****

**2024 — 2025 学年第 二 学期研究生学位课程期末考试论文**

**独创声明：**

本人声明所呈交的论文是本人在任课教师指导下进行的研究工作及取得的研究成果。论文写作符合学术规范，不存在抄袭、剽窃等学术不端行为。本人已经知晓论文中若存在抄袭、剽窃情况将不能获得本门课程相应学分。

|  |  |  |  |
| --- | --- | --- | --- |
| **研究生签字** |  | **日 期** |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **学 院** | **信息科学与工程学院** | | **专 业** | **计算机技术** |
| **学 号** | **2024317115** | | **姓 名** | **董增杰** |
| **课程名称** | **算法分析与设计** | **课程编号** | | **QGC015114** |
| **论文题目** | **从DP到Transformer：TSP算法的对比分析** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **论文重复率检测结果** |  | | |
| **论文成绩** |  | | |
| **教师签字** |  | **日 期** |  |

注：本封面双面打印，附于课程论文首页。

**格式要求**

1. 论文正文不能低于3000-5000字。

2.论文应包括内容摘要：从论文中提取主要论点。

3.参考文献和注释：按每页出现顺序加注序号，标在引文结束句的右上角，如上标①。参考文献和注释均采用页下注释体例，置于当页下脚，按页依次编写序号，序号用①②③标示。具体格式为：

　　① 朱德发主编：《中国现代文学史实用教程》，济南：齐鲁书社，1999年，第58页。

②［美］韦勒克、沃伦著，刘象愚等译：《文学理论》，北京：三联书店，1984年，第56页。

③ 杨守森：《基因科学与文学艺术》，《山东师范大学学报》（人文社会科学版）2011年第4期。

④ 陈建功：《“最后一位”的执着与妥协》，《人民日报》2011年9月6日。

⑤ [德]波兰特：《文学与疾病－－比较文学研究的几个方面》，叶舒宪主编：《文学与治疗》，北京：社会科学文献出版社，1999年，第255页。

4.论文题目字体为宋体2号加粗，正文中标题为3号加粗。

5.论文正文字体为宋体4号，1.5倍行距，页边距设置均为2.5cm。

6.论文纸质版需统一用A4纸正反面打印，采用左侧装订的形式，学院留存至研究生毕业后两年。

7.本论文纸质版与查重电子版需完全一致，否则将视为不合格。

# 从DP到Transformer：TSP算法的对比分析

# 摘要

旅行商问题（TSP）是组合优化领域的典型 NP-hard 模型，在人群疏散、路径规划等场景具有重要应用价值。传统精确算法虽然能够保证最优解，但在问题规模增大时会遭遇指数级计算开销；启发式和元启发式方法能显著提升求解速度，却容易受参数设定影响并陷入局部最优。近年来，深度学习与强化学习推动的神经组合优化框架为大规模 TSP 求解带来了新的可能性。本文首先系统归纳了解决 TSP 的三大类方法——精确算法、启发式/元启发式算法以及基于学习的算法，并简要介绍了各自的求解流程。随后，我们采用随机生成与真实世界（TSPLIB）数据集，对多种代表算法进行了对比实验，从最优性差距、运行时间等指标出发，评估其性能。实验结果显示：在小规模实例中，精确算法仍占最优性优势；随着节点规模的增加，深度学习模型凭借端到端推理显著降低计算资源消耗，而元启发式方法在解质量和效率之间取得了较好的平衡。本文对TSP问题各种解决方法进行了系统化分析，本文所有方法的代码实现和实验结果可在Github中获取：https://github.com/dongzj56/Algorithm2025\_TSP。

**关键词：**旅行商问题（TSP）；精确算法；启发式算法；深度学习；神经组合优化

# 1 引言

旅行商问题（Travelling Salesman Problem，TSP）是经典的 NP难优化难题，其目标是在访问给定城市集合后回到起点，且总路程最短。该模型源于 19 世纪 Hamilton 提出的哈密尔顿回路概念[1]，并由 Menger 在 20 世纪正式抽象为最短巡回优化问题[2]。

凭借问题定义的通用性与经济意义，TSP 被视为组合优化领域最具理论与应用价值的模型之一。近年来，随着深度学习与强化学习的发展，TSP 在路径规划与调度中的研究热度不断攀升。例如，Zhang 等[3]针对复杂建筑环境的人群疏散提出基于 TSP 的最优疏散路线生成方法。在实际应用方面，TSP 及其变体已广泛服务于物流配送[4]、城市交通布局[5]、DNA 测序比对[6]等场景，展现了强大的跨领域适用性与研究潜力。

