该算法具有很好的求解精度和运行效率。

关键词 遗传算法；排挤遗传；旅行商问题；路径价值预算

doi：10.3969/j.issn.1672-9528.2025.11.005

0 引 言

旅行商问题（travelling salesman problem, TSP）在很多领域都有应用，例如物流配送、机器人路径规划、电路设计等。TSP 问题可以描述为一个旅行商想要走遍 n 个地点，每个地点之间的长度是已知的，如何选择一条路线，使得旅行商每个地点只走一次，并且回到起点，最后所走的路线总长度是最短的。其数学模型定义为：

$$\min\ \text{Length}=\sum_{i=1}^{n-1}L(d_i,d_{i+1})+L(d_n,d_1) \tag{1}$$

式中：$L(d_i, d_{i+1})$ 表示两个地点 d_i 到 d_{i+1} 的距离长度，即在一个子集 $D = \{1, 2, \cdots, n\}$ 中找到一个序列 $\{d_1, d_2, \cdots, d_n\}$，使得总长度最小。

TSP 问题是一个 NP 完全问题，当问题规模扩大时，其计算时间会变得很大。求解 TSP 问题的方法有很多，例如人工鱼群算法 [1]、蚁群算法 [2]、神经网络 [3]、粒子群算法 [4]、模拟退火算法 [5]、人工协同搜索算法 [6]、麻雀搜索算法 [7]、深度强化学习 [8]、自适应对抗学习 [9]、禁忌搜索算法 [10] 等。在用遗传算法求解 TSP 问题时，会出现收敛过早的问题，容易导致染色体过早聚焦到某个局部最优解。另外，遗传算法一般只能求解到一个最优解，对于有多个最优解的 TSP 问题，其无法解决。基于此，本文使用小生境排挤遗传算法，使种群在遗传过程中根据距离进行聚集，相似的个体进化出不同的小生境区域，可以同时求出多个不同的最优解。同时，使用路径价值预算操作，使得更优的路径优先选择，有利于快速求出最优解，提高算法的求解效率。

1. 武汉轻工大学数学与计算机学院 湖北武汉 430023

1 改进排挤遗传

1.1 算子选择

遗传算法中，需要根据实际问题设计合理的编码，让问题的解和编码能够对应。对于 TSP 问题，有多种编码形式，例如顺序表示、路径表示或者邻近表示 [11]。本文算法采用路径表示，每个地点对应一位编码，则一个染色体表示的就是一条路径。适应度函数用于求染色体的适应度值，用总距离的倒数表示适应度值，适应度越大表示距离价值越大。其计算公式为：

$$\text{fitness}=\frac{1}{\sum_{i=1}^{n-1}L(d_i,d_{i+1})+L(d_n,d_1)} \tag{2}$$

选择算子 [12] 目的是从种群中选择一部分染色体遗传到下一代染色体中。一般遵循的规则是优胜劣汰，选择价值大的染色体，剔除价值弱的染色体。但要保持一定的种群多样性，防止过早收敛，所以采用一定的选择概率进行选择。一般采用赌轮盘选择方法进行选择。交叉算子 [13] 是遗传算法的重要过程，其作用是对于两个个体，使用一定的交叉方法产生两个新的子代个体。本文算法中，采用随机地点交叉方法。为了保持种群的多样性，在遗传的过程中，需要对染色体进行变异操作 [14]。文中采用单点变异方法实现，随机生成两个变异的位置点，然后对染色体的两个位置点数值进行交换。

1.2 路径价值预算

为了提高计算速度，使得种群在遗传过程中更加快速聚焦到最优解集合中，在算法中，使用一个路径价值预算的方法使得有价值的路径被优先选择。其实现过程如下：

（1）保存选择的某个染色体为临时个体。

（2）生成随机的位置点，由于第 1 个地点固定，所以要剔除第 1 个位置点。

（3）对临时个体进行随机位置点的地点值进行交换，产生新的临时个体。

（4）分别计算新的临时个体和原始个体的适应度值，比较适应度值，如果新的临时个体适应度值大，就用新的临时个体替换原始个体，否则原始个体保持不变。

1.3 海明距离

在排挤算法中，需要用海明距离[15]计算两个染色体之间的距离值，这个距离值一定程度反映相似性，对于相似性很高的两个个体，表示其之间的距离比较小。海明距离的计算过程如下，对于两个染色体个体，如果他们对应的每一位置的地点值相同，那么距离就增加 1，最后的总距离就是表示他们之间的海明距离。其计算公式为：

$$\text{hmdistance}=\sum_{i=1}^{n}\begin{cases}1,\text{if}(C_1(i)=C_2(i))\\0,\text{if}(C_1(i)\neq C_2(i))\end{cases} \quad (3)$$

