华中科技大学

《人工智能导论》课程设计报告

选题名称：基于遗传算法解决旅行商问题

专　　业：数据科学与大数据技术

姓　　名：

学　　号：

指导教师：冯琪

华中科技大学计算机学院

2023 年 12 月 25 日

目　　录

图　　目

表　　目

摘　　要

针对旅行商问题的求解，报告采用基础的遗传算法得到近优解。该算法模拟生物种群的进化过程，通过对种群中染色体个体的操作实现方案的优化。报告在python 环境中进行仿真实验，将得到的结果可视化，较为全面地展示了 GA 求解 TSP 的过程。

关键词：旅行商问题、遗传算法

Abstract

In order to solve the traveling salesman problem, the report uses a basic genetic algorithm to obtain a near-optimal solution. This algorithm simulates the evolutionary process of biological populations and optimizes the solution through the operation of individual chromosomes in the population. The report conducts simulation experiments in the python environment, visualizes the results, and more comprehensively demonstrates the process of solving TSP by GA.

Key words: traveling salesman problem、genetic algorithm

1　旅行商问题

1.1　背景介绍

旅行商问题（traveling salesman problem,TSP）是组合优化中的一个 NP 难题，一般描述为：给定一组城市和每一对城市之间的距离，求解从一座城市出发，访问每座城市一次后再返回初始城市的最短回路。TSP 与很多现实生活和工程问题有关，例如物流路径规划、机械臂控制优化和无人机航迹规划，具有重要的研究价值。

国内外学者对 TSP 求解方法进行了大量研究。在早期阶段，侧重找到最优解的精确算法，如整数线性规划、分支定界算法和动态规划等。然而，随着 TSP 规

模的增大，最优解空间迅速增加，这种“组合爆炸”式的搜索空间使精确算法无法在可接受时间内获得最优解。因此，许多学者开始转向启发式算法研究，在一定的计算时间内找到一个满意解或近优解。遗传算法（genetic algorithm,GA）、蚁群算法（ant colony optimization,ACO）、粒子群算法（particle swarm optimization,PSO）等启发式算法相继被用于求解 TSP。启发式算法的性能主要通过灵活性、收敛速度和稳定性等指标来衡量。GA 原理简单、容易实现，广泛应用于求解 TSP。本篇报告聚焦基础 GA 的算法的实现。

1.2 TSP 数学模型

采用图论知识，TSP 可描述为：给定一个完全图 $G=(V,E), V=\{1,2,\ldots,n\}$ 表示由顶点 i（$1 \le i \le n$）构成的集合，n 为顶点数量；$E=\{(i,j) \mid i,j \in V\}$ $(1 \le i,j \le n)$ 表示顶点之间的边集。$d(i,j)$ 表示边 (i,j) 的权重。对于对称 TSP，$d(i,j)=d(j,i)$，即从 i 到 j 的权重等于从 j 到 i 的权重。完全图中，一个顶点代表一个城市，一条边代表两个城市之间的路径，边的权重代表路径的长度、交通工具行驶成本或时间等。采用整数规划模型，TSP 的目标函数如式（1）所示。

$$\min Z=\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}x_{ij} \tag{1}$$

其中：x_{ij} 表示决策变量。

$$x_{ij}=\begin{cases}1 & \text{旅行商访问城市 } i \text{ 后再访问城市 } j\\ 0 & \text{其他}\end{cases} \tag{2}$$

约束条件表示如下：

$$\sum_{j=1}^{m} x_{ij}=1 \quad i \in V \tag{3}$$

$$\sum_{i=1}^{m} x_{ij}=1 \quad j \in V \tag{4}$$

$$\sum_{i\in S}\sum_{j\in S} x_{ij} < |S| \quad \forall S \in V \tag{5}$$

其中：式（3）与式（4）限制每个顶点仅有一条边进出；式（5）用于消除子回路；S 为顶点集 V 中所有非空子集；$|S|$ 为集合 S 包含的顶点数。

2 遗传算法

GA 根据达尔文进化论和自然选择原理设计，模拟生物种群的进化过程。针对具体问题，建立一个染色体种群，一个染色体代表问题的一个解，通过对父代染色体操作生成子代染色体，然后对子代染色体评价保留更优的染色体，从而逐步找到代表最优解的染色体。GA 求解 TSP 的一般计算步骤如下：

a) 确定编码原则，生成初始种群。对于 TSP，通常采用整数编码，即直接采用城市序号编码，按照访问城市的顺序把城市序号排列成染色体。

b) 计算种群中每条染色体的适应度。求解 TSP 的目的是寻找代表最短哈密顿圈的染色体，因此采用哈密顿圈的长度的倒数或相反数表示染色体的适应度函数。随着种群进化，染色体适应度不断变大，对应的哈密顿圈也越来越优。（实际问题中，适应度也可调整为越小越优）

