Advances in Applied Mathematics 应用数学进展, 2025, 14(5), 40-52
Published Online May 2025 in Hans. https://www.hanspub.org/journal/aam
https://doi.org/10.12677/aam.2025.145231

Hans汉斯

基于深度强化学习的3D-TSP智能路径规划算法研究

周　鑫，周林华*

长春理工大学数学与统计学院，吉林 长春

收稿日期：2025年4月8日；录用日期：2025年5月2日；发布日期：2025年5月9日

摘　要

随着人工智能技术的快速发展，深度强化学习在复杂优化问题中展现出巨大的潜力。三维旅行商问题作为经典路径规划问题的扩展，具有更高的复杂性和实际应用价值，但其大规模动态环境下的实时求解仍面临挑战。传统方法存在计算复杂度高的问题，而启发式算法则受限于求解精度和泛化能力。本文提出一种基于图Transformer和深度强化学习的智能路径规划算法，通过编码器捕捉三维图结构的节点位置关系生成嵌入表示，结合解码器的注意力机制和序贯决策特性构建路径规划策略，并采用强化学习Reinforce算法优化策略。实验结果表明，所提算法在求解时间上显著优于传统方法和启发式算法，尤其在大规模问题中的优势更为明显。此外，模型通过贪婪解码与采样解码策略的权衡，兼顾了求解效率与精度。该方法为无人机路径规划、智能制造等领域的大规模动态路径优化问题提供了高效解决方案，具有重要的理论与应用价值。

关键词

三维旅行商问题，深度强化学习，图Transformer，路径规划，注意力机制

Research on 3D-TSP Intelligent Path Planning Algorithm Based on Deep Reinforcement Learning

Xin Zhou, Linhua Zhou*

School of Mathematics and Statistics, Changchun University of Science and Technology, Changchun Jilin

Received: Apr. 8th, 2025; accepted: May 2nd, 2025; published: May 9th, 2025

*通讯作者。

文章引用: 周鑫, 周林华. 基于深度强化学习的 3D-TSP 智能路径规划算法研究[J]. 应用数学进展, 2025, 14(5): 40-52.
DOI: 10.12677/aam.2025.145231

Abstract

With the rapid development of artificial intelligence technology, deep reinforcement learning has shown great potential in solving complex optimization problems. The three-dimensional traveling salesman problem, as an extension of the classic path planning problem, has higher complexity and practical application value. However, real-time solutions in large-scale dynamic environments still face challenges. Traditional methods suffer from high computational complexity, while heuristic algorithms are limited by solution accuracy and generalization capabilities. This paper proposes an intelligent path planning algorithm based on graph Transformer and deep reinforcement learning. The algorithm captures the positional relationships of nodes in the 3D graph structure through an encoder to generate embeddings, combines the attention mechanism and sequential decision-making characteristics of the decoder to construct a path planning strategy, and optimizes the strategy using the Reinforce learning algorithm. Experimental results show that the proposed algorithm significantly outperforms traditional methods and heuristic algorithms in terms of solving time, especially in large-scale problems. Additionally, the proposed model balances solving efficiency and accuracy through the trade-off between greedy decoding and sampling decoding strategies. This method provides an efficient solution for large-scale dynamic path optimization problems in fields such as drone path planning and intelligent manufacturing, demonstrating significant theoretical and practical value.

Keywords

Three-Dimensional Traveling Salesman Problem, Deep Reinforcement Learning, Graph Transformer, Path Planning, Attention Mechanism

Copyright © 2025 by author(s) and Hans Publishers Inc.
This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).
http://creativecommons.org/licenses/by/4.0/
Open Access

1. 引言

随着人工智能技术的快速发展，深度强化学习(Deep Reinforcement Learning, DRL)在解决复杂优化问题中展现出强大的潜力。特别是在路径规划领域，传统的优化算法如动态规划、分支限界法等在小规模或静态环境中表现良好，但在大规模动态环境中，这些方法因计算开销过高和缺乏实时性而难以适用。三维旅行商问题(3D-TSP)作为经典旅行商问题(TSP)的扩展，具有更高的复杂性和实际应用价值，广泛应用于无人机路径规划、3D 打印路径优化和物流配送等领域。

传统的 3D-TSP 求解方法如分支限界法、动态规划等在小规模问题中表现良好，但在大规模问题中计算复杂度急剧增加，难以满足实时性要求。近年来，深度强化学习在路径规划问题中通过结合图神经网络(Graph Neural Networks, GNN)和自注意力机制，能够在复杂的图结构环境中学习高效的路径规划策略，不仅可以直接从原始数据中提取特征，还能够通过试错机制在动态环境中快速更新策略，从而解决传统方法难以处理的大规模问题。

三维旅行商问题(3D-TSP)的研究最初源于对二维 TSP 求解方法的扩展。传统精确算法，如分支限界法[1] [2]和动态规划[3] [4]，因其能够保证最优解而在小规模问题中得到广泛应用。分支限界法通过系统地搜索解空间，能够在有限时间内找到全局最优路径。然而，随着问题规模的增大，这些方法的时间复

杂度呈指数增长，难以应对大规模 3D-TSP 问题。特别是在目标点数量超过 100 时，计算时间显著增加，限制了其在实际应用中的使用。尽管如此，精确算法在理论研究和小规模问题中仍具有重要价值，为后续启发式和智能化方法提供了基准参考。

为克服精确算法的计算瓶颈，研究者提出了多种启发式与元启发式算法。这些方法通过引入随机化和自然启发策略，在求解效率和解的质量之间取得了较好平衡。典型的启发式算法包括遗传算法[5] [6]、模拟退火[7] [8]和蚁群算法[9] [10]等。遗传算法通过模拟生物进化过程，能够在较短时间内找到接近最优的解；蚁群算法则通过模拟蚂蚁觅食行为，利用信息素机制引导搜索过程。这些方法在大规模 3D-TSP 问题中表现出较强的适应性，尤其是在无人机路径规划和机器人导航等实时性要求较高的场景中。然而，启发式算法也存在易陷入局部最优、参数调优复杂等问题，限制了其进一步应用。