TSP传统解决方法可以分为精确算法和启发式算法两类。（1）精确算法基于全局搜索策略，可以保证找到最优解，但是随着问题规模增加，快速求解就变得不可行，精确算法解决TSP的代表性工作是动态规划[7]、分支界限法[8]和Concorde算法[9]。与精确算法不同，启发式算法兼顾解的质量与计算效率，但是往往也会受到问题规模的限制，并且依赖于模型参数的选择，需要耗费大量时间进行优化，启发式算法的代表性工作有最近邻算法[10]、局部搜索算法LKH[11]，这些方法依靠特定问题的经验规则进行搜索，扩展性有限。而元启发式算法则依靠与问题无关的全局搜索策略，如进化、群体智能等，实现路径寻优，代表性工作有遗传算法[12]、蚁群算法[13]、粒子群优化算法[14]、模拟退火算法[15]、禁忌搜索算法[16]，此外，也有将多种算法混合使用，以提高求解效率和解质量[17][18]。

传统TSP方法在小规模问题上表现良好，但计算开销大、易陷入局部最优，限制了其扩展性。为应对这些问题，近年来机器学习（ML），尤其是深度学习（DL）在TSP中的应用受到广泛关注，出现了很多基于学习的方法。受图像处理和自然语言处理的启发，深度神经网络已被用于自动学习复杂优化问题的解法，这种方法被称为神经组合优化（Neural Combinatorial Optimization, NCO），通过训练自动生成求解器，减少人工干预，与需大量资源和专业知识的传统方法（如Concorde、LKH）相比，ML求解器能以更低资源快速处理大规模实例，且在小规模TSP实验中达到与传统方法相当的解质量，仅耗十分之一的计算资源[19][20]。

目前最先进的 TSP 机器学习方法主要有（1）基于ML或DL的新型启发式优化算法[21]；（2）基于RNN的序列模型，如指针网络（Ptr-Nets）[22]；（3）基于卷积神经网络的方法，多由图像分类模型迁移而来[23]；（4）注意力机制和Transformer 模型[24][25]；（5）图神经网络模型[26]；（6）多模态深度学习的方法[27][28]。从求解流程上来看，基于ML或DL的方法大致可以分为端到端模型和迭代搜索模型两类。从模型的学习方式上则可以分为监督学习和强化学习两类，监督学习策略是让模型通过模仿最优解进行训练，而强化学习策略是通过不断试错和反馈找到最优解。

本文对各类TSP求解策略进行了系统深入的调研，分析了各类别代表方法的原理和算法过程，并在TSPLIB真实世界数据集和随机生成数据集上进行了对比实验。本文主要贡献如下：

1、TSP解法综述：系统梳理精确算法、启发式算法、元启发式算法以及 ML/DL 方法的原理和代表性算法。  
 2、提出实验测评框架：设计覆盖不同规模和节点分布的双数据集基准，使用最优性差距、运行时间等多维指标，对各类代表算法进行可复现对比。

3、基于实验结果，分析各算法在扩展性、计算效率、解质量上的表现，为后续 TSP 及更广泛组合优化研究提供路线指引。

文章结构安排如下：第2节对TSP问题进行公式化描述，第3节解释用于实验的数据集，第4节简单介绍几种代表性算法的实现过程，第5节给出实验设计与结果分析，第6节讨论。

# 2 问题公式化

给定一组个城市及任意两城市间的旅行代价（或距离、时间）矩阵 ，其中，。在对称TSP中，；在非对称 TSP 中则不必满足该条件。要求找到一条经过每座城市恰好一次并最终回到出发城市的哈密顿回路（Hamiltonian cycle），使得沿途总代价最小，目标函数可以表示为：

TSP问题的约束：①基本约束：节点的度为2，表示每个城市恰好进出各一次：子回路消除约束：，即对任意真子集S（大小至少2且不包含全部结点），若仅在S内连边，最多只能形成条弧，从而阻止独立子回路的产生。

# 3 数据集

为了对各类TSP解法进行横向、纵向对比，使用两个不同的数据集进行实验，一个是真实世界的数据集TSPLIB，用于横向对比各类算法的效果，另一个是随机生成的数据集，主要用于纵向对比各个算法在不同规模TSP上的表现。

## 3.1真实世界数据集

本文实验部分使用真实世界的TSP数据集（https://github.com/pdrozdowski/TSPLib.Net），该数据集对TSPLIB95库中的对称TSP问题113个实例进行了系统化打包，节点规模覆盖 14 至 85 900 个城市，既包含经典的小规模基准（如 eil51、st70、kroA100），也囊括中大型难例（如 pr2392、rl11849、usa13509、pla85900）。每个实例均提供 .tsp 描述文件，部分实例有对应的最佳已知或最优巡回 .opt.tour，距离类型使用二维欧氏距离（EUC\_2D）。