式中：C_1 和 C_2 表示两个染色体个体。

1.4 算法步骤

整个算法的实现过程如下：

（1）初始化初始种群，设置变异算子和交叉算子概率。

（2）计算种群的适应度值，并设置位置选择列表，用来保存染色体是否被选择的标志。

（3）随机从种群中选择两个染色体，并在选择标志列表中标记此染色体被选择的标志，此两个染色体称为父代染色体。

（4）根据交叉概率，对这两个父代染色体进行交叉算子操作，产生两个新的染色体。

（5）根据变异概率，对两个新的染色体进行变异算子操作，产生新的子代染色体。

（6）对两个子代染色体，进行路径价值运算操作。把路径价值更高的染色体进化到下一代种群中。

（7）求出父代个体 1 和子代个体 1 的海明距离 $d(p_1, c_1)$，求出父代个体 1 和子代个体 2 的海明距离 $d(p_1, c_2)$，求出父代个体 2 和子代个体 1 的海明距离 $d(p_2, c_1)$，求出父代个体 2 和子代个体 2 的海明距离 $d(p_2, c_2)$，然后根据海明距离进行判断，过程如下：

if ($d(p_1, c_1) + d(p_2, c_2) <= d(p_1, c_2) + d(p_2, c_1)$

then

if ($f(c_1) > c(p_1)$ then $p_1 = c_1$ endif

if ($f(c_2) > f(p_2)$ then $p_2 = c_2$ endif

else then

if ($f(c_2) > f(p_1)$) then $p_1=c_2$ endif

if ($f(c_1) > f(p_2)$) then $p_2=c_1$ endif

endif

其中，函数 f 用于求出染色体的适应度值。通过比较海明距离进行判断，如直系子染色体更近，就用直系子染色体替换父代染色体，如果旁系子染色体更近，就用旁系子染色体替换父代染色体。这种排挤操作，会使得染色体进化出多个小生境区域，使得算法可以求出不同的最优解，而不会导致都聚集到唯一最优解。对于求解有多个最优解的 TSP 问题，这种方式可以同时求出更多不同的最优解。

（8）用新的染色体替换旧的染色体。

（9）循环执行（3）~（8），直到种群中的所有染色体都选择完毕。

（10）循环执行（2）~（9），直到运行满足指定的遗传代数。根据适应度函数值，求出最后种群中的最优解，即适应度函数值最大的染色体。对于具有相同适应度值但具有不同形式的最优解都要求出。

2 实验测试

2.1 算例选择

为了设计出具有多个最优解的 TSP 测试数据，需要对地点之间的距离值进行处理。每个地点之间的距离值是对称的，TSP 算例的所有数据表示形式为：

$$\text{Data}=\begin{cases}\{d_i,d_j,L(d_i,d_j)\}\\d_i\in\{0,1,\cdots,n-1\},d_j\in\{0,1,\cdots,n-1\},d_i<d_j\end{cases} \quad (4)$$

这里给出 5 个算例的数据信息，包括地点距离数据、地点个数、算例的最优解距离值，以及最优解的个数，用于算法测试对比。

算例 1：

{0,1,29}, {0,2,32}, {0,3,32}, {0,4,32}, {0,5,32}, {1,2,29}, {1,3,29}, {1,4,31}, {1,5,31}, {2,3,32}, {2,4,32}, {2,5,32}, {3,4,29}, {3,5,29}, {4,5,29}

算例 1 有 6 个地点，最优解的距离是 180，共有 20 个最优解。

算例 2：

{0,1,63}, {0,2,61}, {0,3,62}, {0,4,65}, {0,5,63}, {0,6,64}, {0,7,65}, {1,2,63}, {1,3,63}, {1,4,64}, {1,5,65}, {1,6,64}, {1,7,63}, {2,3,61}, {2,4,65}, {2,5,62}, {2,6,64}, {2,7,61}, {3,4,62}, {3,5,65},{3,6,64},{3,7,64}, {4,5,63}, {4,6,63}, {4,7,63}, {5,6,65}, {5,7,63}, {6,7,62}