近年来，随着深度学习和强化学习技术的快速发展，智能化方法在 3D-TSP 求解中展现出巨大的潜力。深度强化学习(DRL)通过结合深度学习的特征提取能力和强化学习的决策优化能力，能够有效处理高维状态空间和复杂约束条件。2015 年，Vinyals 等人第一个通过新的指针网络模型采用深度学习解决 TSP [11]。2018 年，Nazari 等人提出了一种对输入序列不变的新模型，大大降低了算法的计算复杂性[12]。2018 年，Kool 等人尝试使用 RL 技术来训练他们的模型[13]。他们还利用 Vaswani 等人 2017 年提出的 Transformer 架构[14]来进行特征提取，并使用多个多头注意力层，以允许更灵活和准确地表示环境状态。结果表明，深度强化学习方法提高了学习效率，算法收敛速度更快，为使用深度强化学习解决组合优化问题打开了一扇更具可扩展性的大门。

总体而言，3D-TSP 的研究已从传统的精确算法和启发式方法，逐步发展到结合深度学习和强化学习的智能化求解方法。尽管现有方法在求解效率和精度上取得了显著进展，但仍面临计算复杂度高、模型泛化能力不足等挑战。未来研究将聚焦于算法优化、计算效率提升以及在实际动态环境中的应用拓展，为无人驾驶、智能制造、航空航天等领域提供更高效的解决方案。

2. 3D-TSP 问题

三维旅行商问题(3D-TSP)是指在三维空间中寻找一条经过所有给定目标点的最短闭合路径的组合优化问题。在 3D-TSP 问题中，给定一组目标点集合，目标是规划出一条最优路径，使旅行商能够访问每个目标点恰好一次，并最终返回起点，同时最小化总路径长度。这一问题广泛应用于无人机路径规划、3D 打印路径优化和物流配送等领域，具有重要的实际意义。

具体来说，3D-TSP 问题可以表示为：给定目标集合 $P=\{p_1,p_2,\cdots,p_n\}$，每个目标点 p_i 的三维坐标为 (x_i,y_i,z_i)，目标是找到一条闭合路径，使得从起点出发，访问每个目标点一次并返回起点，且路径总长度最小。目标之间的距离通常使用三维欧几里得距离计算。

3D-TSP 问题的数学模型为：

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

$$\begin{cases} \sum_{i=1}^{n} x_{ij}=1, & j=1,\cdots,n,\quad i\neq j, \\ \sum_{j=1}^{n} x_{ij}=1, & i=1,\cdots,n,\quad j\neq i, \\ u_i-u_j+nx_{ij}\le n-1, & 1\le i\neq j\le n, \\ 1\le u_i\le n-1, & i=1,\cdots,n \\ x_{ij}\in\{0,1\}, & i,j=1,\cdots,n. \end{cases} \tag{2.1}$$

3. 基于深度强化学习的智能路径规划求解算法

针对 3D-TSP 问题，本节提出一种新的深度强化学习算法进行求解。首先，我们给出 3D-TSP 问题的强化学习形式定义，并设计相应的路径规划策略模型。然后，建立 Transformer 强化学习模型，如图 1 所示，该模型由图 Transformer 部分和强化学习训练部分组成。Transformer 模型用于对路径规划策略进行建模，包含编码器和解码器两部分。在模型中，编码器通过多头注意力机制对三维空间中的节点特征进行提取，并将节点信息传递给解码器。解码器则通过多头注意力机制及掩码操作生成路径规划策略，逐步选择节点并形成路径。

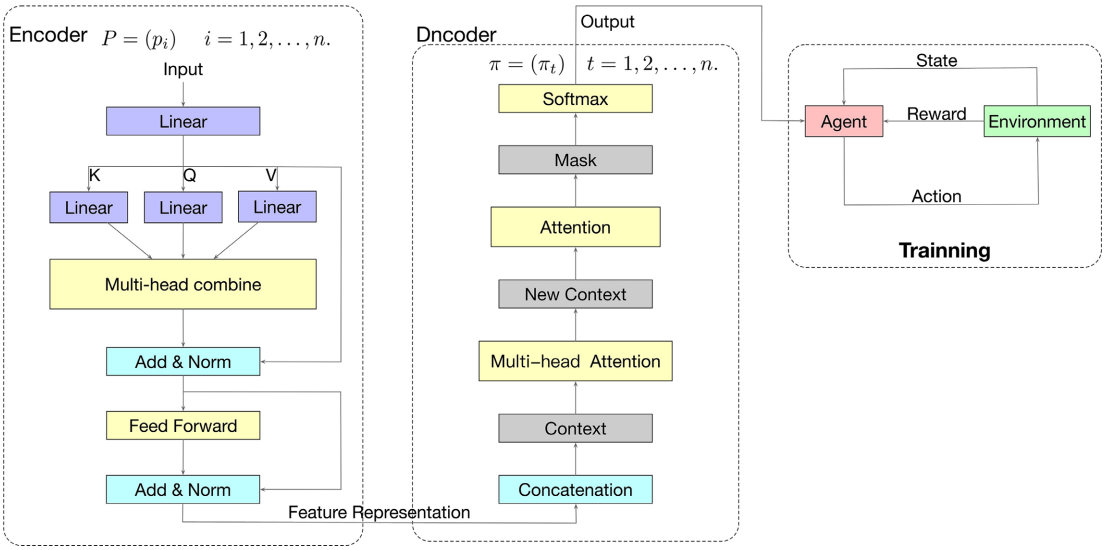

Figure 1. Graph Transformer reinforcement learning model architecture
图 1. 图 Transformer 强化学习模型架构