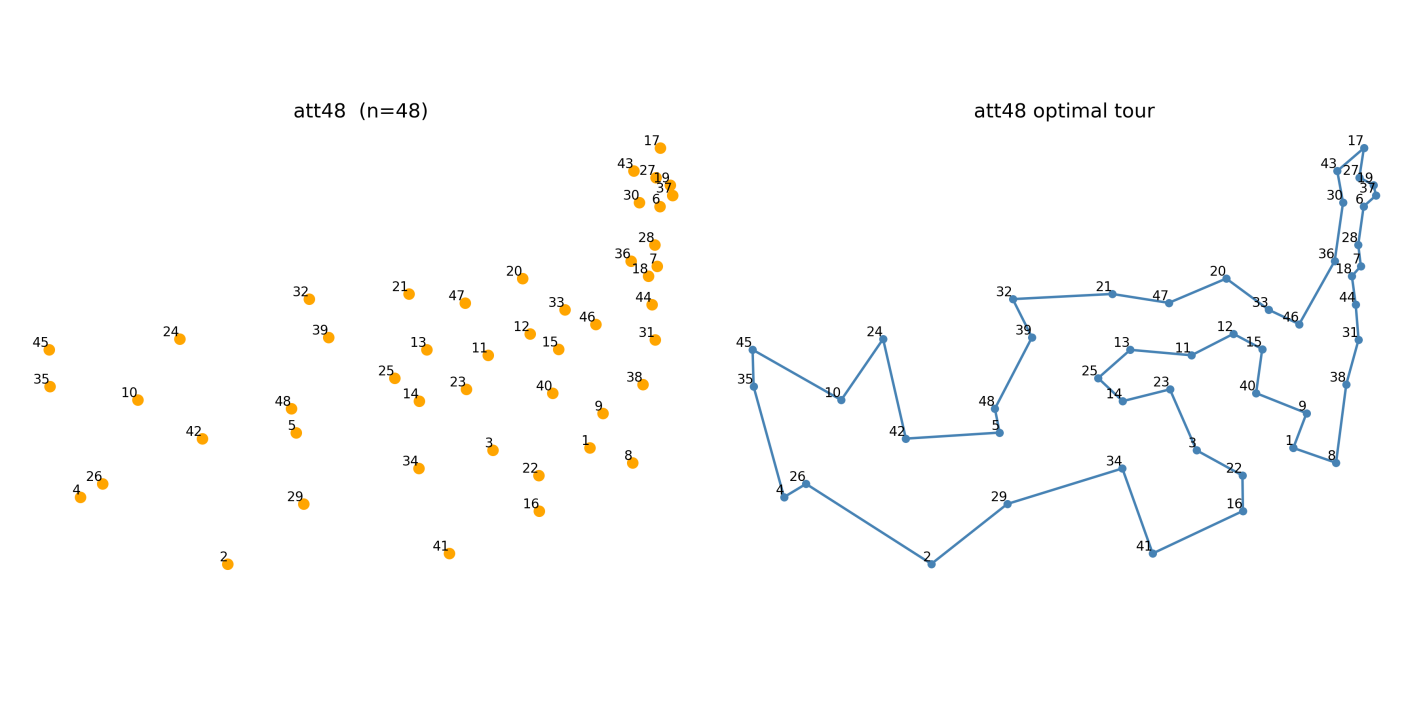


图1：真实世界数据集TSPLIB-实例att48及其最优解

## 3.2随机生成数据集

为对比各类算法在不同TSP规模下的性能，本文参考Xiao Y等人[29]的实验方案，按照结点数（城市个数）将实例划分为7个规模段，并为每段生成若干TSP样本。需要注意的是，完整数据集只是用于训练深度学习方法，而对于动态规划、遗传算法等方法，使用完整数据集太费时间，只需要从中取一部分TSP实例，计算得到平均指标就可以用于比较。生成的数据集如下表所示：

表1：生成数据集的问题规模和样本量

|  |  |  |  |
| --- | --- | --- | --- |
| 数据规模 | 城市个数 | 实例数 | 环境大小 |
| TSP-20 | 20 | 200 | 45×45 |
| TSP-50 | 50 | 200 | 71×71 |
| TSP-100 | 100 | 200 | 100×100 |
| TSP-500 | 500 | 200 | 224×224 |
| TSP-1000 | 1000 | 100 | 316×316 |
| TSP-5000 | 5000 | 30 | 707×707 |
| TSP-10000 | 10000 | 20 | 1000×1000 |

对于每个实例，在不同区间内独立均匀采样n个点，区间长度按照结点数平方根缩放。输出为标准的.tsp文件，与TSPLIB数据集文件格式一致。为保证结果可复现，设置数据生成器的全局随机种子base\_seed = 3407，然后为每个实例创建独立的局部随机数生成器，这样可以保证实例间互不干扰，即使调整生成顺序或采用并行，仍能复现相同坐标。

