北京航空航天大學

元启发式算法
在旅行商问题中的应用

作 者 姓 名　高欢

学　　号　SY2306234

学 科 专 业　软件工程

培 养 院 系　计算机学院

1. 引言

旅行商问题(Traveling salesman problem, TSP)是指对给定的 n 个城市以及每对城市之间的距离，求解访问每个城市一次并返回到起始城市的最短回路的问题。TSP 是一个组合优化问题，属于 NP-hard 问题，这意味着没有已知的有效算法能够在多项式时间内解决 TSP。对于大规模的 TSP 实例来说，如果使用传统的蛮力穷举算法求解精确的最短距离，计算时间会非常长，而采用启发式算法能够在可接受的时间范围内找到较好的解。本文将介绍并研究遗传算法、免疫算法、模拟退火、蚁群算法以及粒子群优化算法五种元启发式算法对 TSP 的优化效果。本文实验代码请见 https://github.com/HuanBit/MH-TSP.git。

2. 元启发式算法

元启发式算法(meta-heuristic algorithm)按照灵感来源可以划分为四个类别[1]，分别是基于生物进化、基于群体智能、基于物理定律以及基于人类行为的优化算法。

第一类是基于进化的算法。此类算法受到生物自然进化的启发，以随机生成的解开始其优化过程，不断将最好的解决方案放在一起创造新的解。新的个体通过变异、交叉和自然选择等进化方法生成。其中，最广为人知的是基于达尔文进化理论的遗传算法[2]。除此之外，禁忌搜索[3]以及差分进化[4]等算法也广为人知。

第二类是基于群体智能的算法，这些算法的灵感来自昆虫、鱼类或鸟类等生物群体的行为，其中最流行的是 Kennedy 和 Eberhart 等人[5]提出的粒子群优化算法、蚁群优化算法[6]以及猴群优化算法[7]等。

第三类是基于物理定律的启发算法，此类算法包括模拟退火算法[8]和谐搜索算法[9]等。

第四种是与人类行为相关的启发式优化算法，目前比较流行的算法包括基于教学学习的优化算法(Teaching learning-based optimization algorithm, TLBO)[10]以及联赛冠军算法(League Championship algorithm, LCA)[11]。

3. 方法设计

本节将介绍 5 种元启发式算法，包括遗传算法、免疫优化算法、模拟退火算法、蚁群算法与粒子群优化算法，并分析各优化方法针对 TSP 的设计。

3.1 遗传算法

遗传算法(Genetic Algorithm, GA)是一种受达尔文自然进化论启发的探索性搜索，由 Holland 等人[2]于 1975 年首次提出。该算法反映了自然选择的过程，即选择最适合繁殖的个体来产生后代。

基于生物遗传进化过程的启发式优化算法，用于解决搜索与优化问题。GA 模拟了生物在进化过程中的自然选择、遗传编译以及基因传递等机制，并通过迭代进化的方式逐渐向最优解靠拢。

遗传算法解决 TSP 问题的基本步骤描述如下。

（1） 初始化种群： 在问题空间中随机生成一个种群，其中的每个个体都是一种可行的回路。

（2） 适应度评估： 对每个个体计算适应度，表示其解的质量。

（3） 选择： 模拟自然选择现象，根据个体的适应度，选择一些个体作为父代，通常适应度较高的个体被选中的概率更大。

（4） 交叉：模拟生物学中的遗传交叉过程，从父代中选出一对个体，通过某种方式交换它们的基因，产生新的个体（解）。

（5） 变异：模拟生物进化中的变异现象，对子代中的一些解中的路线进行突变。

（6） 替换： 将新生成的子代替换掉原来的父代，形成新一代种群。

（7） 重复迭代： 重复上述步骤，直到达到迭代次数上限或找到满意的解。

3.2 免疫算法

免疫算法(Immune Algorithm, IA)[12]是指在人工免疫系统的理论基础上，模拟生物免疫系统的抗原识别、抗体生成、选择、迭代优化功能的优化算法。免疫算法本质上是更新适应度的过程，对于抽取的一个抗原，选取一个抗体去解决，并计算该抗体的亲和度，而后对抗体进行免疫处理，从而得到适应度更高

的抗体，在一次次的免疫处理过程中逐渐接近最优解。

免疫算法解决 TSP 问题的基本步骤描述如下。

（1） 初始化抗体： 将城市的排列表达为抗体，并随机生成一群抗体。

（2） 亲和力计算： 计算每个抗体与最优化目标（即最小化回路距离）的亲和力。

（3） 选择： 模拟自然免疫中的抗体选择过程，选择一些亲和力较高的抗体，作为下一代的种群。

（4） 克隆： 模拟免疫系统中的克隆和变异过程，将亲和力较高的抗体进行克隆，并根据某种机制引入一定的变异。

（5） 替换： 引入新的多样性，将克隆的抗体替代当前种群中的一部分抗体，防止早熟收敛。

（6） 重复迭代： 重复上述步骤，直到达到迭代次数上限或找到满意的解。

免疫遗传算法和遗传算法的结构基本一致，最大的不同之处在于，在免疫遗传算法中引入了浓度调节机制。进行个体选择操作时，遗传算法值只利用适应度值指标对个体进行评价，而免疫遗传算法的抗体选择策略变为：适应度越高，浓度越小，个体复制的概率越大；适应度越低，浓度越高的个体，被选择的概率就越小。

3.3 模拟退火算法

模拟退火算法(Simulated Annealing, SA)是一种模拟金属退火过程的元启发式算法。金属在退火时需要先加热升温，再逐渐降低温度，在此过程中使局部有序区域向外生长，从而增加材料的延展性，降低材料的应力，根据逐渐降低的逻辑“温度”，对原始路径进行随机扰动，扰动程度逐渐减小。

在采用模拟退火解决 TSP 问题时，温度变为对路径变化的随机性的度量，以寻求将路径距离最小化。当温度较高时，会产生较大的随机变化，避免陷入局部最小值的风险，然后随着温度下降接近最优的最小值。温度在指数衰减时间表上以一系列步骤下降，其中每一步的温度是前一步的 0.9 倍。

模拟退火算法解决 TSP 问题的基本步骤描述如下。

（1） 初始化：在待求解最短路径的地图空间中随机化一条满足条件的回路，并设定初始最高温度和最低温度。

（2） 温度控制： 算法通过一个温度参数来控制搜索过程。温度开始时较高，允许接受更差的解以避免陷入局部最优解，然后逐渐降低温度。温度的降低遵循一个退火计划，通常是指数函数或线性函数。

（3） 状态转移： 在当前解的附近随机生成一个新解，根据一定的概率条件决定是否接受新解。接受较差的解的概率在初始时较高，随着温度降低，接受较差解的概率逐渐减小。这一步模拟了原子在退火过程中的随机运动。

（4） 更新当前解： 如果新解被接受，则将其作为当前解。否则，保持当前解不变。

（5） 重复迭代： 重复上述过程，直到温度降低到设定的结束温度或达到一定的迭代次数。

模拟退火算法的关键之处在于它以一定的概率接受较差的解，这有助于避免陷入局部最优解，使算法更有可能全局搜索到最优解。算法性能的好坏与参数设置、初始解的选择以及温度降低的速度等因素有关。

3.4 蚁群算法

蚁群算法(Ant Colony Optimization, ACO)是一种群体智能算法，通过模拟蚂蚁在寻找食物过程中利用信息素交流来解决优化问题。ACO 的基本思想来源于蚁群在觅食时的行为。当蚂蚁个体找到食物后会返回巢穴，并沿途释放一种用于标记路径的信息素，其他蚂蚁在寻找食物时能够根据这些信息素的引导，沿着信息素浓度较高的路径前进。与此同时，信息素会随着时间推移挥发。随着蚂蚁不断沿着更短的路径前进，蚁群最终的路径会接近最短路径。

蚁群算法解决 TSP 问题的基本步骤描述如下。

（1） 初始化信息素：在待求解最短路径的地图空间中初始化一条满足条件的回路，并为每条路线上的点分配一些初始信息素。

（2） 模拟蚂蚁行为：一群虚拟的蚂蚁根据某种规则在问题空间中移动。每只蚂蚁根据信息素浓度和启发式信息（例如距离）做出决策。

（3） 信息素更新： 蚂蚁完成移动后，根据它们的表现更新路径上的信息素。与此同时，路线上的信息素会以一定的速度削弱，从而模拟信息素的挥发过程。同时，更优线路上信息素浓度不断增加，使其更有可能被其他蚂蚁选择。