3.1. 3D-TSP 问题的深度强化学习形式定义

对于规模为 n 的 3D-TSP 问题(2.1)的任意一个实例 $P=\{p_1,p_2,\cdots,p_n\}$ 来说，任意两个目标点间的欧几里得距离 d_{ij} 则表示为：

$$d_{ij}=\sqrt{\left(x_i-x_j\right)^2+\left(y_i-y_j\right)^2+\left(z_i-z_j\right)^2}. \tag{3.1}$$

3D-TSP 问题的路径规划过程是一个贯序决策过程，可以根据策略依次选择经过的目标位置，我们定义一个路径规划策略 $\pi_t=k,t=1,2,\cdots,n;k=1,2,\cdots,n$，这表示第 t 步选择经过的目标位置 p_k，并且这一路径规划过程满足马尔可夫性质，则可以将路径规划过程描述为马尔可夫决策过程，以图 2 的路径规划过程为例，例子展示了规模为 6 的 3D-TSP 问题的一种分配方式。在路径规划过程中，每个步骤都有对于目标位置节点的选择概率，这些概率构成了路径规划策略 $P_\theta(\pi\mid s)$，它可以通过链式规则分解为：

$$P_\theta(\pi\mid s)=\prod_{t=1}^{n}P_\theta\left(\pi_t\mid s,\pi_1:\pi_{t-1}\right). \tag{3.2}$$

下面基于路径规划过程的马尔可夫决策过程给出 3D-TSP 问题的深度强化学习形式定义，主要定义了状态空间(State)、动作空间(Action)、状态转移函数(Transition)、奖励函数(Reward)、回报(Return)。

状态空间：存在状态空间 S，使得 $s_t\in S$，其中 s_t 定义为所有目标点集合，$s_t=\pi_1,\pi_2,\cdots,\pi_t$。

动作空间：存在动作空间 A，使得 $a_t\in A$，其中 a_t 代表在第 t 时刻智能体选择经过的目标位置 π_t。

状态转移函数：状态转移函数 $P(s_t \mid s_{t-1}, a_t)$ 表示当前时间步 t，在状态 s_{t-1} 执行动作 a_t，状态空间通过策略网络 $P_\theta(\pi_t \mid s_{t-1})$ 将状态从 s_{t-1} 转移为 s_t。这里的策略网络参数 θ 是一个可学习的权重。

奖励函数：奖励函数 $r(s_{t-1}, a_t)$ 取决于状态 s_{t-1} 和动作 a_t，作为每次执行动作后获得的即时反馈，我们定义 $r(s_{t-1}, a_t)$ 为目标节点 p_{π_t} 与上一个目标节点 p_{t-1} 之间的欧几里得距离 $\left\| p_{\pi_t} - p_{t-1} \right\|$。

回报：在分配问题的马尔可夫决策过程中，从初始时刻 $t=1$ 初始状态 s_0 开始，直到终止状态时，所有奖励的衰减之和称为回报 R，公式如下：

$$R = r_1 + \gamma r_2 + \gamma^2 r_3 + \cdots + \gamma^{n-1} r_n = \sum_{t=1}^{n} \gamma^{t-1} r_t,$$

其中，r_t 表示在时刻获得的奖励，且 $\gamma = 1$。

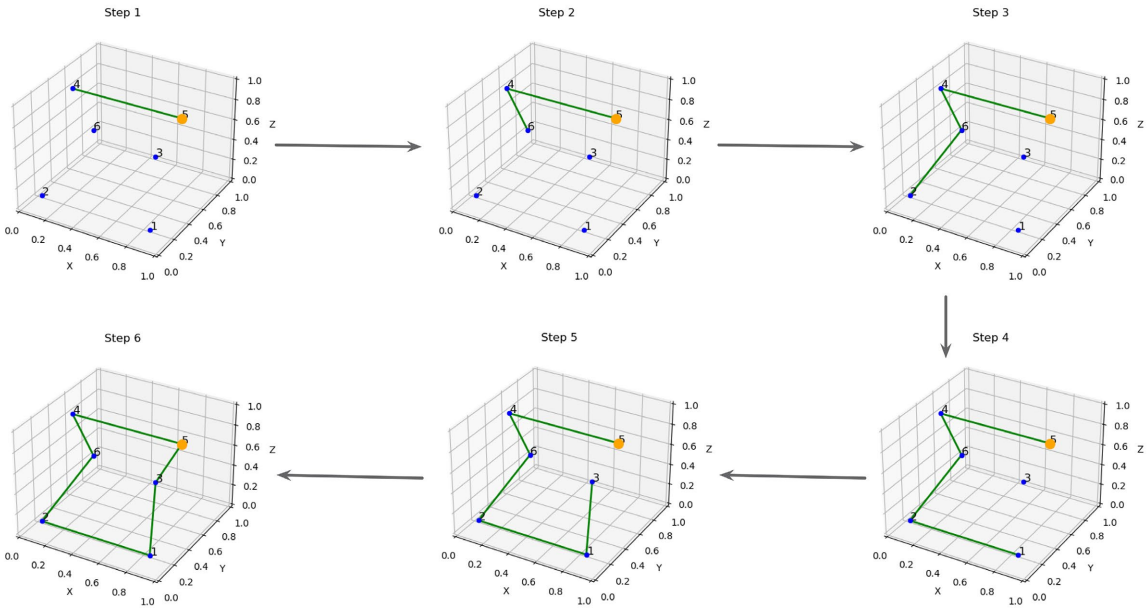

Figure 2. Sequential decision-making process for path planning in 3D-TSP problem
图 2. 3D-TSP 问题的路径规划贯序决策过程