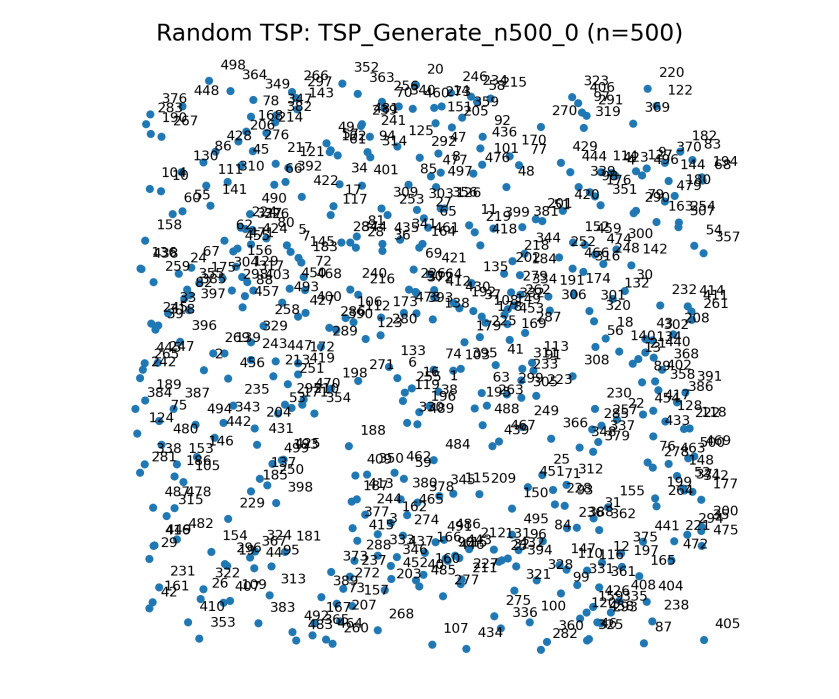
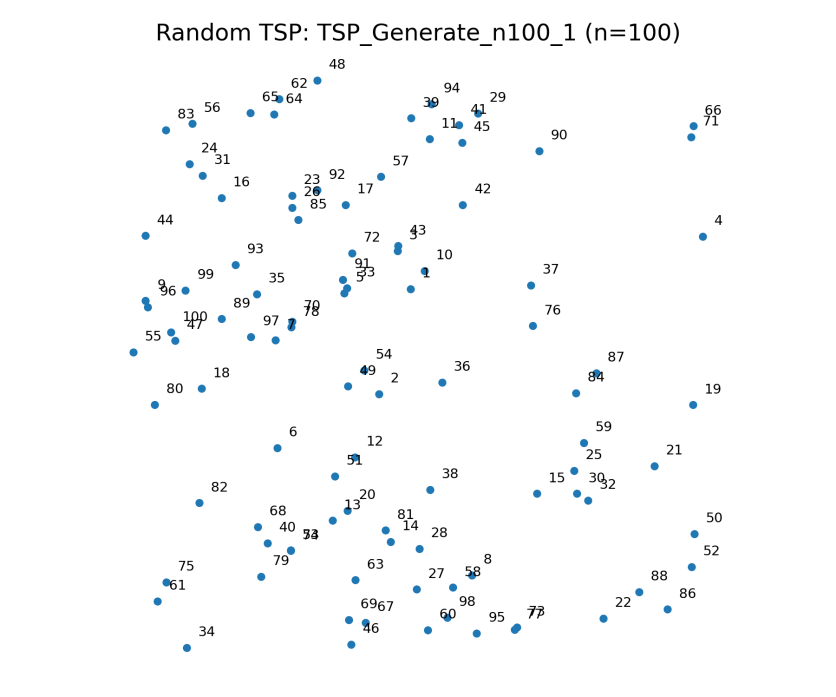


图2：生成数据集实例（左侧规模100，右侧规模500）

# 4 方法

当前，TSP的主流求解策略可以大致划分为精确算法合近似算法两大类，其中近似算法又可细分为启发式算法、元启发式算法和基于学习的方法。各类方法在解的最优性、复杂度、扩展性和实际应用表现存在显著差异。为了更直观地展示不同方法，本文将主要算法的分类、基本思想与特点整理如下表所示：

表2：TSP问题所有解法和特点及代表性算法

|  |  |  |
| --- | --- | --- |
| 算法范畴 | 特点 | 代表算法 |
| 精确算法 | 利用子集状态转移等方法进行系统搜索，保证全局最优，时间复杂度指数级，计算代价高，适用于小规模TSP问题。 | \*\*动态规划法[7] |
| 分支界限法[8] |
| \*\*Concorde算法[9] |
| 启发式算法 | 基于局部构造与改进策略，速度快、实现简单，但易陷入局部最优，解的质量不稳定，适用于初始解生成或快速近似。 | 最近邻算法[10] |
| 贪心搜索算法 |
| 局部搜索算法[11] |
| 元启发式算法 | 模拟自然智能机制进行全局搜索，具有跳出局部最优的能力，适应性强，但是参数选择敏感，调优复杂。 | \*\*遗传算法[12] |
| 蚁群算法[13] |
| 模拟退火算法[15] |
| 禁忌搜索算法[16] |
| \*\*粒子群算法[14] |
| 学习型算法 | 使用神经网络学习并预测解，推理快速，端到端建模能力和泛化性强，但是对训练数据要求高，训练代价大。 | 指针网络[22] |
| \*\*Transformer[24] |
| \*\*图神经网络[26] |