c) 选择操作。主要选择方法有轮盘赌选择法与锦标赛选择法。前者依式（6）和（7）计算每条染色体的被选择概率和累计概率（N 为染色体种群规模），再根据一个随机数 r 确定要保留的染色体；后者则直接选择和保留每代种群中一定数量的高适应度的染色体。适应度越大的染色体，通过选择操作越容易被保留，而适应度小的染色体逐渐被淘汰，从而驱动种群个体往适应度增大的方向进化。

$$P_i = \frac{f(H_i)}{\sum_{i=1}^{N} f(H_i)} \tag{6}$$

$$Q_i = \sum_{j=1}^{i} P_j \tag{7}$$

d) 交叉操作。根据设定的交叉概率 p_c 选择若干父代染色体并进行交叉操作，按照交叉操作规则生成子代染色体，常用的交叉操作方法有单点交叉、顺序交叉、部分映射交叉、循环交叉等。

e) 变异操作。根据设定的变异概率 p_m，选择一定数量的父代染色体按照变异操作产生子代染色体，并评估子代染色体的适应度。常用的变异操作方法有逆序变异、交换突变、倒换突变等。变异操作用于增加种群的多样性，有助于 GA 跳出局部最优解和产生更优解。

f) 迭代终止操作。通常设定一个最大迭代次数，当算法达到最大迭代次数时，输出计算结果；或者在一定迭代次数内种群中个体的最大适应度都未变化，

则停止迭代，输出计算结果。否则，转至步骤 *b*)，重复上述计算步骤。

遗传算法的实现流程图简化如下：

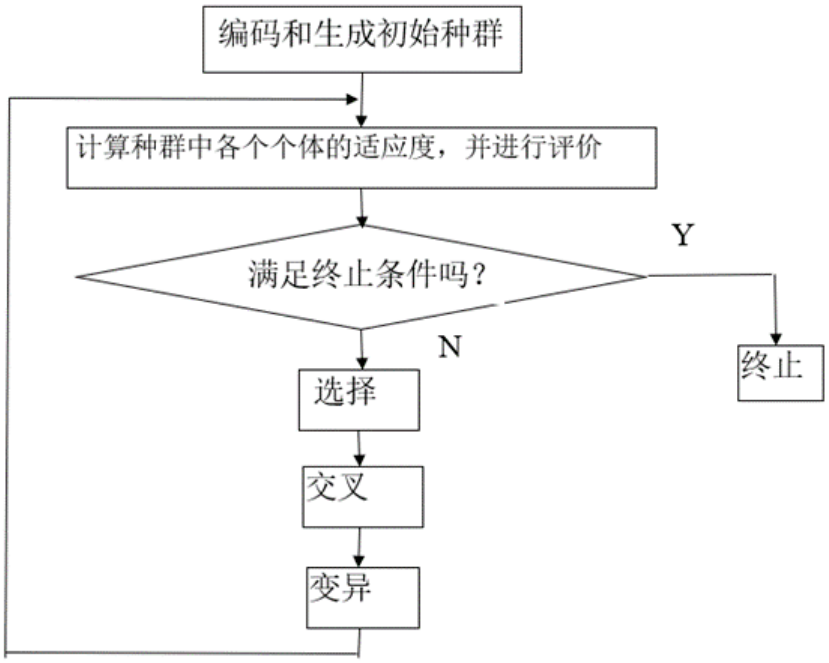

图 1　遗传算法实现简化流程图

基于这样的流程，我们可以设计一个 TSP 类来定义上述种群中发生的行为，将相关的功能的函数封装完成后将数据集传入程序入口接口即可得到我们想要的近优解。

3 仿真实验

3.1 实验相关参数及环境

本报告以中国 34 个主要城市为研究对象，在此基础上建立 TSP 模型，在 python 环境中采用 GA 对该实例进行测试。下面介绍 34 个城市的编码方式：

a) 给每个城市一个序号，如 1-> 北京，2-> 上海，3-> 广州等。

b) 用包含 n 个城市的序号的序列表示一种路线（基因序列）。

c) 数值序列中的值不重复，即每个城市只去一次。

操作系统为 Windows11，处理器为 AMD Ryzen 7 7840H，16G，运行环境为 python 3.11.4。遗传算法（GA）种群规模为 100，最大迭代次数为 500，交叉概率为 0.7，变异概率为 0.1。

实验的数据集来自中国 34 个主要城市的经纬度信息，以此作为平面坐标直接做欧氏距离计算。适应度取作总距离的倒数。

3.2 仿真结果分析

按照上述实验前的准备，在 China.csv 实例上独立运行 20 次，记录每次的运行结果，找到最优解，并结合网上找到的最优解，根据公式计算偏差率，具体公式如下：

$$\text{偏差率} = \frac{\text{最优解} - \text{已知最优解}}{\text{已知最优解}} \times 100\% \tag{8}$$