3.2. 基于图 Transformer 的策略网络模型

1) 编码器

编码器将节点三维空间坐标数据作为输入，通过线性变换生成高维的节点嵌入，然后利用多头注意力机制将节点嵌入信息进行传递，生成更新后的包含其他节点嵌入信息的新的节点嵌入，并定义一个包含整个图信息的图嵌入。编码过程如图 3 所示。

基于 Transformer 论文[14]，构建了图 Transformer 的编码器。输入包含位置节点的 $d_p = 3$ 维的位置信息 $p_i, i = 1, 2, \cdots, n$，即位置节点的三维空间坐标数据，并生成对应的 $d_h = 128$ 维的节点嵌入。节点嵌入通过一个学习的线性映射计算：

$$h_i^{(0)} = W^p p_i + b^p,$$

其中，参数 W^p 是 $d_h \times d_p$ 的矩阵，b^p 是 $d_h = 128$ 维的向量，上标表示节点嵌入更新的次数或其注意力层数。

模型通过 N 个注意力层来更新节点嵌入，每个注意力层由两个子层组成。

第一子层是执行节点之间信息传递的多头注意力层：

$$\hat{h}_i = BN^l\left(h_i^{(l-1)} + MHA_i^l\left(h_1^{(l-1)},\cdots,h_n^{(l-1)}\right)\right), \tag{3.3}$$

这里 l 是注意力层索引，且每个注意力层不共享参数。多个注意力头可以使节点从其他节点接收更多的信息，这里可以将注意力机制解释为图中节点之间的加权信息传递算法。

对于一个注意力头，节点嵌入 h_i 从节点嵌入 h_j 接收信息的值 v_j 的权重取决于 h_i 的查询 q_i 与 h_j 的键 k_j 计算的相关性 u_{ij}。通过节点嵌入 h_i 定义维数 d_k 和 d_v，并计算每个节点的 $q_i \in R^{d_k}$、$k_i \in R^{d_k}$ 和 $v_i \in R^{d_v}$：

$$q_i = W^Q h_i, k_i = W^K h_i, v_i = W^V h_i,$$

其中，参数 W^Q 和 W^K 是 $d_k \times d_h$ 的矩阵，W^V 是 $d_v \times d_h$ 的矩阵。通过 h_i 的查询 q_i 与 h_j 的键 k_j 计算相关性 $u_{ij} \in R$：

$$u_{ij} = \begin{cases} \dfrac{q_i^{\mathrm{T}} k_j}{\sqrt{d_k}} & \text{if } i \text{ adjacent to } j, \\ -\infty & \text{ortherwise.} \end{cases}$$

在图结构中，利用掩码操作将非相邻节点的兼容性定义为 $-\infty$ 可以阻止这些节点之间的信息传递。根据相关性 u_{ij}，使用归一化指数计算注意力权重 $a_{ij} \in [0,1]$：

$$a_{ij} = \frac{\mathrm{e}^{u_{ij}}}{\sum_{j'} \mathrm{e}^{u_{ij'}}}.$$

最后，节点 i 的结果向量 h_i' 是 v_j 的凸组合：

$$h_i' = \sum_j a_{ij} v_j.$$

本章使用的参数 $d_k = d_v = \dfrac{d_h}{M} = 16$，且 $M = 8$，并计算 M 个结果向量 h_i' 的值。对于不同的注意力头，即 $m \in \{1,\cdots,M\}$，用 h_{im}' 表示各自的结果向量，然后使用 $d_h \times d_v$ 的参数矩阵 W_m^O 将这些结果向量映射成单个 $d_h = 128$ 维向量。节点嵌入 h_i 的最终多头注意力值是关于 $h_1,\cdots,h_n$ 的函数：

$$MHA_i\left(h_1,\cdots,h_n\right) = \sum_{m=1}^{M} W_m^O h_{im}'.$$

第二子层是前馈层：

$$h_i^{(l)} = BN^l\left(\hat{h}_i + FF^l\left(\hat{h}_i\right)\right). \tag{3.4}$$

前馈子层使用维度 $d_{ff} = 512$ 的隐藏子层和 ReLu 激活函数来计算节点映射：

$$FF\left(\hat{h}_i\right) = W^{ff,1} \cdot \mathrm{ReLu}\left(W^{ff,0}\hat{h}_i + b^{ff,0}\right) + b^{ff,1}.$$

此外，可以从公式(3.3)和(3.4)中看到，在每个子层都添加了一个跳跃连接和批归一化操作。使用具有可学习 $d_h = 128$ 维线性变换参数 w^{bn} 和 b^{bn} 批量归一化：

$$BN\left(h_i\right) = w^{bn} \odot \overline{BN}\left(h_i\right) + b^{bn},$$

其中，$\odot$ 表示逐元素乘积，$\overline{BN}$ 表示没有仿射变换的批量归一化。

通过 N 个注意力层更新节点嵌入之后，得到更新后的包含其他节点信息的节点嵌入 $h_i^{(N)}$，并根据节

点嵌入计算包含整个图信息的图嵌入 $h_{(g)}^{(N)}$：

$$h_{(g)}^{(N)} = \frac{1}{n}\sum_{i=1}^{n} h_i^{(N)},$$

最后将节点嵌入 $h_i^{(N)}$ 及图嵌入 $h_{(g)}^{(N)}$ 输入解码器进行解码。

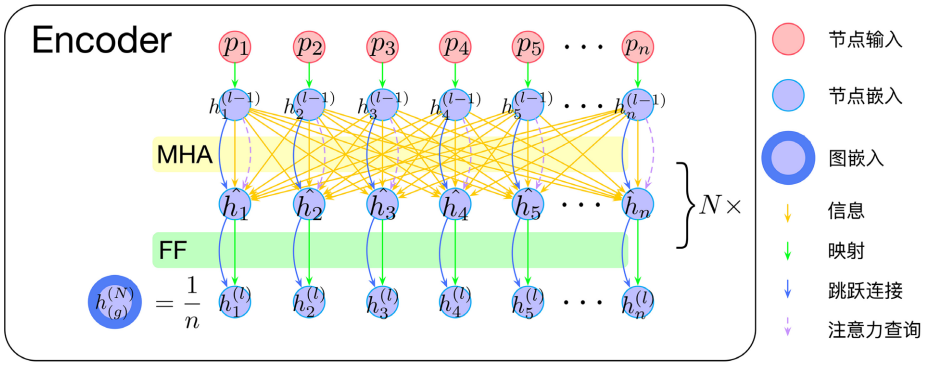

Figure 3. Multi-head attention mechanism-based encoder for 3D-TSP path planning problem
图 3. 3D-TSP 路径规划问题基于多头注意力机制的编码器