注：带有\*\*标记的，是在两个数据集上进行了对比实验的方法

下面将分别介绍各种TSP问题求解算法的原理和实现过程，在4.实验部分，将详细展示各种方法在TSPLIB数据集上解决不同规模TSP问题的性能表现，并进行对比。

## 4.1精确算法

精确算法是指通过严格数学推导或系统化搜索，在有限时间内保证找到问题的全局最优解的算法范畴。其核心优势在于确定性与最优性，这一点与启发式或近似算法形成鲜明对比——后者通常只能快速返回高质量但无法证明最优性的解。然而，精确算法的计算复杂度往往随问题规模呈指数级增长。例如在旅行商问题（TSP）中，经典动态规划（Held-Karp）求解的时间复杂度约为。当城市数 𝑛 增大时，所需时间与内存将急剧膨胀，导致大型实例难以在现实时间内求解。因此，精确算法更适用于小规模场景（如几十至数百城市的 TSP、变量较少的资源分配问题等），在这些场景下它能为决策者提供可验证的最优方案。

### 4.1.1动态规划

动态规划是一类通过将原问题划分为规模更小的子问题，并通过子问题的最优解构建全局最优解的方法。针对旅行商问题（TSP），Held 和 Karp 提出了一种基于子集状态转移的动态规划解法，被称为 Held–Karp 算法[7]。该算法显著降低了对称 TSP 问题的计算复杂度，是精确求解小规模 TSP 问题的代表性方法之一。下面简介DP算法求解TSP问题的思路：

定义距离矩阵，表示城市到的距离。算法的核心思想在于构建函数，表示从起点城市出发，访问完城市集合，最后停在城市的最短路径长度。该状态函数的递推关系如下：

（1）边界条件：

；

（2）状态转移方程：

（3）最终目标：

该算法通过自底向上计算所有可能子集状态并组合得到最终解，其时间复杂度为，空间复杂度为。虽然理论上算法的复杂度仍为指数级，但与完全枚举的相比已显著优化。因此，Held–Karp 算法被广泛用于小规模 TSP 问题的精确求解，尤其在城市数量 𝑛≤1000的情形下可在合理时间内完成计算。此外，其状态转移结构也为后续分支限界和启发式方法的下界估计提供了理论基础。

### 4.1.2 Concorde算法

Concorde TSP Solver 是目前国际上最强大、最权威的TSP精确求解器之一，由David等人开发[9]，该算法可在合理时间内求解上万个城市的旅行商问题，并严格保证最优解。

为求解TSP问题的最优路径，Concorde算法采用了基于整数线性规划的精确方法框架，即分支-割平面法（Branch-and-Cut）。其基本思想是将TSP建模为一个0-1整数规划问题，通过线性松弛、剪枝和剪切平面等手段，逐步收紧可行解空间，从而精确获得全局最优解。

## 4.2 启发/元启发式

启发式与元启发式算法是求解TSP问题等NP难问题中非常重要的两类方法。相较于传统精确算法（如动态规划、分支界限法），它们更注重搜索效率与可扩展性，通过经验规则或模拟自然机制，快速得到近似最优解，尤其适合大规模实例或实时系统。

### 4.2.1遗传算法

遗传算法（Genetic Algorithm, GA）是一种基于自然选择和遗传机制的启发式搜索方法，适用于解决NP难问题如旅行商问题（TSP）[12]。它通过模拟“适者生存”的进化过程，对问题解空间进行全局搜索，寻找近似最优路径。

在TSP中，个体（染色体）表示一次完整的城市访问顺序，即一个排列，其中，，表示访问顺序。路径长度（目标函数）为：

优化目标可以表示为：，适应度函数通常为路径长度的倒数，用于转化为最大化问题：，遗传算法解决TSP问题，流程如下：

①初始化：随机生成一定数量的城市排列，每个染色体为一个TSP路径排列，构成初始种群Population0；

②适应度评估：对每个个体计算路径长度L(P) 并求其适应度f(P)，使路径越短，其适应度越高；

③选择：使用轮盘赌方法，选出适应度高（路径短）的个体作为父代；

④交叉：对父代进行有序交叉（Order Crossover, OX）、部分匹配交叉（PMX）等操作，生成子代个体，交叉操作确保产生的新路径依然为合法的TSP解（即包含所有城市且不重复）；