（4） 重复迭代： 重复（2）~（3）步骤，直到达到迭代次数上限或找到满

足某个阈值的解。

通过模拟蚂蚁在路径上的信息交流和合作，蚁群算法能够有效地解决组合优化问题，如旅行商问题、调度问题等。

3.5 粒子群优化算法

粒子群优化算法(Particle Swarm Optimization, PSO)的思想受到鸟群觅食的启发，鸟群在觅食中通过集体的信息共享使群体找到最优的目的地。鸟群在森林中搜索食物，群体的目的是找到食物最多的位置。但事实上所有的鸟都不知道食物的具体位置，而只能感受到食物的大概方向。因此每只鸟沿着自己判定的方向进行搜索，并在搜索的过程中记录自己曾经找到过食物且量最多的位置，同时所有的鸟都共享自己每一次发现食物的位置以及食物的量，这样鸟群就知道当前在哪个位置食物的量最多。在搜索的过程中每只鸟都会根据自己记忆中食物量最多的位置和当前鸟群记录的食物量最多的位置调整自己接下来搜索的方向。鸟群经过一段时间的搜索后就可以找到森林中哪个位置的食物量最多。

粒子群算法解决 TSP 问题的基本步骤描述如下。

（1） 初始化粒子群：随机生成一群粒子，并初始化粒子的位置和速度。

（2） 评估适应度：计算每个粒子的适应度，即当前回路的距离。

（3） 更新粒子速度和位置：根据粒子当前在 i 时刻的位置$x_i(t)$、i+1 时刻的速度$v_i(t+1)$、个体最优位置p_{best}和整个群体的全局最优位置$gbest$来更新粒子的速度和位置。粒子位置更新公式如下。

$$x_i(t+1) = x_i(t) + v_i(t+1)$$

粒子速度更新公式如下，其中，w 是控制粒子的速度惯性，c_1与c_2分别控制个体与全局的影响，r_1与r_2为[0,1]的随机数。

$$v_i(t+1) = w * v_i(t) + c_1 r_1(p_{best} - x_i(t) + c_2 * r_2 * [gbest - x_i(t)]$$

（4） 迭代优化： 重复（2）~（3），直到达到最大迭代次数或目标函数值小于设定阈值。

粒子群优化算法通过当前搜索的最优点进行信息共享，目前被广泛应用于参数优化、神经网络训练与图像处理等领域。

4. 实验设计与结果

为了对 GA、SA、ACO、PSO 和 IA 算法进行这些实验，本文使用了来自 GitHub 的开源库 scikit-opt。scikit-opt 是一个专门为启发式算法构建的 Python 库和模块，可以被用于解决参数优化、函数优化等问题。scikit-opt 支持连续和离散优化： 可以用于处理包括连续和离散变量的优化问题，并且其中部分算法支持并行化以提高计算效率。

本实验总共在 1x1 大小的坐标平面上随机生成了三组地图，每组包含五个地图，每个地图分别含有 10、20、30、40 及 50 个城市，上述每个地图均可以作为待求解的 TSP 问题。

本实验基于 scikit-opt 实现了上述五种启发式算法在 TSP 问题中的应用，并在随机生成的地图上测试了各算法的应用效果。五个优化算法的目标均为最小化旅行回路总距离，即最小化 cal_total_distance(routine)函数，迭代次数均设置为 200 次，待求解 TSP 问题所包含的城市数量即 NUM_POINTS 为变量。遗传算法中，初始种群数量 size_pop 设定为 200；免疫算法中，抗体间亲和度阈值 T 设定为 0.7，抗体浓度重要性 alpha 设定为 0.95，初始抗体与抗原数量和为 200；模拟退火算法中，初始温度为 T_max=100，最低温度 T_min 为 1，每个温度下的最大迭代次数 L 设定为 10*NUM_POINTS 次；蚁群算法中，信息素重要程度设定为 1，适应度重要程度设定为 2，信息素挥发速度设定为 0.1；粒子群算法中，粒子数量设定为 200，c_1与c_2均为 0.5。

4.1 不同算法的优化性能比较

本节从生成路径的距离与迭代 200 次所用时间两个指标，探讨五种优化算法在同一地图下的性能表现。

对于城市数较少的情况，即 10 个城市，五种方法产生了近似的结果，其中 GA、ACO 与 PSO 算法产生了相同的最优路径。SA 方法迭代 200 轮的耗时最短。

表 1 五种方法的优化性能比较（10 个城市）

优化算法	GA	IA	SA	ACO	PSO
最短距离	**3.1010**	3.1403	3.2851	**3.1010**	**3.1010**
时间(s)	1.0078	1.5449	**0.4340**	3.8247	2.1518

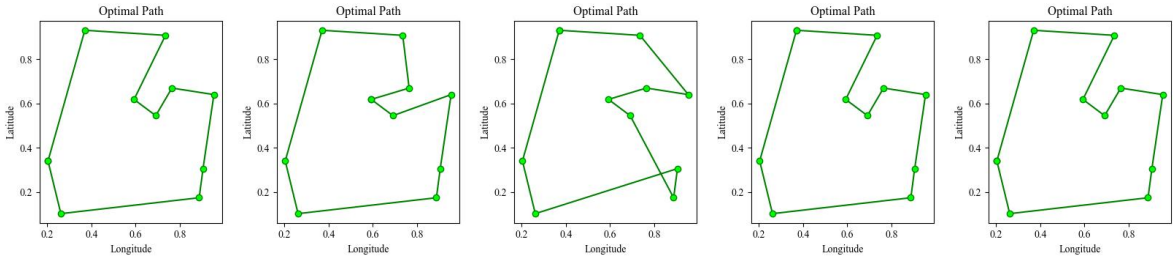


图 1 五种优化方法生成的最优回路（10 个城市）

随着城市数量增加到 30 个，五种方法的优化效果出现了明显的区别。其中，SA 算法生成的路径仍然存在诸多重叠的交叉线路，计算所得的距离最长。GA 与 IA 算法生成的路径同样存在明显的交叉。ACO 与 PSO 算法产生的结果较为相近，ACO 算法的优化结果最佳，但其耗时显著高于 PSO 算法。

表 2 五种方法的优化性能比较（30 个城市）

优化算法	GA	IA	SA	ACO	PSO
最短距离	7.9665	9.4378	11.3903	4.8557	**4.8159**
时间(s)	2.1925	3.7089	**2.0692**	12.9509	3.3079

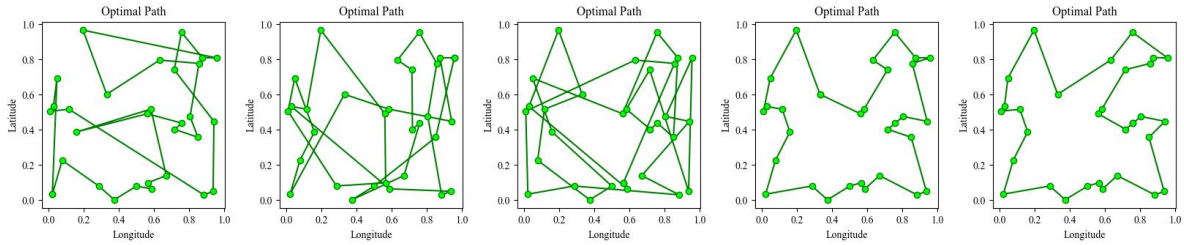


图 2 五种优化方法生成的最优回路（30 个城市）

当城市数量进一步增多至 50 个城市，五种方法在迭代 200 次后产生的结果与 30 个城市结点时的相似。同时，尽管 SA 方法生成的距离约是 ACO 方法的 3 倍，但其耗时仅为 ACO 方法的$\frac{1}{5}$。

表 3 五种方法的优化性能比较（50 个城市）

优化算法	GA	IA	SA	ACO	PSO
最短距离	13.2401	15.3781	20.4174	**6.4321**	6.5086
时间(s)	**3.4850**	5.7569	4.7041	23.9980	4.6741