算例 2 有 8 个地点，最优解的距离是 499，共有 22 个最优解。

算例 3：

{0,1,10}, {0,2,15}, {0,3,20}, {0,4,25}, {0,5,31}, {0,6,351}, {0,7,40}, {0,8,45}, {0,9,502}, {1,2,35}, {1,3,25}, {1,4,30}, {1,5,352}, {1,6,40}, {1,7,45}, {1,8,50}, {1,9,55}, {2,3,302}, {2,4,35}, {2,5,402}, {2,6,45}, {2,7,50}, {2,8,55}, {2,9,60}, {3,4,403}, {3,5,45}, {3,6,501}, {3,7,55}, {3,8,60}, {3,9,65}, {4,5,502}, {4,6,55}, {4,7,602}, {4,8,651}, {4,9,70}, {5,6,603}, {5,7,45}, {5,8,701}, {5,9,75}, {6,7,702}, {6,8,753}, {6,9,80}, {7,8,45}, {7,9,90}, {8,9,95}

算例 3 有 10 个地点，最优解的距离是 445，共有 198 个

最优解。

算例 4：

{0,1,12}, {0,2,16}, {0,3,16}, {0,4,17}, {0,5,12}, {0,6,12}, {0,7,12}, {0,8,17}, {0,9,12}, {1,2,16}, {1,3,16}, {1,4,12}, {1,5,12}, {1,6,12}, {1,7,17}, {1,8,12}, {1,9,12}, {2,3,17}, {2,4,16}, {2,5,16}, {2,6,16}, {2,7,12}, {2,8,12}, {2,9,12}, {3,4,16}, {3,5,17}, {3,6,16}, {3,7,17}, {3,8,17}, {3,9,17}, {4,5,17}, {4,6,12}, {4,7,12}, {4,8,17}, {4,9,12}, {5,6,16}, {5,7,16}, {5,8,12}, {5,9,16}, {6,7,12}, {6,8,16}, {6,9,12}, {7,8,12}, {7,9,12}, {8,9,17}

算例 4 有 10 个地点，最优解的距离是 128，共有 408 个最优解。

算例 5：

{0,1,18}, {0,2,18}, {0,3,20}, {0,4,18}, {0,5,18},{0,6,21}, {0,7,18}, {0,8,20}, {0,9,18}, {0,10,21}, {1,2,20}, {1,3,18}, {1,4,18}, {1,5,19}, {1,6,19}, {1,7,18}, {1,8,18}, {1,9,21}, {1,10,20}, {2,3,21}, {2,4,21}, {2,5,18}, {2,6,18}, {2,7,20}, {2,8,20}, {2,9,19}, {2,10,18}, {3,4,20}, {3,5,21}, {3,6,18}, {3,7,18}, {3,8,18}, {3,9,21}, {3,10,18}, {4,5,20}, {4,6,18}, {4,7,20}, {4,8,18}, {4,9,18}, {4,10,19}, {5,6,18}, {5,7,18}, {5,8,20}, {5,9,20}, {5,10,19}, {6,7,21}, {6,8,20}, {6,9,18}, {6,10,21}, {7,8,21}, {7,9,20}, {7,10,18}, {8,9,20}, {8,10,20}, {9,10,18}

算例 5 有 11 个地点，最优解的距离是 198，共有 410 个最优解。

2.2 测试结果

实验所用硬件为 Intel CPU 1.80 GHz，内存 32 GB，软件为 Windows10 操作系统，C++ 开发语言。算法对算例 1 和 2 的计算结果如表 1 所示，其中出发点是序号为 0 的地点。可以看出本文算法对算例 1 和 2 都可以求出全部最优解，正确率 100%，对于较小规模的 TSP 问题，本算法具有较高的精度。

表 1 算例 1、2 的计算结果

算例 1	算例 2
{0,1,2,3,4,5}	{0,1,3,2,7,6,4,5}
{0,1,2,3,5,4}	{0,1,3,4,6,7,2,5}
{0,1,2,4,3,5}	{0,1,3,4,6,7,5,2}
{0,1,2,4,5,3}	{0,1,6,7,2,3,4,5}
{0,1,2,5,3,4}	{0,1,6,7,2,5,4,3}
{0,1,2,5,4,3}	{0,1,6,7,5,4,3,2}
{0,1,3,4,5,2}	{0,1,7,6,4,3,2,5}
{0,1,3,5,4,2}	{0,1,7,6,4,5,2,3}
{0,2,1,3,4,5}	{0,2,3,1,7,6,4,5}
{0,2,1,3,5,4}	{0,2,3,4,5,7,6,1}
{0,2,4,5,3,1}	{0,2,5,4,6,7,1,3}
{0,2,5,4,3,1}	{0,2,5,7,6,4,3,1}