2) 解码器

解码按时间顺序进行，根据 3D-TSP 问题可知其解码时间步长，与任务分配问题的想法类似，用一个组合嵌入的上下文节点来进行解码，在当前时间步 $t \in \{1,\cdots,n\}$ 时，解码器根据编码器的图节点嵌入 $h_{(g)}^{(N)}$、节点嵌入 $h_t^{(N)}$，及首个时间步输出节点嵌入 $h_{\pi_1}^{(N)}$ 构成的上下文节点来输出接下来经过的目标位置节点 p_{π_t}。并且只使用单头注意力机制计算最终概率，以此提高解码效率，解码过程如图 4 所示。在这个过程中，解码器将图嵌入和节点嵌入作为输入，在每个时间步骤，上下文由图嵌入以及节点嵌入组成，其中，已经过的目标位置节点经过掩码技术处理不参与之后的路径规划过程。

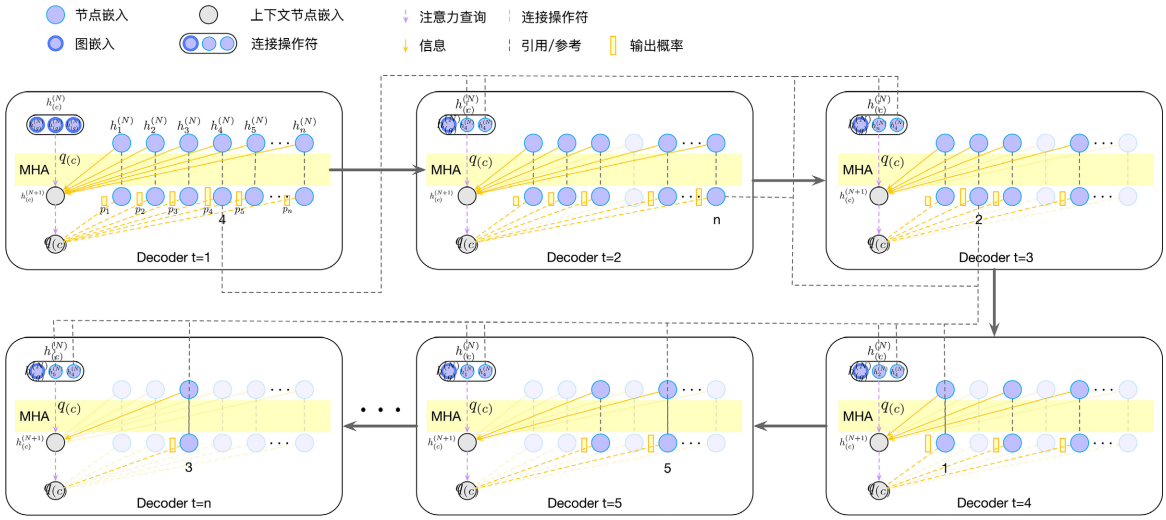

Figure 4. Attention mechanism-based decoder for 3D-TSP problem
图 4. 3D-TSP 问题基于注意力机制的解码器

对于每个解码步骤 $t \in \{1,\cdots,n\}$，解码器首先将编码器输出的图嵌入及节点嵌入作为输入，并得到上下文节点，如前文所述上下文节点由图节点嵌入 $h_{(g)}^{(N)}$、节点嵌入 $h_t^{(N)}$，及首个时间步输出节点嵌入 $h_{\pi_1}^{(N)}$ 构

成，如下所示：

$$h_{(c)}^{(N)}=\begin{cases}\left[h_{(g)}^{(N)},h_{(\pi_{t-1})}^{(N)},h_{(\pi_1)}^{(N)}\right] & t>1,\\ \left[h_{(g)}^{(N)},h_{(g)}^{(N)},h_{(g)}^{(N)}\right] & t=1,\end{cases}$$

这里$[\cdot,\cdot,\cdot]$是水平连接运算符，将$(3\cdot d_h)$维结果向量写作$h_{(c)}^{(N)}$，表示将其解释为特殊上下文节点(c)的嵌入，并使用上标(N)来与节点嵌入$h_i^{(N)}$对齐。这里没有将上下文节点嵌入投影回d_h维，而是在下面的方程中通过参数W^Q将这个转换吸收掉。

其次使用多头注意机制计算一个新的上下文节点嵌入$h_{(c)}^{(N+1)}$：

$$h_{(c)}^{(N+1)}=MHA_i\left(h_1^{(N)},\cdots,h_n^{(N)}\right),$$

这里利用单层$M=8$的多头注意力机制计算上下文节点的最终多头注意力值$h_{(c)}^{(N+1)}$，用以计算对数概率，这个机制类似于上文提到的编码器，但不使用跳跃连接、批量归一化或前馈子层以实现最大效率。