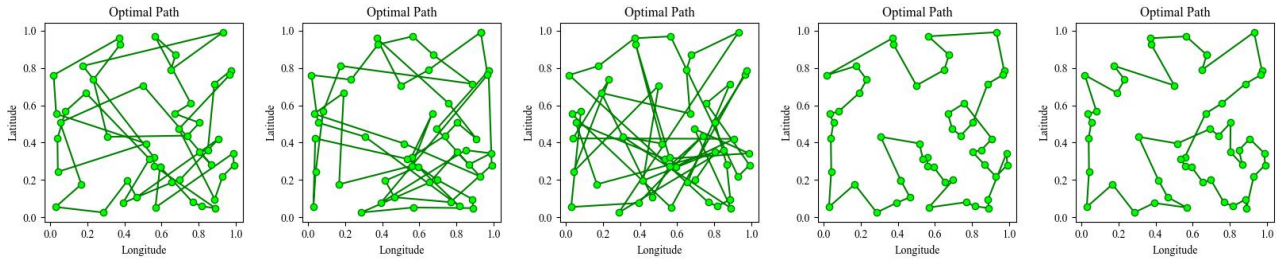


图 3 五种优化方法生成的最优回路（50 个城市）

4.2 在不同地图上性能的比较

本节研究城市数量为 30 的情况下，五种方法在三组随机生成的地图上的优化表现。

表 4 GA 在三组地图上的优化性能比较

地图编号	1	2	3
最短距离	7.6205	7.6197	7.9452
时间(s)	2.1684	2.2103	2.2649

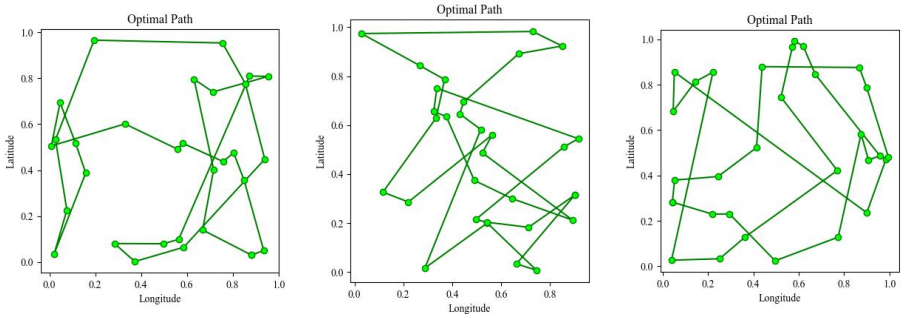


图 4 GA 在三组地图上生成的最优回路

表 5 IA 在三组地图上的优化性能比较

地图编号	1	2	3
最短距离	9.2867	8.3421	10.0928
时间(s)	3.6426	3.6022	3.4410

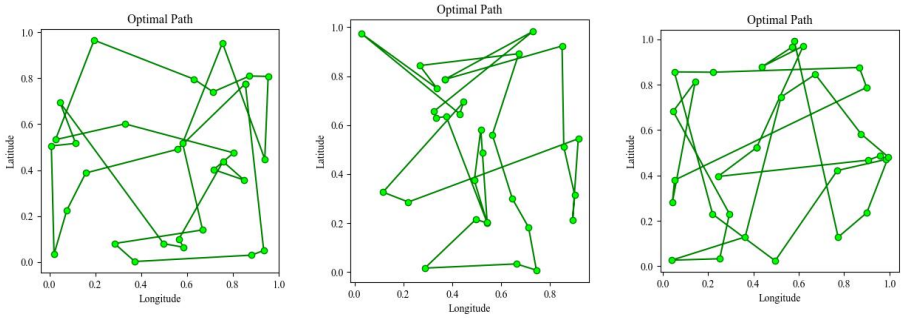


图 5 IA 在三组地图上生成的最优回路

表 6 SA 在三组地图上的优化性能比较

地图编号	1	2	3
最短距离	11.6351	9.4422	11.7155
时间(s)	2.3953	2.1910	3.1276

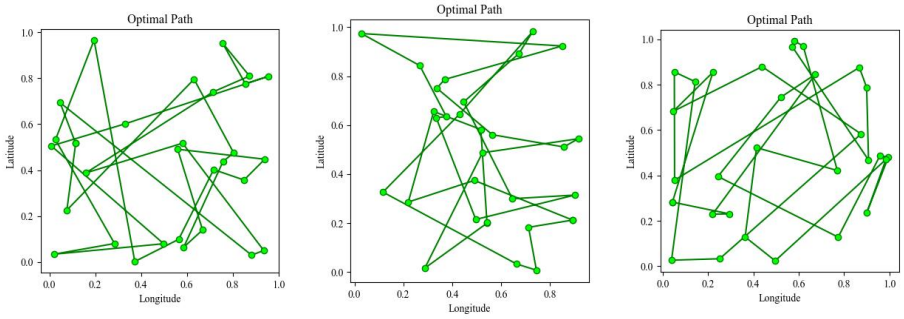


图 6 SA 在三组地图上生成的最优回路

表 7 ACO 在三组地图上的优化性能比较

地图编号	1	2	3
最短距离	4.8330	4.6326	4.8934
时间(s)	13.3648	13.3557	13.2198

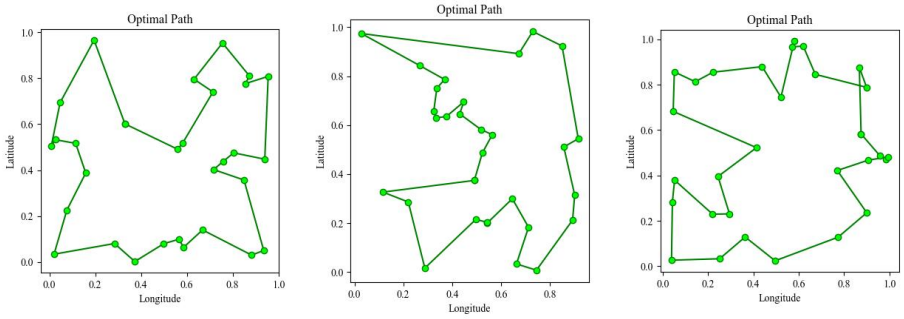


图 7 ACO 在三组地图上生成的最优回路

表 8 PSO 在三组地图上的优化性能比较

地图编号	1	2	3
最短距离	4.8283	4.6921	4.7038
时间(s)	3.5685	3.4058	3.5315

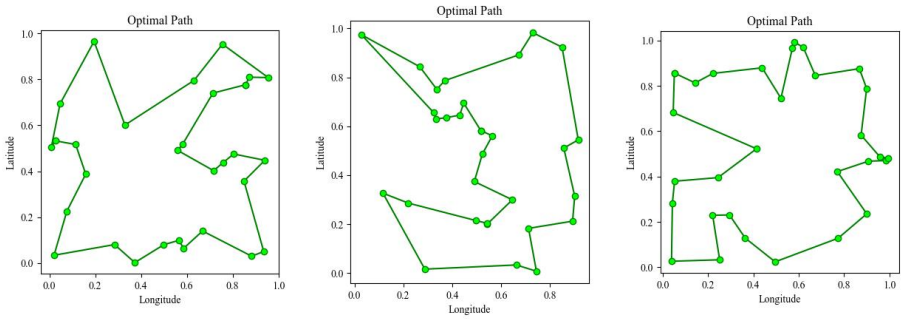


图 8 PSO 在三组地图上生成的最优回路

通过观察可以发现，本文所讨论的五种算法的效果较为稳定，对于三组随机生成的地图均能够在 200 次迭代后生成较为理想的优化路径。其中，SA 算法生成的路径较为复杂，产生交叉的路线较多，对于三组地图其迭代生成的路径长度均最差，但对于同一地图的耗时最低。

4.3 在不同城市数量上的比较

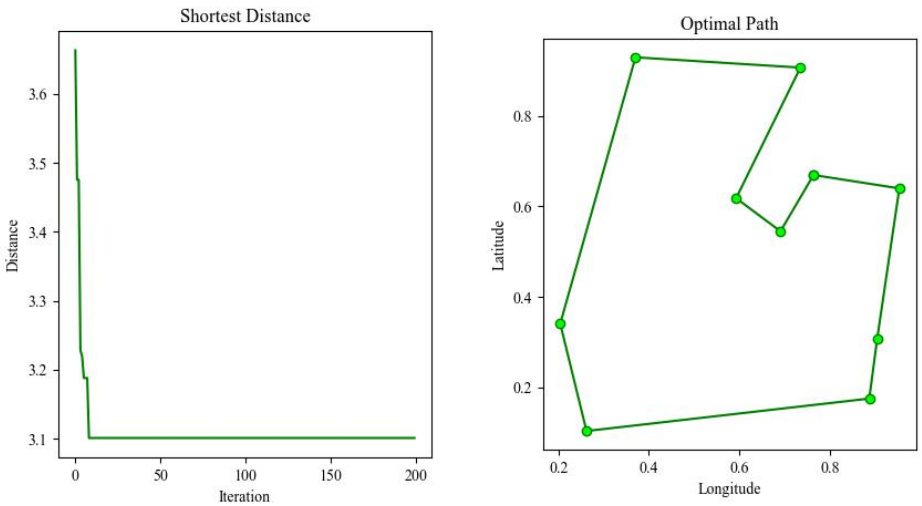
本节对于本文所讨论的五种优化方法，研究其在不同城市数量下的表现。

4.3.1~4.3.5 分别是五种算法在同一组地图下的优化结果，每小节中包括各算法在 10、20、30、40 与 50 五种城市数量下的优化表现。结果描述于 4.3.6。