表 1（续）

算例 1	算例 2
{0,3,4,5,2,1}	{0,2,7,6,1,3,4,5}
{0,3,5,4,2,1}	{0,3,1,7,6,4,5,2}
{0,4,3,5,2,1}	{0,3,2,5,4,6,7,1}
{0,4,5,3,1,2}	{0,3,4,5,2,7,6,1}
{0,4,5,3,2,1}	{0,5,2,3,4,6,7,1}
{0,5,3,4,2,1}	{0,5,2,7,6,4,3,1}
{0,5,4,3,1,2}	{0,5,4,3,1,6,7,2}
{0,5,4,3,2,1}	{0,5,4,3,2,7,6,1}
	{0,5,4,6,7,1,3,2}
	{0,5,4,6,7,2,3,1}

算例 3~5 的计算结果比较如表 2 所示，其中 SGA 是简单遗传算法，BCGA 是基本排挤遗传算法，ICGA 是本文改进排挤遗传算法。EN 表示算例序号；TA 表示算法种类；UTILD 表示最后的多样性；AVERD 表示平均多样性；NOOS 表示最优解个数；AOPS 表示最终路径最优解正确率；APOSA 表示平均路径最优解正确率。

表 2 算例 3、4、5 的计算结果比较

EN	TA	UTILD	AVERD	NOOS	APOSA	AOPS
	SGA	0.87	0.88	5	0.02	0.03
3	BCGA	0.89	0.90	174	0.60	0.88
	ICGA	0.86	0.88	190	0.87	0.96
	SGA	0.95	0.94	1	0.01	0.00
4	BCGA	0.87	0.89	313	0.54	0.77
	ICGA	0.85	0.87	351	0.78	0.86
	SGA	0.93	0.94	1	0.00	0.00
5	BCGA	0.91	0.92	209	0.28	0.51
	ICGA	0.89	0.90	344	0.65	0.84

表 2 可以看出，随着 TSP 问题规模变大，所有算法的正确率都会降低。但无论哪种算例，ICGA 正确率都比 SGA 和 BCGA 要高，无论是最后的正确率还是平均正确率。SGA 的正确率最低，因为其算法过早收敛于局部最优解。SGA 的多样性会有波动，但 ICGA 的多样性都比 BCGA 的多样性要低，表明 ICGA 种群更加集中，综合可以看出 ICGA 具有更高的求解精度。

算法对算例 3 求解的最优解正确率变化曲线如图 1 所示，对算例 3 求解的平均适应度如图 2 所示，对算例 3 求解的种群多样性变化曲线如图 3 所示。从图 1 可以看出，SGA 算法的正确率很低，因为过早聚焦于某个局部最优解，无法在后面的遗传过程中探索更多不同的最优解。BCGA 算法引入遗传排挤操作，使得种群能够聚集到不同的最优解区域，可以求得更多的最优解。但 BCGA 的求解速度比较低，需要经过一定的遗传代数才能够达到求解精度，过程比较缓慢。

可以看出 ICGA 算法的正确率最高，而且其在较少的遗传代数下就可以达到较高的正确率，既比 BCGA 的效率高，也比 BCGA 的求解正确率高。从图 2 可以看出，算法 SGA 的适应度值最低，而且几乎未上升。BCGA 的适应度缓慢上升，速度较慢，而 ICGA 能够快速上升，在较短的遗传代数下就可以达到很高的适应度值。从图 3 可以看出，SGA 的多样性会有较大波动，趋势向下。BCGA 和 ICGA 的多样性会缓慢下降，表示种群在遗传过程中慢慢聚集到最优解附近，而且 ICGA 比 BCGA 下降速度要快，并且都比 BCGA 要低，表明 ICGA 种群更加集中到最优解。所以，从 3 张图中可以看出 ICGA 既有更高的求解精度，又有较快的求解速度。