对于一个注意力头，键k_i和值v_i来自节点嵌入$h_i^{(N)}$，但仅从上下文节点$h_{(c)}^{(N)}$计算单个查询$q(c)$，为了清晰起见，省略了上标(N)。

$$q(c)=W^Q h_{(c)},k_i=W^K h_i,v_i=W^V h_i,$$

然后计算上下文节点查询$q(c)$与所有节点的相关性，并在时间t无法访问的节点设置掩码$u_{(c)j}=-\infty$，对于3D-TSP问题，这意味着已经经过目标位置节点：

$$u_{(c)j}=\begin{cases}\dfrac{q_{(c)}^{\mathrm{T}}k_j}{\sqrt{d_k}} & \text{if } j\neq\pi_{t'}\ \forall t'<t,\\ -\infty & \text{ortherwise,}\end{cases}$$

这里$d_k=d_h$是查询$q(c)$和键k_j的维度。

根据相关性$u_{(c)j}$，使用归一化指数计算注意力权重$a_{(c)j}\in[0,1]$：

$$a_{(c)j}=\frac{\mathrm{e}^{u_{(c)j}}}{\sum_{j'}\mathrm{e}^{u_{(c)j'}}}.$$

上下文节点嵌入$h_{(c)}$的结果向量$h'_{(c)}$是v_j的凸组合：

$$h'_{(c)}=\sum_j a_{(c)j}v_j.$$

对于不同的注意力头，即$m=1,2,\cdots,M$，用$h'_{(c)m}$表示各自的结果向量，然后同样使用$d_h\times d_v$的参数矩阵W_m^O将这些结果向量映射成单个$d_h=128$维向量。上下文节点嵌入$h_{(c)}$的最终多头注意力值是关于$h_1,\cdots,h_n$的函数：

$$MHA_i\left(h_1,\cdots,h_n\right)=\sum_{m=1}^{M}W_m^O h'_{(c)m}.$$

最后为了输出概率$P_\theta(\pi_t\mid s,\pi_1:\pi_{t-1})$，添加了一个最终的解码器层，其中只有一个注意力头$(M=1,d_k=d_h)$。

$$q_{(c)}=W^Q h_{(c)}^{N+1},k_i=W^K h_i^N,v_i=W^V h_i^N.$$

在这里计算兼容性时，在应用掩码之前加入激活函数tanh，将结果剪切在$[-C,C],C=10$范围内：

$$u_{(c)j}=\begin{cases}C\cdot\tanh\left(\dfrac{q_{(c)}^{\mathrm{T}}k_j}{\sqrt{d_k}}\right) & \text{if } j\neq\pi_{t'},\ \forall t'<t,\\ -\infty & \text{ortherwise.}\end{cases}$$

然后利用兼容性计算非归一化的对数概率，并使用 softmax 计算最终的输出概率向量 p_i：

$$p_i=p_\theta\left(\pi_t=i\mid s,\pi_{1:t-1}\right)=\frac{\mathrm{e}^{u_{(c)i}}}{\sum_j \mathrm{e}^{u_{(c)j}}}.$$

从而得到在当前时间步 $t\in\{1,2,\cdots,n\}$ 时，接下来要经过的目标位置节点 x_{π_t}，则进一步可以得到路径规划的策略：

$$P_\theta\left(\pi\mid s\right)=\prod_{t=1}^{n}P_\theta\left(\pi_t\mid s,\pi_1:\pi_{t-1}\right).$$

3.3. 动作选择及局部搜索策略

经过多轮次学习的策略网络 $P_\theta\left(\pi\mid s\right)$ 拥有较好的决策能力，动作选择策略根据网络输出的概率向量选择动作。本章采用两种不同的动作选择策略。

1) 贪婪动作选择策略完全信任策略网络，每一步都以解码器输出概率值为基准，这种方式可以根据路径规划策略的概率 $P_\theta\left(\pi_t\mid s,\pi_1:\pi_{t-1}\right)$，选择拥有最大概率值的动作 p_{π_t}，从而得到一个确定的路径规划方式。

2) 采样动作选择策略以解码器输出概率 $P_\theta\left(\pi_t\mid s,\pi_1:\pi_{t-1}\right)$ 作为采样概率分布，在该分布上进行动作采样选择，因此该策略并非每次都会选择最大概率值的动作，而是以不同的概率选择对应动作，每次采样得到的路径规划方式是不确定的，但可以进一步搜索解空间。

3.4. 路径规划策略网络训练方法

为了训练模型，定义了损失函数：

$$\mathcal{L}\left(\theta\mid s\right)=E_{p_{\theta(\pi|s)}}\left[L\left(\pi\right)\right],$$

即成本 $L\left(\pi\right)$ 的期望，其中 $L\left(\pi\right)$ 定义为：

$$L\left(\pi\right)=\sum_{t=1}^{n}\left\|p_{\pi_t}-p_{\pi_{t+1}}\right\|_2,$$

并且 $p_{\pi_{n+1}}=p_1$。

最后使用基线为 $b\left(s\right)$ 的 REINFORCE 梯度估计器，通过梯度下降来优化损失函数：

$$\nabla\mathcal{L}\left(\theta\mid s\right)=E_{p_\theta(\pi|s)}\left[L\left(\pi\right)-b\left(s\right)\nabla\log p_\theta\left(\pi\mid s\right)\right].$$

这里的基线 $b\left(s\right)$ 会定期更新。$b\left(s\right)$ 是通过当前最佳模型定义的策略进行确定性贪婪推演得到的解决方案的成本。模型通过随机策略学习演员网络的参数 θ。计算和更新上述公式的梯度，通过迭代训练获得最优策略 $P_\theta\left(a_t\mid s,a_{1:t-1}\right)$。算法伪代码见表 1。

Table 1. Pseudocode of the GTRLT model's Reinforce with Rollout Baseline training algorithm
表 1. GTRLT 模型的 Reinforce with Rollout Baseline 训练算法伪代码