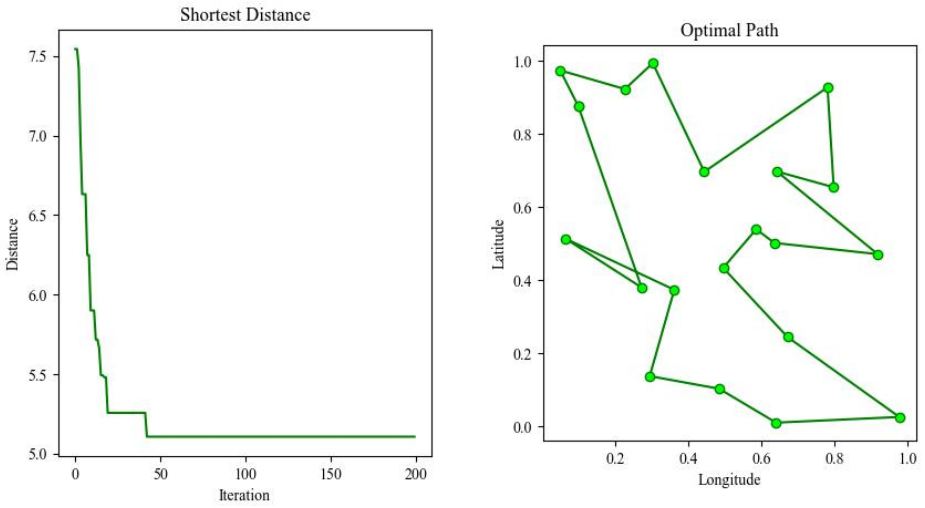
4.3.1 遗传算法

表 9 GA 求解 TSP 问题的结果

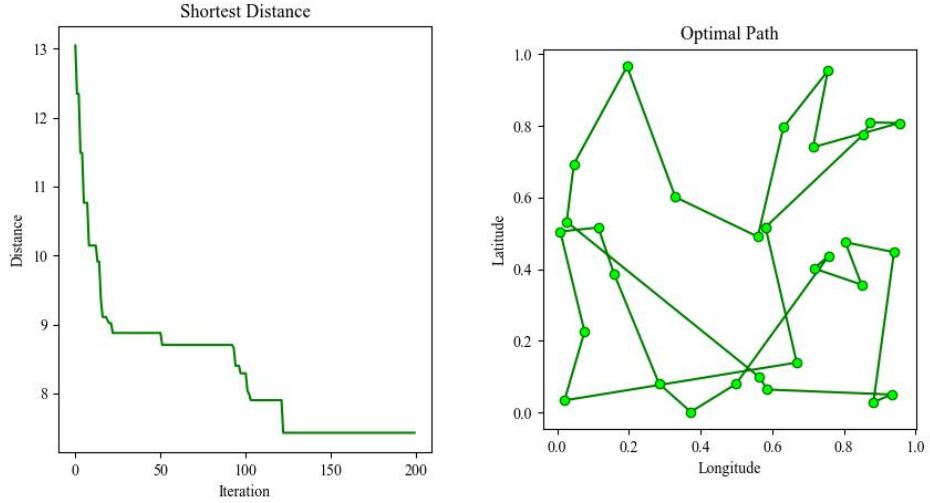
城市数	10	20	30	40	50
最短距离	3.1010	5.1070	7.4292	9.0154	12.8235
时间(s)	1.1322	1.8272	2.3581	2.9891	3.6106