⑤变异：对子代执行交换变异操作，如随机选择两个城市位置进行交换，从而增加解的多样性并跳出局部最优；

⑥环境选择与更新：子代与父代合并，选出适应度最优的若干个体构成下一代;

⑦终止条件：若达到最大迭代次数或最优路径稳定不变，则终止算法，输出最优路径。

### 4.2.2粒子群算法

粒子群优化算法（PSO）是一种群体智能优化算法，最初由 Eberhart 和 Kennedy 提出。它模拟鸟群觅食行为，通过个体（粒子）间的信息共享和自主学习，在解空间中寻找最优解。由于TSP是一个离散组合优化问题，经典PSO需进行适当的离散化改造[14]。

在TSP中，每个粒子表示一个完整的城市访问路径，即一个排列，其中，，表示访问顺序。目标函数和适应度函数都与遗传算法类似。PSO解决TSP优化流程如下：

①初始化：随机生成若干个粒子（路径排列），并设定其个体最优解 pBest 及群体最优解 gBest；

②粒子位置与速度定义：位置为当前路径排列（城市访问顺序），速度表示一种到达新路径的交换操作序列（如“交换第3和第5个城市”）；

③更新：粒子位置通过交换操作更新：；

④适应度评估与更新：对每个粒子路径计算适应度，更新对应的pBest，若优于当前全局最优则更新 gBest；

⑤终止条件：达到最大迭代次数或收敛稳定，输出最优路径和长度。

## 4.3基于学习的方法

### 4.3.1图神经网络

TSP 本质上是在一张完全加权图上寻找最小哈密顿环，决策只与节点间的相对拓扑和边权有关，而与节点编号或输入顺序无关。图神经网络以消息传递为核心，每一层用可学习函数汇聚邻接信息，进而更新节点表示：

图神经网络的消息传递机制天然满足TSP特性，同时共享参数和让模型可以在任意规模的图上扩展。

Joshi 等人[26]就是基于上述思想，提出将 TSP 映射为一张加权完全图，并利用图卷积网络（GCN）直接在图结构上学习一套端到端的逐边评分策略。

### 4.3.2 注意力机制

随着Transformer 等深度学习框架在 NLP 与 CV 领域的卓越表现，有研究者受到启发将其引入 TSP 求解，凭借全局自注意力机制，可以对城市成对距离进行建模。Kool 等人[24]首次在TSP问题中引入注意力机制，用多头自注意力和指针架构直接学习 TSP 的巡回策略，这一框架证明了全局自注意力+策略梯度可在无监督条件下自动学习高质量 TSP 近似解。Bresson等人[25]则把完整的Transformer 直接迁移到 TSP，用多层自注意力对全图编码，再在解码端配合束搜索逐点“指针”选择城市；训练仍靠策略梯度，不依赖最优解标签。与传统方法相比，该方法免除手工规则设计，推理高效且易迁移至其他组合优化问题。

Transformer解决TSP过程可以视作一次端到端的策略学习。设为n座城市的二维坐标，经线性映射层和位置编嘛得到输入嵌入。随后使用多层自注意力机制迭代计算：

最终得到对全图上下文感知的结点表征：

在解码阶段，由已走序列得到查询向量*q*，对尚未访问的结点计算指针得分：

进一步地，按策略采样下一座城市，得到完整的巡回，其代价为：

模型用策略梯度更新，损失函数可以表示为：

下图给出了使用Transformer解决TSP的完整框架：

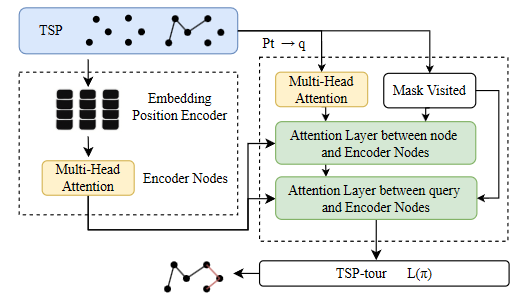


图3：Transformer解决TSP框架示意图

# 5 实验

## 5.1 评价指标

（1）路径长度：模型求得的路线长度；

（2）推理（求解）时间（ms）：模型完成一次求解的时间开销（相同计算环境下）；

（3）相对误差gap（%）：模型求得路线长度相对于最优解的百分比差异，Gap的值越小表示TSP路径规划性能越优，0%表示达到最优：

其中，最优解一般为LKH-3近似最优解，所用TSPLIB真实数据集，只有部分TSP实例有最优解，因此这个指标只用于最终验证。

## 5.2 纵向对比实验及分析

所有方法均在相同环境下运行，CPU为AMD Ryzen 7 8845HS，8核16线程，显卡为NVIDIA RTX 3090，内存24G。所有方法均使用Python 3.10实现，深度学习的框架为PyTorch 2.4.1。