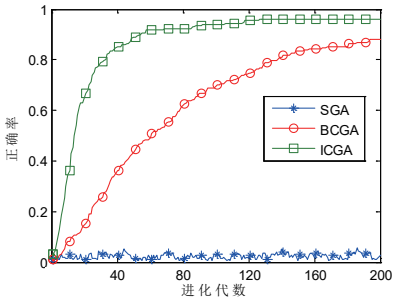

图 1 算例 3 的最优解正确率

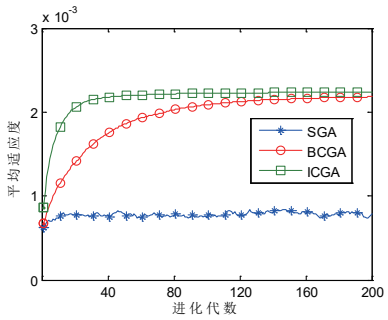

图 2 算例 3 的平均适应度

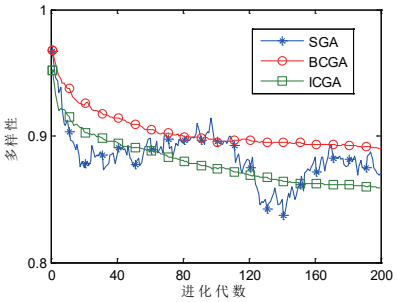

图 3 算例 3 的多样性

3 结语

遗传算法是求解 TSP 问题的一种启发式算法，但其具有过早收敛的问题，无法胜任对于有多个最优解的 TSP 问题。本文利用排挤遗传算法和基于路径价值预算处理，设计了一种改进遗传算法。其中排挤遗传利用小生境排挤遗传方式，使得在遗传过程中能让染色体聚集在不同区域，从而具有获得更多最优解的能力。基于路径价值预算处理使得在遗传过程中能更加快速地寻找最优解，提高了算法的求解效率。从实验结果来看，该算法对于存在多个最优解的 TSP 问题，具有较好的求解精度和求解速度。

参考文献：

[1] 王璞，刘宏杰，周永录．基于改进人工鱼群算法求解旅行商问题及多点路径规划 [J]. 科学技术与工程，2024, 24(35): 15090-15097.

[2] 杨一健，李明，方赛银．基于信息熵的改进蚁群算法求解 TSP 问题 [J]. 计算机工程与设计 ,2024,45(9):2874-2881.

[3] 韦念念，韩曙光．基于图卷积和注意力神经网络的旅行商问题新解法 [J]. 计算机科学 ,2024,51(S1): 210-217.

[4] 徐福强，邹德旋，章猛，等．改进混合粒子群算法求解旅行商问题 [J]. 智能计算机与应用 ,2022,12(11):229-235.

[5] 李昌兴，黄杉．求解旅行商问题的联合算子模拟退火算法 [J]. 西安邮电大学学报 ,2022,27(4):89-94.

[6] 唐阳丽．人工协同搜索算法及在旅行商问题中的应用 [D]. 西安：西安理工大学 ,2022.

[7] 张月栋，莫愿斌．改进的麻雀搜索算法及其求解旅行商问题 [J]. 计算机系统应用 ,2022,31(2):200-206.

[8] 陈浩杰，范江亭，刘勇．深度强化学习解决动态旅行商问题 [J]. 计算机应用 ,2022,42(4):1194-1200.

[9] 熊文瑞，陶继平．自适应对抗学习求解旅行商问题 [J]. 计算机工程与应用 ,2022,58(17):224-229.

[10] 唐文秀．基于改进禁忌搜索算法求解 TSP 问题 [J]. 科学技术创新 ,2022,(4):154-157.

[11] 姚明海，王娜，赵连朋．改进的模拟退火和遗传算法求解 TSP 问题 [J]. 计算机工程与应用 ,2013,49(14):60-65.

[12] JEBARI K, MADIAFI M. Selection methods for genetic algorithms[J]. International journal of emerging sciences. 2013, 3(4): 333-344.

[13] SONI N, KUMAR T. Study of various crossover operators in genetic algorithms[J]. International journal of computer science and information technologies, 2014, 5 (6):7235-7238.

[14] LIM S M, MD SULTAN A B, MUSTAPHA A,et al. Crossover and mutation operators of genetic algorithms[J]. International journal of machine learning and computing, 2017, 7(1):9-12.

[15] 邹泽桦，曾九孙，蔡晋辉．改进遗传算法求解柔性作业车间调度问题 [J]. 计算机测量与控制 ,2017,25(4):167-171.

【作者简介】

刘文涛（1977—），男，湖北广水人，硕士，副教授，研究方向：智能计算。

（收稿日期：2025-06-03 修回日期：2025-11-14）