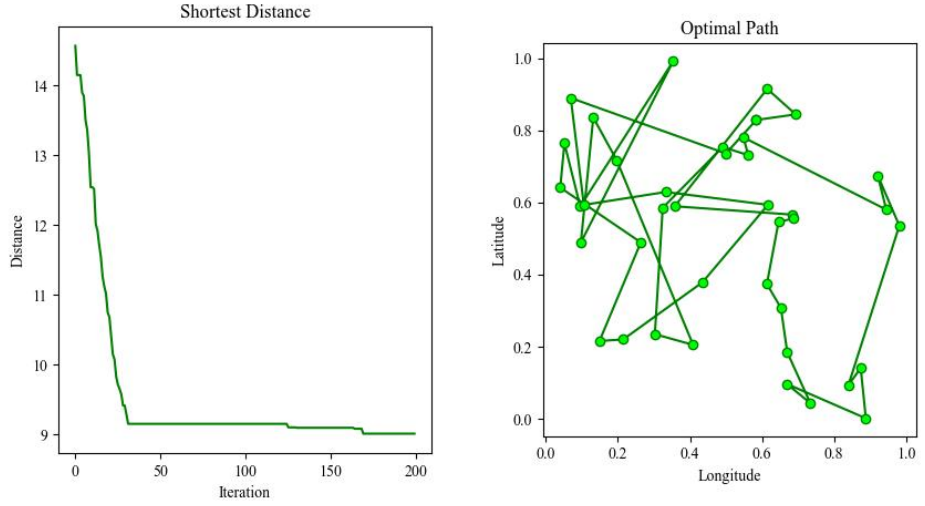
(a) 10 个城市



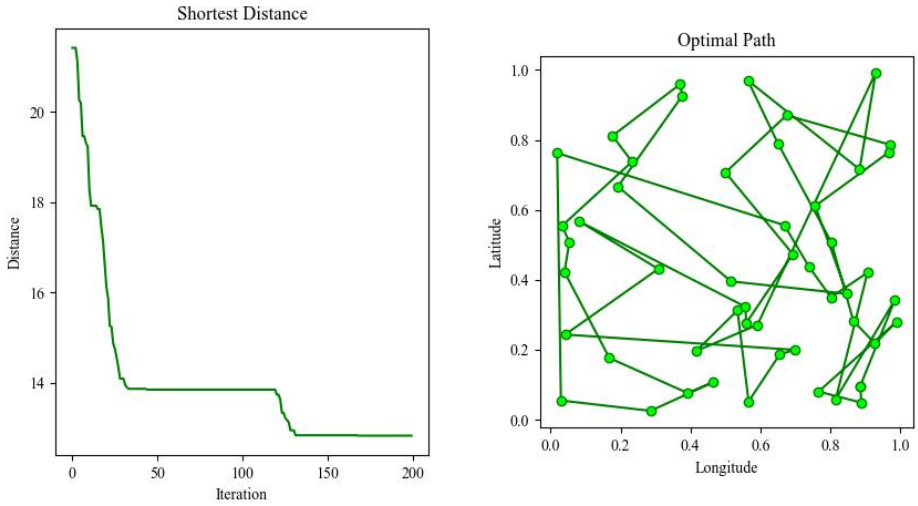
(b) 20 个城市



(c) 30 个城市



(d) 40 个城市



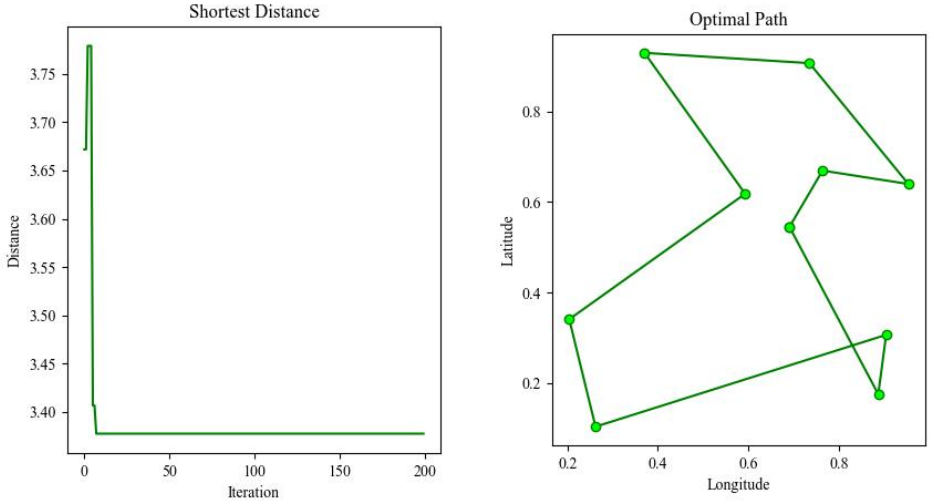
(e) 50 个城市

图 9 GA 生成的最优回路

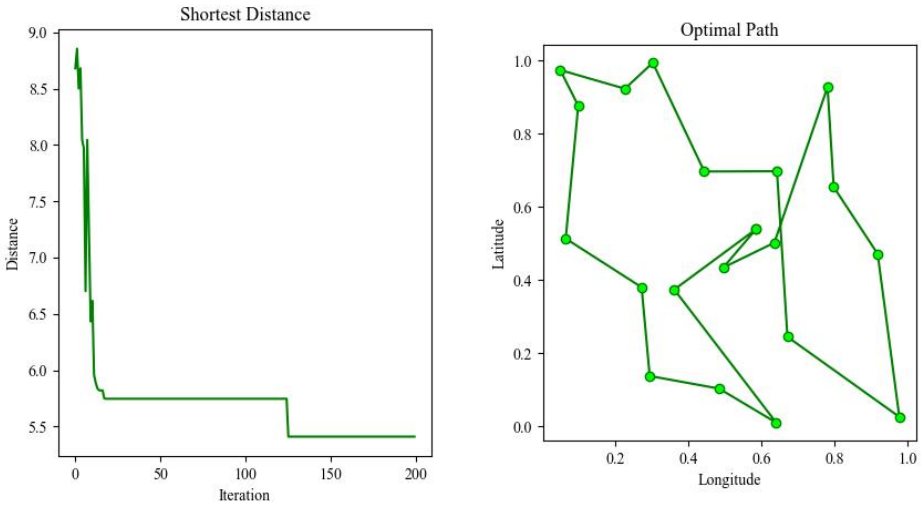
4.3.2 免疫优化

表 10 IA 求解 TSP 问题的结果

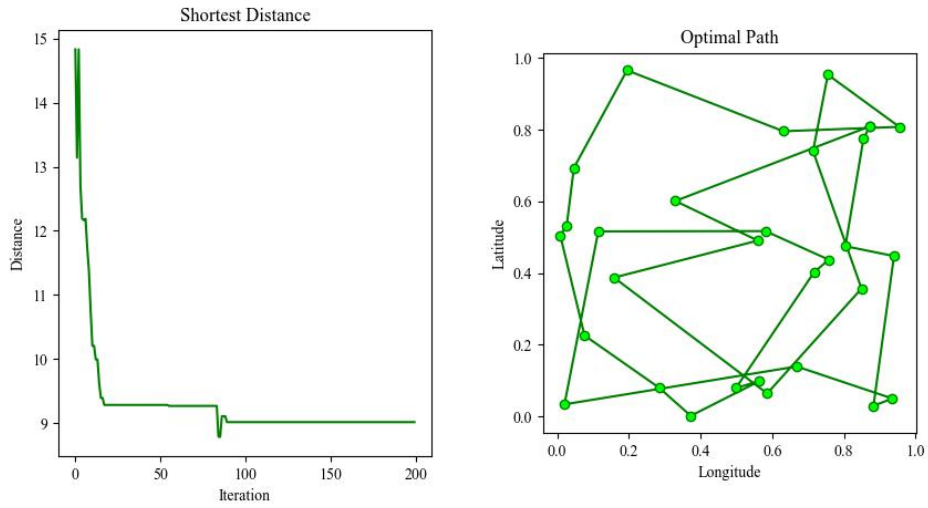
城市数	10	20	30	40	50
最短距离	3.3772	5.41	8.7865	9.4499	16.1586
时间(s)	1.5771	2.7097	3.8831	4.788	5.9383