GA 具体实验结果见表 1。为了更加直观的了解 GA 求解 TSP 的具体过程，本报

表 1 仿真实验结果

测试实例	已知最优解	算法	最优解	偏差率
china.csv	156.85	GA	161.42	2.91 %

告绘制了 china.csv 实例的迭代曲线与最优路径图。如图 2 所示，我们发现曲线在一些坐标点存在跳跃现象，说明算法的鲁棒性有待提高。图 3 则给出了本报告算法求解到的最优路径图。

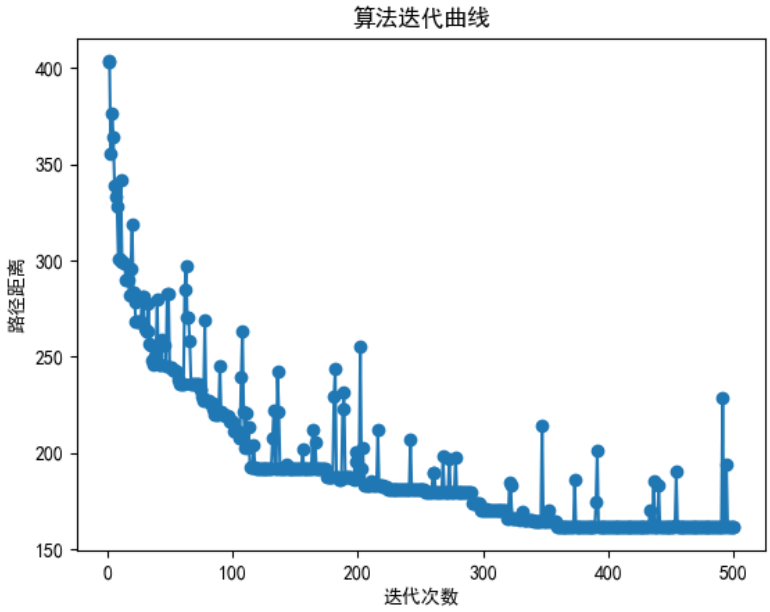

图 2　china.csv 迭代曲线

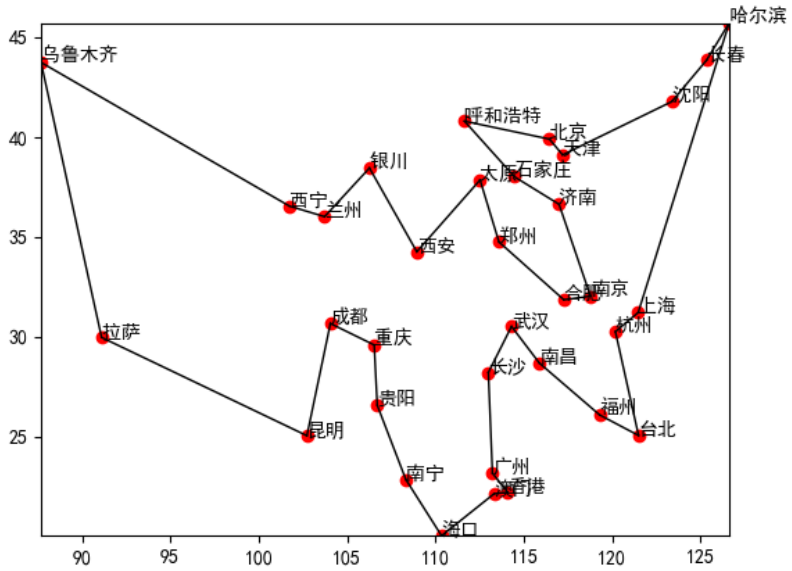

图 3　china.csv 最优路径图

4 小结与展望

由于 TSP 的许多背景知识和之前所学《离散数学》的图论内容相关。整个项目的完成思路整体上还是很流畅的，除了对于 python 的一些库的调用不太熟悉。对于 TSP，目前已经有了很多成熟的算法，因我水平有限与报告篇幅只展现了基础 GA 求解 TSP 的过程。事实上，传统遗传算法的种群初始化通常是由计算机随机生成，这样容易使初始种群的个体适应度与我们想要求得的最优适应度偏差较大，从而使算法收敛速度变慢。同时，传统遗传算法可能陷入局部最优解的陷阱，无法得到我们满意的解。

对于遗传算法的优化，目前也存在多种方案。较为流行的一种改进方法是采用基于基因库的遗传算法（GPGA）。这种算法在 GA 的基础上巧妙引入基因片段与基因库的概念，有效提高了算法的全局寻优能力，可以更快地计算出最优解。

对于这门课，这个选题的核心是让我们充分理解 TSP 背后的原理，项目中从数学逻辑到代码逻辑的框架构建让我进一步加深对初级人工智能的理解。同时，我也在项目中感受到了人工智能的逻辑之美，培养了深入探究人工智能领域的兴趣。

最后，十分感谢老师和同学的帮助，让我在人工智能领域更进了一步。

主要参考文献

[1] GitHub Repository: https://github.com/kellenf/TSP_collection

[2] GitHub Repository: https://github.com/realysy/som-tsp

[3] DataSet Source: https://github.com/chaolongzhang/tsp/tree/master/data