### 5.2.1动态规划

Held M等人[7]使用子集-终点状态表示，利用最优子结构+重叠子问题，优化了时间复杂度，并且还使用1-tree下界进行强剪枝，使大规模TSP也能在有限时间得到结果。使用该DP方法解决不同规模TSP的结果如下表所示：

表3：DP在不同规模TSP问题上的表现

|  |  |  |  |
| --- | --- | --- | --- |
| 问题规模 | 测试样本量 | 平均路径长度 | 平均用时（ms） |
| TSP-20 | 200 | 179.46 | 2.15 |
| TSP-50 | 200 | 433.23 | 10.59 |
| TSP-100 | 200 | 855.24 | 38.87 |
| TSP-500 | 100 | 4150.17 | 900.71 |
| TSP-1000 | 50 | 8276.01 | 6408.36 |
| TSP-5000 | 10 | 41158.46 | 153896.26 |
| TSP-10000 | 5 | 81538.98 | 615216.14 |

### 5.2.2 Concorde算法

表4：Concorde在不同规模TSP问题上的表现

|  |  |  |  |
| --- | --- | --- | --- |
| 问题规模 | 测试样本量 | 平均路径长度 | 平均用时（ms） |
| TSP-20 | 200 | 176.45 | 120.31 |
| TSP-50 | 200 | 437.21 | 160.86 |
| TSP-100 |  | 887.55 | 312.73 |
| TSP-500 |  | 4437.81 | 3289.22 |
| TSP-1000 |  | 8905.04 | 11688.28 |
| TSP-5000 |  | 43435.35 | 248140.62 |
| TSP-10000 |  |  |  |

### 5.2.3遗传算法

遗传算法：所用遗传算法，种群大小为50，迭代次数为100，选择比例为0.2，变异率为0.05。

表5：GA在不同规模TSP问题上的表现

|  |  |  |  |
| --- | --- | --- | --- |
| 问题规模 | 测试样本量 | 平均路径长度 | 平均用时（ms） |
| TSP-20 | 200 | 176.45 | 120.31 |
| TSP-50 | 200 | 437.21 | 160.86 |
| TSP-100 | 200 | 887.55 | 312.73 |
| TSP-500 | 100 | 4437.81 | 3289.22 |
| TSP-1000 | 50 | 8905.04 | 11688.28 |
| TSP-5000 | 10 | 43435.35 | 248140.62 |
| TSP-10000 | 5 | 87524.07 | 940031.25 |

### 5.2.4粒子群算法

粒子群算法：所用PSO算法，粒子个数为50，迭代次数为100，惯性权重为0.5，个体和群体学习因子均为2，由于TSP问题将PSO进行了离散化，所以不需要设置最大速度限制参数。

表6：PSO在不同规模TSP问题上的表现

|  |  |  |  |
| --- | --- | --- | --- |
| 问题规模 | 测试样本量 | 平均路径长度 | 平均用时（ms） |
| TSP-20 | 200 | 180.99 | 337.42 |
| TSP-50 | 200 | 447.72 | 695.78 |
| TSP-100 | 200 | 896.41 | 929.22 |
| TSP-500 | 100 | 4440.86 | 11691.15 |
| TSP-1000 | 50 | 9012.52 | 37068.75 |
| TSP-5000 | 10 | 43735.75 | 632328.12 |
| TSP-10000 | 5 |  |  |

### 5.2.5注意力机制

### 5.2.6图神经网络

## 5.3横向对比实验及分析

|  |  |  |  |
| --- | --- | --- | --- |
| TSP实例 | 城市数量 | 路径长度 | 推理用时（ms） |
| TSP-20 | 200 | 180.99 | 337.42 |
| TSP-50 | 200 | 447.72 | 695.78 |
| TSP-100 | 200 | 887.55 | 312.73 |
| TSP-500 | 100 |  |  |
| TSP-1000 | 50 | 8905.04 | 11688.28 |
| TSP-5000 | 10 | 43435.35 | 248140.62 |
| TSP-10000 | 5 |  |  |

# 6 讨论

本文总共实现了六种TSP问题解法，涵盖精确算法、启发式算法、基于学习的算法三大类，在实验部分分横向和纵向详细展示了各种算法在不同规模TSP问题求解的表现，并给出了合理的实验结果分析。