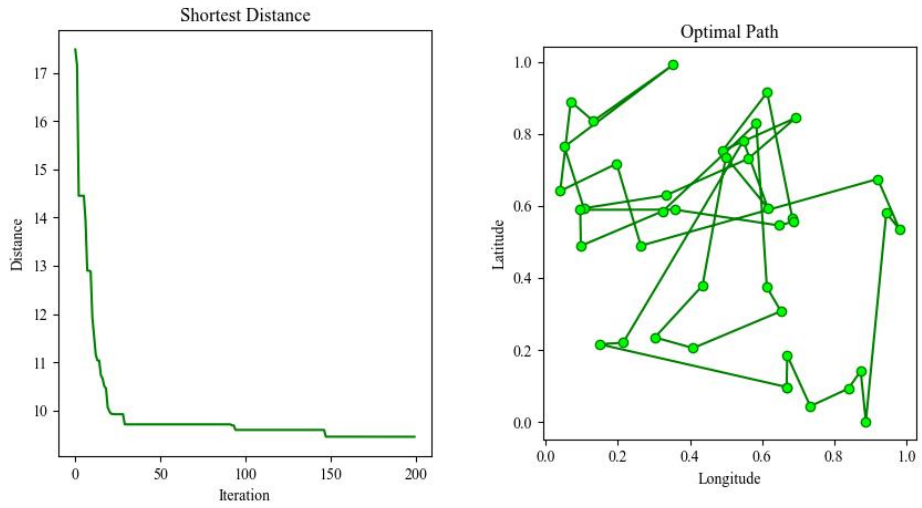
(a) 10 个城市



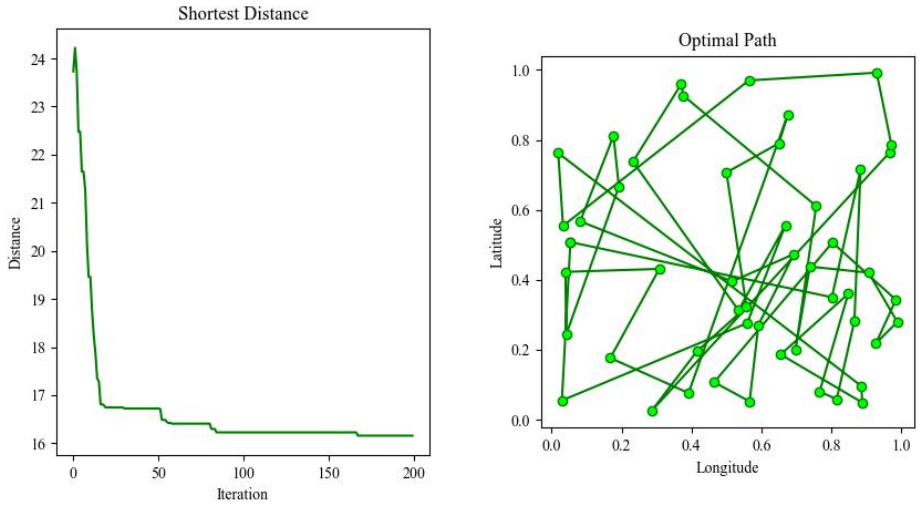
(b) 20 个城市



(c) 30 个城市



(d) 40 个城市



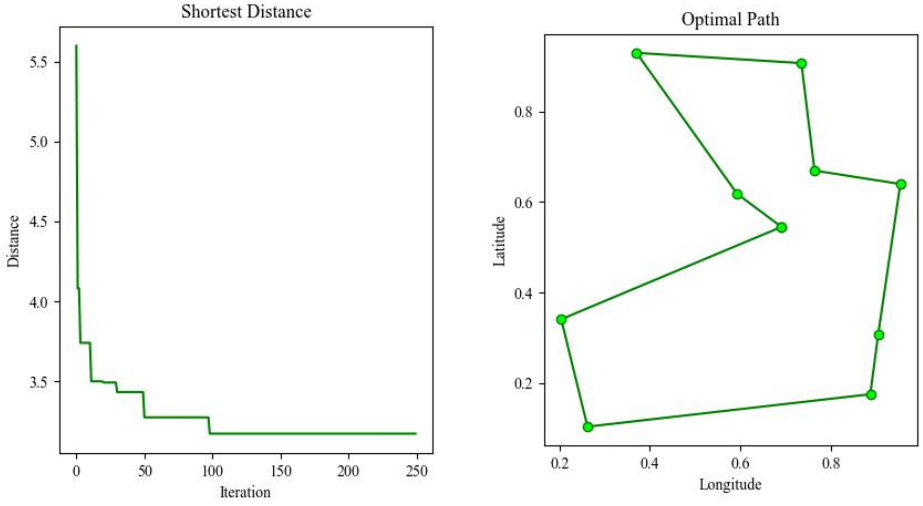
(e) 50 个城市

图 10 IA 生成的最优回路

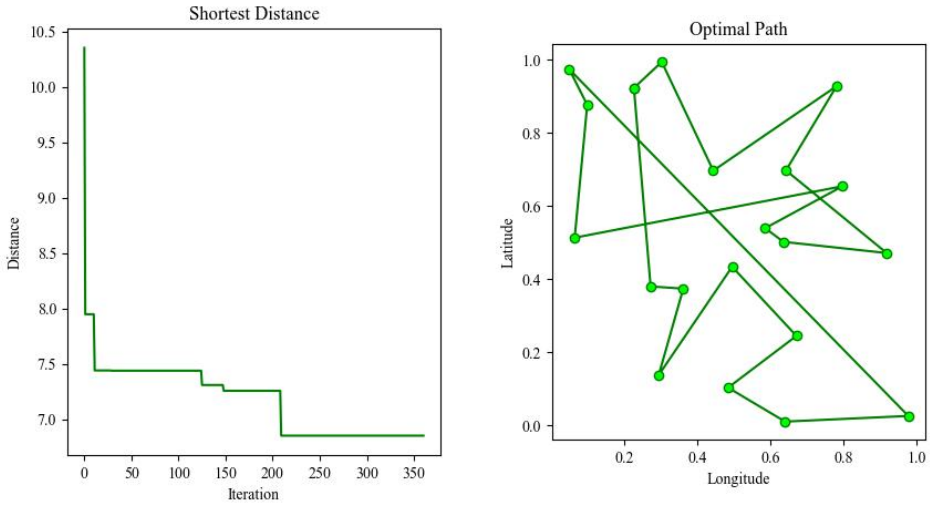
4.3.3 模拟退火

表 11 SA 求解 TSP 问题的结果

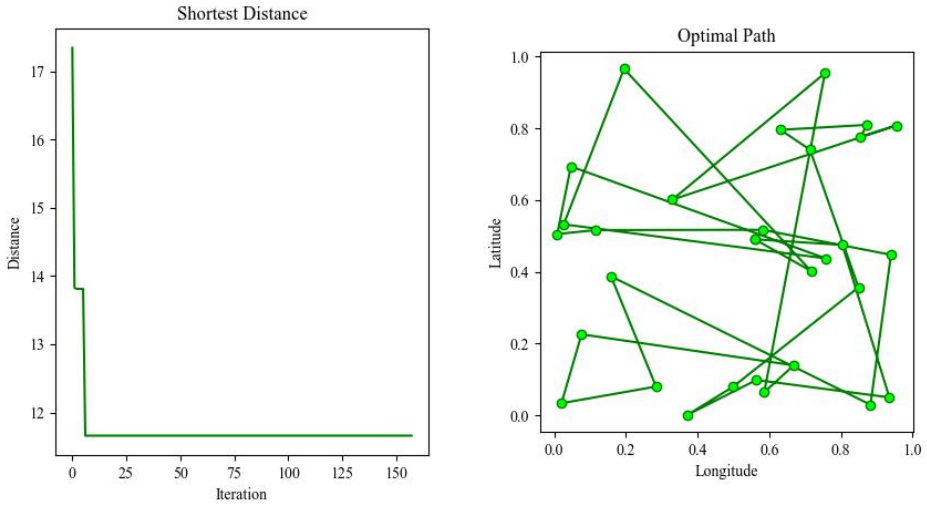
城市数	10	20	30	40	50
最短距离	3.1732	6.8538	11.6595	13.9634	19.4704
时间(s)	0.6017	1.9567	1.5859	2.5138	9.4254