Algorithm 1	Reinforce with Rollout Baseline
1:	Input: number of epochs E, steps per epoch T, batch size B, significance α
2:	Init $\theta,\theta^{bl}\leftarrow\theta$

续表

3:	for epoch = $1,\cdots,E$ do
4:	for step = $1,\cdots,E$ do
5:	$s_i \leftarrow \text{RandomInstance}()\quad \forall i \in \{1,\cdots,B\}$
6:	$\pi_i \leftarrow \text{SampleRollout}(s_i, p_\theta)\quad \forall i \in \{1,\cdots,B\}$
7:	$\pi_i^{BL} \leftarrow \text{GreedyRollout}(s_i, p_\theta^{BL})\quad \forall i \in \{1,\cdots,B\}$
8:	$\nabla\mathcal{L} = \sum_{i=1}^{B}\left(L(\pi_i) - L(\pi_i^{BL})\right)\nabla_\theta \log p_\theta(\pi_i)$
9:	$\theta \leftarrow \text{Adam}(\theta, \nabla\mathcal{L})$
10:	end for
11:	If $\text{OnesidePairedTTest}(p_\theta, p_\theta^{BL}) \le \alpha$ then
12:	$\theta^{bl} \leftarrow \theta$
13:	end if
14:	end for

4. 实验结果与分析

本研究包含三组不同规模的 3D-TSP 问题实验，并针对每组任务训练了相应的三个求解模型，分别命名为 GTRLT20、GTRLT50 和 GTRLT100。模型经过了 50 个训练周期，在平均情况下，每个周期的训练时间分别为 24 分钟、66 分钟和 190 分钟，对应于 GTRLT20、GTRLT50 和 GTRLT100。最后，使用训练完成的最优模型在规模为 200、300 和 400 的测试集上进行了求解，以观察模型对更大规模问题的求解效果。在每组实验中，所有数据的节点坐标均是在单位正方形[0, 1] × [0, 1]中随机生成的，训练集包括 1,280,000 条数据，验证集包括 10,000 条数据，测试集包含 100 条数据。实验超参数设置见表 2。

Table 2. Experimental hyperparameter settings
表 2. 实验超参数设置

超参数	值
周期数	50
批量大小	512
注意力层	3
训练集规模	1,280,000
优化器	Adam
学习率	1e−3
验证集规模	10,000
测试集规模	100
采样解码规模	1280

4.1. 求解时间对比分析

为了观察训练的 GTRLT 算法求解的时效性，使用 Greedy 解码策略与其他几类算法进行了对比，主要包括 Google OR Tool、遗传算法、蚁群算法等，求解时间是求解 100 条测试集数据时间的平均值，求解时间具体结果见表 3。

Table 3. Solution time of GTRLT model and other methods for solving 3D-TSP problems
表 3. GTRLT 模型及其他方法求解 3D-TSP 问题的时间

规模 / 算法	$n = 20$	$n = 50$	$n = 100$
OR Tool	0.0278 s	0.1134 s	0.6698 s
遗传算法	32.9213 s	48.1911 s	67.7212 s
蚁群算法	29.4899 s	75.2491 s	218.6175 s
GTRLT(Greedy)算法	0.0138 s	0.0284 s	0.0489 s

根据求解时间结果可以知道，GTRLT (Greedy)算法的求解时间最少。与 GTRLT (Greedy)算法对比，Google OR Tool 求解时间相对表现较好。GTRLT 算法在求解时间上对比 Google OR Tool 算法有大幅减少，随着问题规模的增加，GTRLT 算法求解时间增加的幅度远小于 Google OR Tool 算法。对于更大大规模 3D-TSP 问题，GTRLT 算法求解时间更少，更能满足求解的实时性要求。

4.2. 模型泛化性分析

针对模型泛化性，分别利用训练的模型 GTRLT20、GTRLT50、GTRLT100，求解规模为 20、50、100 的 3D-TSP 问题，以观察其在不同规模实例上的泛化性，其中 GTRLT20、GTRLT50、GTRLT100 均训练 50 轮，对比结果见表 4。

Table 4. Generalization comparison of GTRLT model
表 4. GTRLT 模型泛化性对比

	$n = 20$		$n = 50$		$n = 100$	
模型	目标值	求解时间	目标值	求解时间	目标值	求解时间
GTRLT20	6.5157	0.0138 s	11.9139	0.0284 s	19.3344	0.0489 s
GTRLT50	6.6837	0.0138 s	11.6216	0.0284 s	18.2761	0.0489 s
GTRLT100	6.9234	0.0138 s	11.8643	0.0284 s	18.2297	0.0489 s

由模型泛化性可知，对于不同规模的模型，对相同规模问题的求解时间相同。当训练时期相同时，训练的模型与求解问题相同，求解精度会更高，但是模型的训练集越大，求解能力也会越强。

从上述分析可知，GTRLT 模型的泛化性较好，因此在求解 3D-TSP 问题时，可以选取适当规模的训练集，对模型进行长时间训练，得到一个通用模型，对任意规模的 3D-TSP 问题进行求解。

4.3. 路径规划结果及采样策略

由于追求求解的时效性，其精确性会有所损失，求解目标值具体结果见表 5，其中目标值是 100 条测试集数据求得路径规划结果的平均值。

从表 5 中可以看出，对于相同规模的 3D-TSP 问题，本章方法在求解精度上不如 Google OR Tool 等精确方法，相较于启发式算法中的蚁群算法，也具有一定差距，但当问题规模增大，其效果远好于遗传算法。为了满足求解过程的时间效率需求，可以选择使用贪婪解码策略来输出确定解，尽管这样做可能会牺牲一定的解的精确性。

为了提高解的准确性，可以采用采样解码策略来解决 3D-TSP 问题。然而，采样解码策略是通过牺牲时间效率来获得精确解的。因此，主要比较了贪婪解码策略的结果。理论上，可以通过不断增加采样数