本文仍存在局限，包括

# 参考文献

1. Applegate, David L. The traveling salesman problem: a computational study. Vol. 17. Princeton university press, 2006.
2. Schrijver A. On the history of combinatorial optimization (till 1960)[J]. Handbooks in operations research and management science, 2005, 12: 1-68.
3. Zhang C, Liu H, Li W. An exploration-driven framework for path planning in complex buildings using improved MADDPG[J]. Journal of Building Engineering, 2025, 107: 112626.
4. Larni-Fooeik A, Ghasemi N, Mohammadi E. Insights into the application of the traveling salesman problem to logistics without considering financial risk: A bibliometric study[J]. Manage. Sci. Lett, 2024, 14(3): 189-200.
5. Mostafa S M, Habashy S M, Salem S A. A New Framework for Multi-objective Route Planning in Smart Cities[C]//International Conference on Advanced Intelligent Systems and Informatics. Cham: Springer International Publishing, 2022: 824-837.
6. Wang Z C, Liang K, Bao X G, et al. A novel algorithm for solving the prize collecting traveling salesman problem based on DNA computing[J]. IEEE Transactions on NanoBioscience, 2023, 23(2): 220-232.
7. Held M, Karp R M. A dynamic programming approach to sequencing problems[J]. Journal of the Society for Industrial and Applied mathematics, 1962, 10(1): 196-210.
8. Balas E. Branch and bound methods[J]. The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization, 1984.
9. Applegate D, Bixby R, Chvatal V, et al. Concorde TSP solver[EB/OL].(2006-3)
10. Rosenkrantz D J, Stearns R E, Lewis, II P M. An analysis of several heuristics for the traveling salesman problem[J]. SIAM journal on computing, 1977, 6(3): 563-581.
11. An Analysis of Several Heuristics for the Traveling Salesman Problem
12. Brady R M. Optimization strategies gleaned from biological evolution[J]. Nature, 1985, 317(6040): 804-806.
13. Dorigo M, Gambardella L M. Ant colonies for the travelling salesman problem[J]. biosystems, 1997, 43(2): 73-81.
14. Onwubolu G C, Babu B V, Clerc M. Discrete particle swarm optimization, illustrated by the traveling salesman problem[J]. New optimization techniques in engineering, 2004: 219-239.
15. Kirkpatrick S, Gelatt Jr C D, Vecchi M P. Optimization by simulated annealing[J]. science, 1983, 220(4598): 671-680.
16. Knox J. Tabu search performance on the symmetric traveling salesman problem[J]. Computers & Operations Research, 1994, 21(8): 867-876.
17. Dewantoro R W, Sihombing P. The combination of ant colony optimization (ACO) and tabu search (TS) algorithm to solve the traveling salesman problem (TSP)[C]//2019 3rd International Conference on Electrical, Telecommunication and Computer Engineering (ELTICOM). IEEE, 2019: 160-164.
18. Khan I, Pal S, Maiti M K. A hybrid PSO-GA algorithm for traveling salesman problems in different environments[J]. International journal of uncertainty, fuzziness and knowledge-based systems, 2019, 27(05): 693-717.
19. Joshi C K, Cappart Q, Rousseau L M, et al. Learning the travelling salesperson problem requires rethinking generalization[J]. Constraints, 2022, 27(1): 70-98.
20. Liu S, Zhang Y, Tang K, et al. How good is neural combinatorial optimization? A systematic evaluation on the traveling salesman problem[J]. IEEE Computational Intelligence Magazine, 2023, 18(3): 14-28.
21. Mele U J, Gambardella L M, Montemanni R. A new constructive heuristic driven by machine learning for the traveling salesman problem[J]. Algorithms, 2021, 14(9): 267.
22. Vinyals O, Fortunato M, Jaitly N. Pointer networks[J]. Advances in neural information processing systems, 2015, 28.
23. Miki S, Yamamoto D, Ebara H. Applying deep learning and reinforcement learning to traveling salesman problem[C]//2018 international conference on computing, electronics & communications engineering (ICCECE). IEEE, 2018: 65-70.
24. Kool W, Van Hoof H, Welling M. Attention, learn to solve routing problems![J]. arxiv preprint arxiv:1803.08475, 2018.
25. Bresson X, Laurent T. The Transformer Network for the Traveling Salesman Problem[EB/OL]. arXiv:2103.03012, 2021-03-04[2025-06-21]. <https://doi.org/10.48550/arXiv.2103.03012>
26. Joshi C K, Laurent T, Bresson X. An efficient graph convolutional network technique for the travelling salesman problem[J]. arxiv preprint arxiv:1906.01227, 2019.
27. Pan X, \*\* Y, Ding Y, et al. H-tsp: Hierarchically solving the large-scale traveling salesman problem[C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2023, 37(8): 9345-9353.
28. Ye H, Wang J, Liang H, et al. Glop: Learning global partition and local construction for solving large-scale routing problems in real-time[C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2024, 38(18): 20284-20292.
29. Xiao Y, Wang D, Cao R, et al. GELD: A Unified Neural Model for Efficiently Solving Traveling Salesman Problems Across Different Scales[J]. arxiv preprint arxiv:2506.06634, 2025.