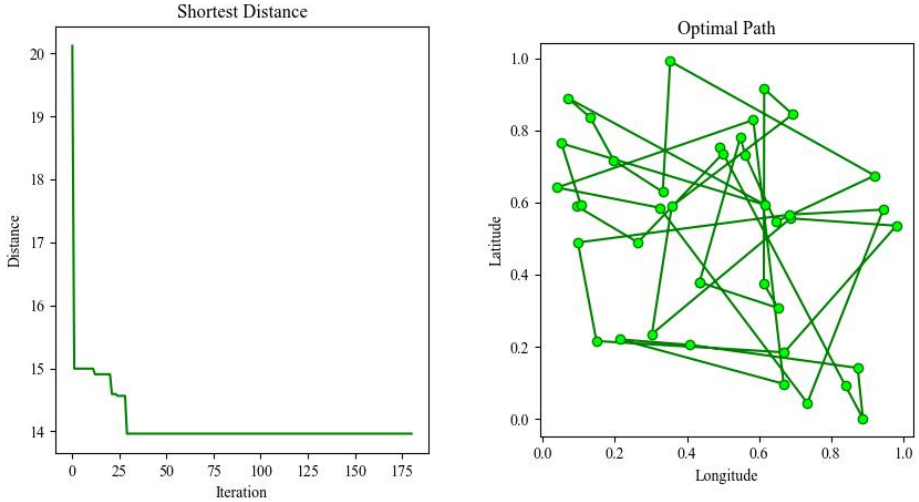
(a) 10 个城市



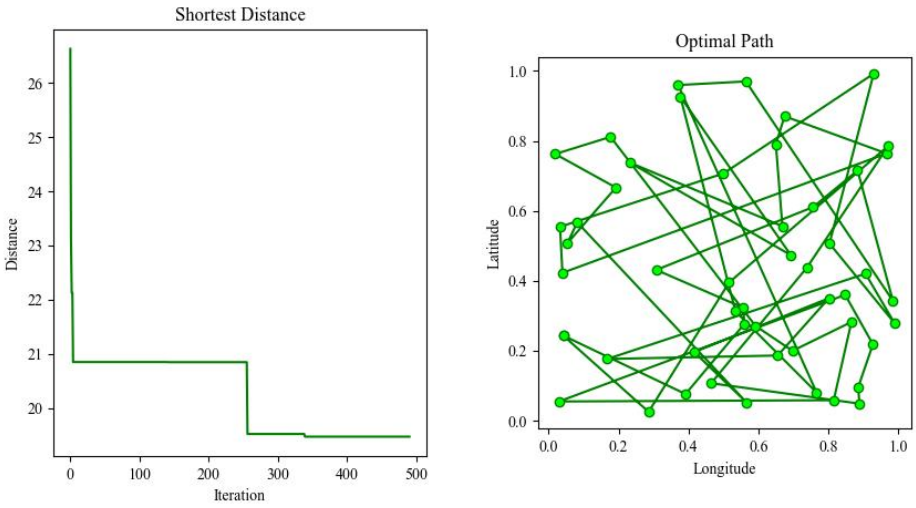
(b) 20 个城市



(c) 30 个城市



(d) 40 个城市



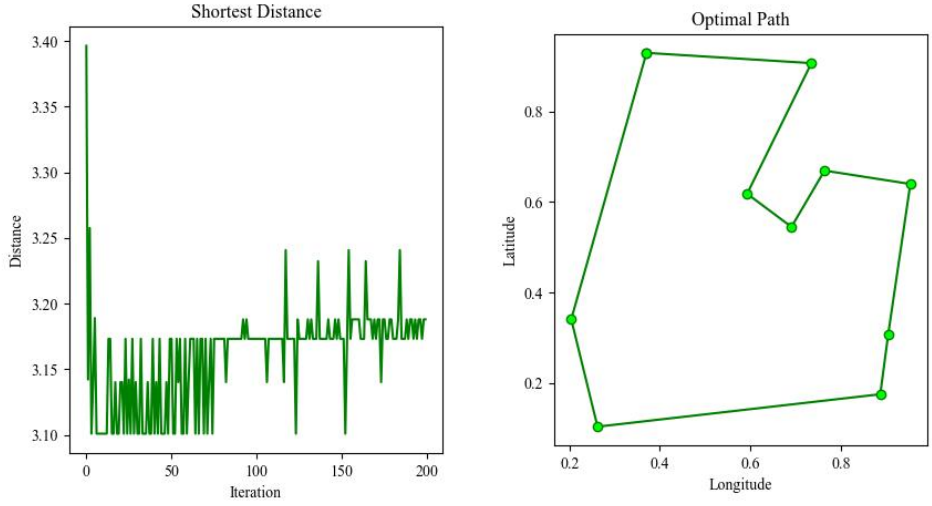
(e) 50 个城市

图 11 SA 生成的最优回路

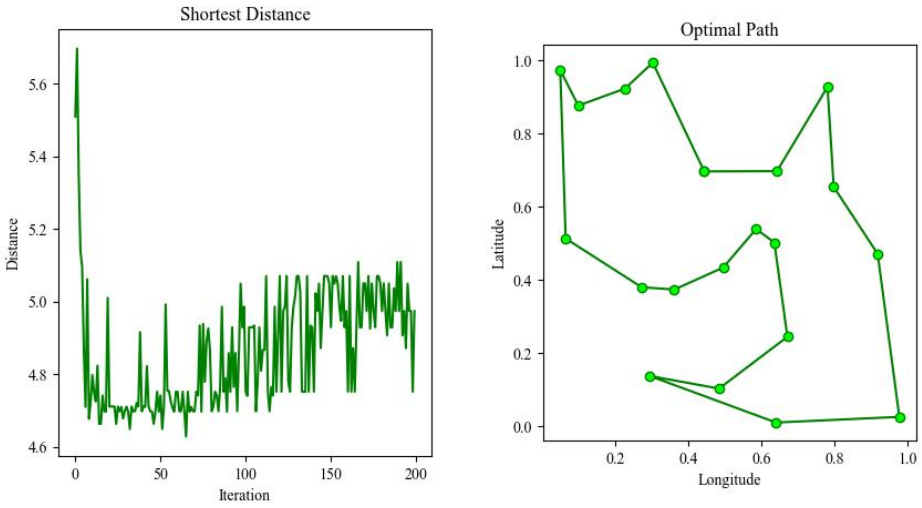
4.3.4 蚁群算法

表 12 ACO 求解 TSP 问题的结果

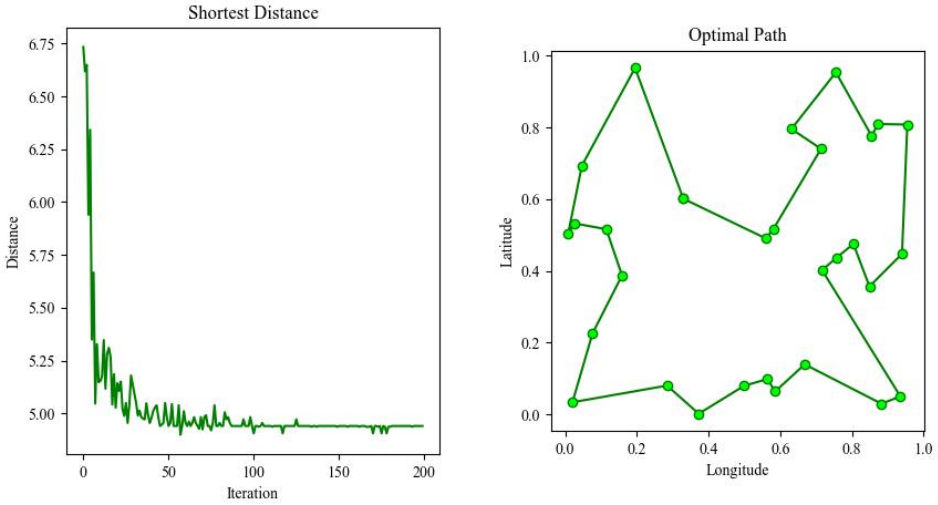
城市数	10	20	30	40	50
最短距离	3.1010	4.6276	4.8992	4.9847	6.4227
时间(s)	3.8713	9.1271	13.4935	18.9972	25.6035



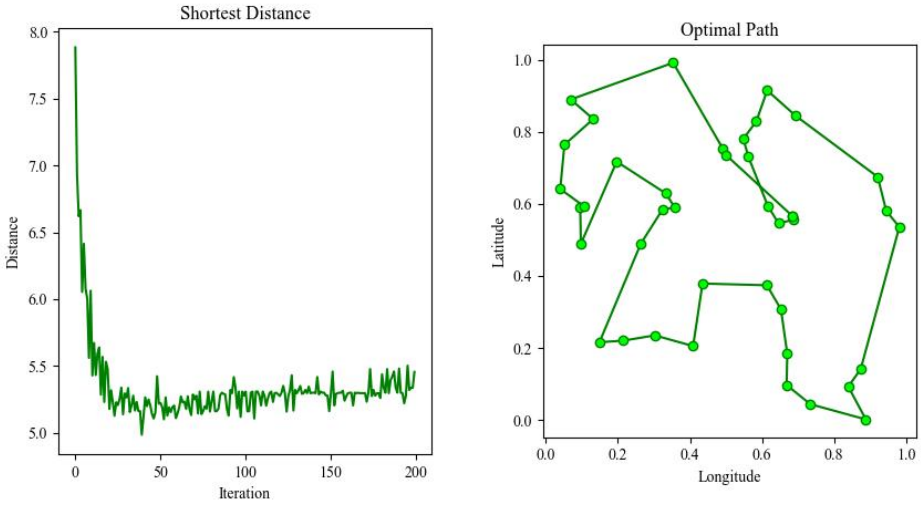
(a) 10 个城市



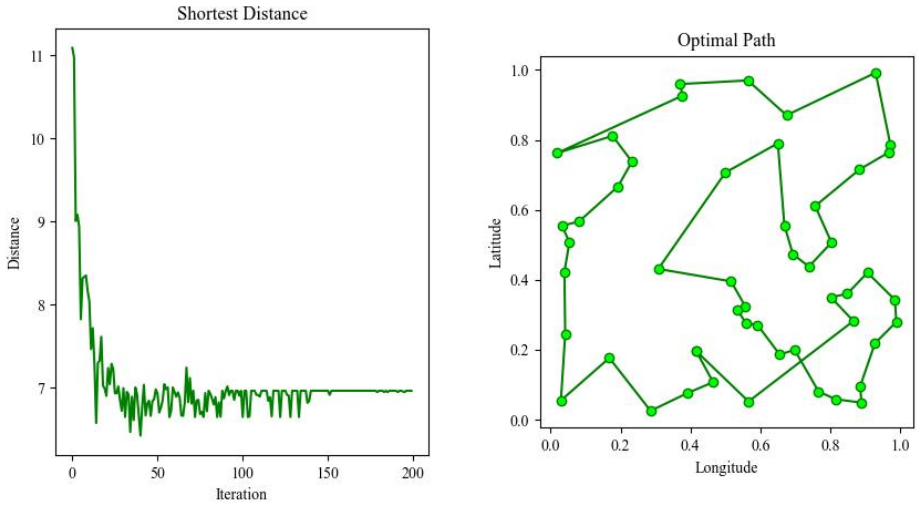
(b) 20 个城市



(c) 30 个城市



(d) 40 个城市



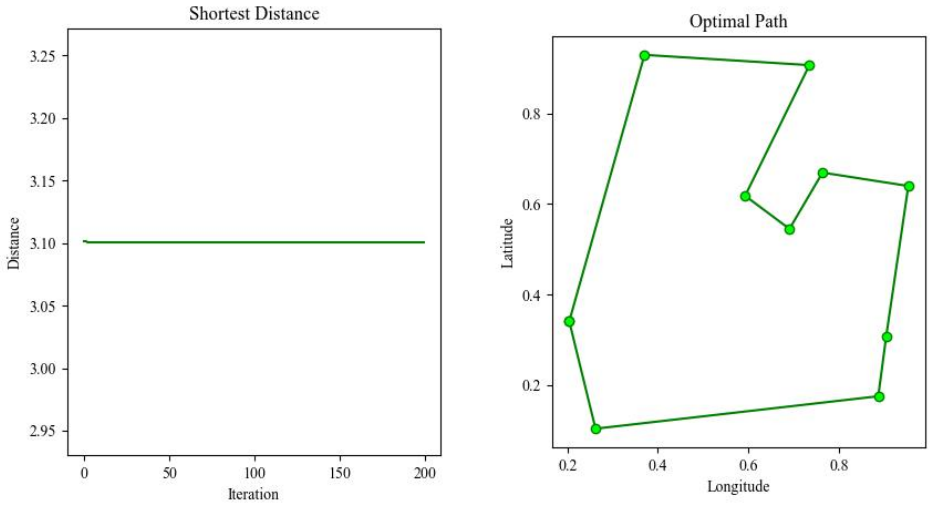
(e) 10 个城市

图 12 ACO 生成的最优回路

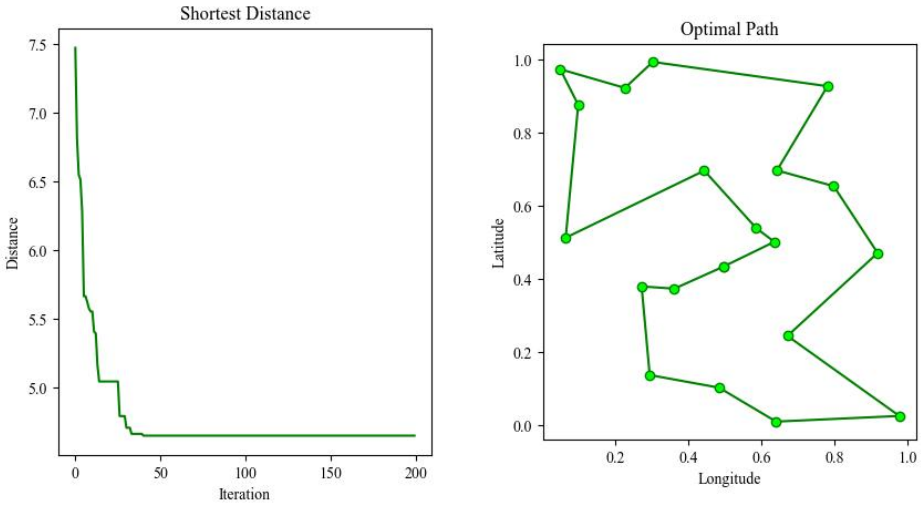
4.3.5 粒子群算法

表 13 PSO 求解 TSP 问题的结果

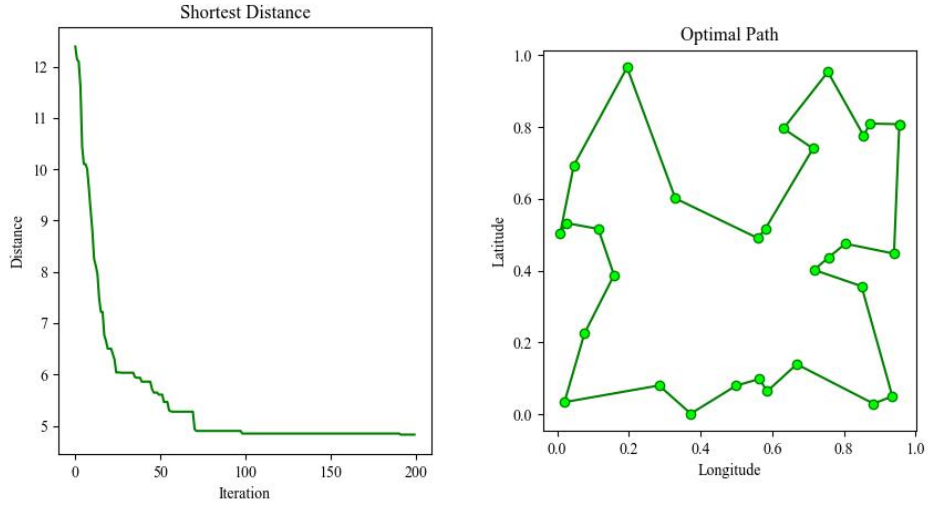
城市数	10	20	30	40	50
最短距离	2.2290	2.8778	3.8564	4.4688	5.2449
时间(s)	3.1010	4.6497	4.8271	5.2028	6.7922



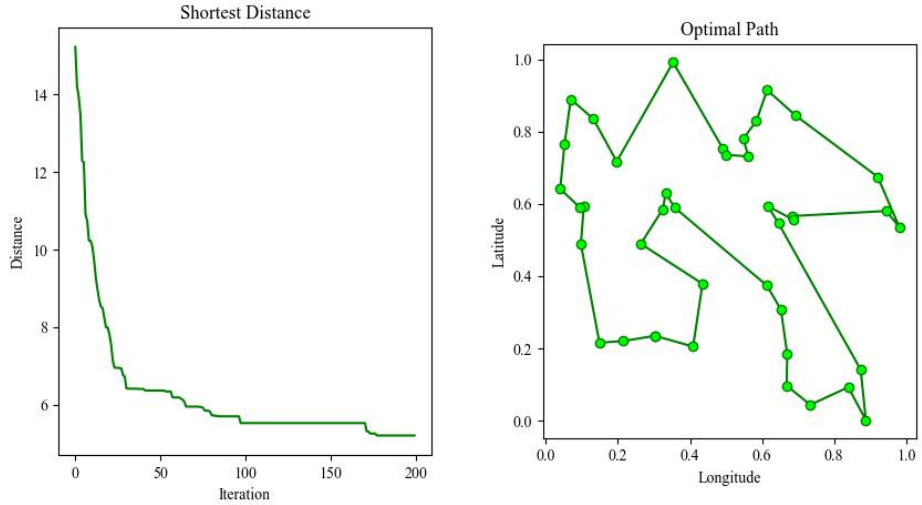
(a) 10 个城市



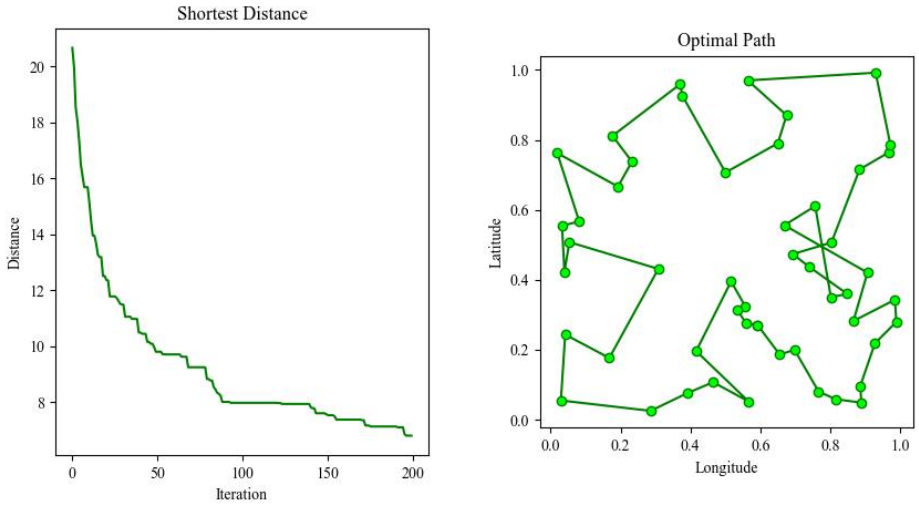
(b) 20 个城市



(c) 30 个城市



(d) 40 个城市



(e) 50 个城市

图 13 PSO 生成的最优回路

4.3.6 结果描述

分析上述实验结果可以观察到，在迭代初期，五种优化算法均快速收敛，路径距离在短时间内显著降低。随后，GA、SA、PSO 与 IA 四种优化算法缓慢的降低，而 ACO 算法不断跳出局部最优解，不断探索更短的距离。

5. 分析与总结

本文将蚁群算法、遗传算法、模拟退火算法、粒子群算法以及免疫优化五种元启发式算法应用于旅行商问题，通过实验对比研究了五种方法的优化效果、各方法在三组地图上的优化性能及其在不同城市数量的 TSP 中的优化性能。

基于本文中的实验测试，我们观察到五种优化算法在随机构造的不同规模的 TSP 实例中均能够稳定的输出优化路径。其中，遗传算法与免疫优化不仅在实现思路上相近，并且在时间与生成的最短距离两个指标上展现出了相近的优化性能。与此同时，通过五种算法之间的性能比较可以发现，在运行时间方面，GA、SA 以及 PSO 通常具有更高的时间效率；而在生成的最短距离方面，ACO 与 PSO 方法能够稳定生成距离更短的回路；综合两个指标对比分析，PSO 方法能够兼顾时间效率与最优距离两个指标。

基于本文对于 5 种优化方法的实验与分析，我们可以得出以下结论。当具体应用场景能够接受以较高的时间代价获取更优的最短距离时，可以选择 ACO 与 PSO 算法。而当应用场景更注重耗时低而不苛求生成最优的距离时，GA、SA 以及 PSO 算法更为合适。

不可忽略的是，本文所研究算法的优化性能，均受到运行环境、实验的具体参数设定以及算法属性设置等因素的影响。

参考文献

[1] Agrawal P, Abutarboush H F, Ganesh T, et al. Metaheuristic algorithms on feature selection: A survey of one decade of research (2009-2019)[J]. Ieee Access, 2021, 9: 26766-26791.

[2] Holland J H. Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence[M]. MIT press, 1992.

[3] Glover F. Future paths for integer programming and links to artificial intelligence[J]. Computers & operations research, 1986, 13(5): 533-549.

[4] Storn R, Price K. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces[J]. Journal of global optimization, 1997, 11: 341-359.

[5] J. Kennedy and R. Eberhart, "Particle swarm optimization", Proc. IEEE Int. Conf. Neural Netw. (ICNN), vol. 4, pp. 1942-1948, Nov./Dec. 1995.

[6] M. Dorigo, V. Maniezzo and A. Colorni, "Ant system: Optimization by a colony of cooperating agents", IEEE Trans. Syst. Man Cybern. B Cybern., vol. 26, no. 1, pp. 29-41, Feb. 1996.

[33] Karaboga D. An idea based on honey bee swarm for numerical optimization[R]. Technical report-tr06, Erciyes university, engineering faculty, computer engineering department, 2005.

[7] Mucherino A, Seref O. Monkey search: a novel metaheuristic search for global optimization[C]//AIP conference proceedings. American Institute of Physics, 2007, 953(1): 162-173.

[8] Kirkpatrick S, Gelatt Jr C D, Vecchi M P. Optimization by simulated annealing[J]. science, 1983, 220(4598): 671-680.

[9] Geem Z W, Kim J H, Loganathan G V. A new heuristic optimization algorithm: harmony search[J]. simulation, 2001, 76(2): 60-68.

[10] Rao R V, Savsani V J, Vakharia D P. Teaching–learning-based optimization: an optimization method for continuous non-linear large scale problems[J]. Information sciences, 2012, 183(1): 1-15.

[11] Kashan A H. League championship algorithm: a new algorithm for numerical function optimization[C]//2009 international conference of soft computing and pattern recognition. IEEE, 2009: 43-48.
[12] Wang L, Pan J, Jiao L. The immune algorithm[J]. ACTA ELECTONICA SINICA, 2000, 28(7): 96.