量来提高解的准确性。这个选择可以在实际应用中根据不同情况进行权衡。本研究采用了采样策略来采样 1280 个解，并输出了最优解。求解结果如表 6 所示。

可以看到，采样策略对解的准确性会有一定效果，但与此同时，也牺牲了求解的时效性，因此可以对求解时效性和求解精度进行权衡，选择合适的采样策略进行求解。

Table 5. Results of GTRLT algorithm and other methods for solving 3D-TSP problems
表 5. GTRLT 算法及其他方法求解 3D-TSP 问题结果

规模 / 算法	$n = 20$	$n = 50$	$n = 100$
OR Tool	6.4719	11.3395	17.4333
遗传算法	6.6525	13.6615	28.4909
蚁群算法	6.4109	11.3867	17.8385
GTRLT (Greedy)算法	6.5157	11.6216	18.2297

Table 6. Comparison of GTRLT model with greedy decoding strategy, sampling decoding strategy, and OR Tool solutions
表 6. GTRLT 模型贪婪解码策略、采样解码策略及 OR Tool 求解结果对比

规模 / 算法	$n = 20$	$n = 50$	$n = 100$
OR Tool	6.4719	11.3395	17.4333
GTRLT (Greedy)算法	6.5157	11.6216	18.2297
GTRLT (Sample)算法	6.4188	11.2185	17.5522

5. 结论与展望

本文针对三维旅行商问题(3D-TSP)，提出了一种基于图 Transformer 与深度强化学习的智能路径规划算法。通过编码器捕捉三维空间节点的位置关系并生成嵌入表示，结合解码器的注意力机制与序贯决策特性构建路径规划策略，并利用强化学习 Reinforce 算法优化策略，实现了对 3D-TSP 问题的高效求解。实验结果表明，GTRLT 算法在求解时间上显著优于传统精确算法与启发式算法。此外，模型通过贪婪解码与采样解码策略的灵活权衡，兼顾了求解效率与精度需求，为动态环境下的实时路径规划提供了实用化解决方案。

本文提出的算法在 3D-TSP 求解中取得了显著进展，但仍存在可探索的方向：

1) 算法优化与扩展：当前模型主要针对静态环境设计，未来可研究动态环境(如实时变化的障碍物或移动目标点)下的自适应路径规划策略。此外，引入图神经网络(GNN)的动态更新机制或结合元学习技术，有望进一步提升模型对复杂场景的适应能力。

2) 多目标优化与约束增强：实际应用中常需考虑多目标(如能耗、风险)与多约束(如时间窗、负载限制)的路径规划问题。未来可探索多目标强化学习框架，结合约束满足机制，增强算法的工程实用性。

3) 泛化性与鲁棒性验证：当前实验基于均匀分布的随机节点生成，未来需在非均匀分布、高噪声或部分观测场景下验证模型的鲁棒性，并通过迁移学习策略提升跨场景泛化能力。

总之，基于深度强化学习的智能路径规划方法在 3D-TSP 问题中展现出巨大的潜力，未来研究可进一步推动其在工业、物流、无人系统等领域的落地应用，为复杂优化问题提供更高效、更智能的解决方案。

基金项目

吉林省自然科学基金自由探索重点项目(YDZJ202201ZYTS585)。

参考文献

[1] Lawler, E.L. and Wood, D.E. (1966) Branch-and-Bound Methods: A Survey. *Operations Research*, **14**, 699-719. https://doi.org/10.1287/opre.14.4.699

[2] Papadimitriou, C.H. and Steiglitz, K. (1998) Combinatorial Optimization: Algorithms and Complexity. Courier Corporation.

[3] Bertsekas, D.P. (1995) Dynamic Programming and Optimal Control. Athena Scientific.

[4] Sniedovich, M. (2010) Dynamic Programming: Foundations and Principles. CRC Press.

[5] Rezoug, A., Bader-El-Den, M. and Boughaci, D. (2018) Guided Genetic Algorithm for the Multidimensional Knapsack Problem. *Memetic Computing*, **10**, 29-42. https://doi.org/10.1007/s12293-017-0232-7

[6] Lin, B., Sun, X. and Salous, S. (2016) Solving Travelling Salesman Problem with an Improved Hybrid Genetic Algorithm. *Journal of Computer and Communications*, **4**, 98-106. https://doi.org/10.4236/jcc.2016.415009

[7] Teoh, E.J., Tang, H. and Tan, K. (2006) A Columnar Competitive Model with Simulated Annealing for Solving Combinatorial Optimization Problems. *The* 2006 *IEEE International Joint Conference on Neural Network Proceedings*, Vancouver, 16-21 July 2006, 3254-3259. https://doi.org/10.1109/ijcnn.2006.247320

[8] van Laarhoven, P.J.M., Aarts, E.H.L. and Lenstra, J.K. (1992) Job Shop Scheduling by Simulated Annealing. *Operations Research*, **40**, 113-125. https://doi.org/10.1287/opre.40.1.113

[9] Deng, W., Xu, J. and Zhao, H. (2019) An Improved Ant Colony Optimization Algorithm Based on Hybrid Strategies for Scheduling Problem. *IEEE Access*, **7**, 20281-20292. https://doi.org/10.1109/access.2019.2897580

[10] Ramadhani, T., Hertono, G.F. and Handari, B.D. (2017) An Ant Colony Optimization Algorithm for Solving the Fixed Destination Multi-Depot Multiple Traveling Salesman Problem with Non-Random Parameters. *AIP Conference Proceedings*, **1862**, Article ID: 030123. https://doi.org/10.1063/1.4991227

[11] Vinyals, O., Fortunato, M. and Jaitly, N. (2015) Pointer Networks. *Advances in Neural Information Processing Systems*, **28**, 1-9.

[12] Nazari, M., Oroojlooy, A., Snyder, L., *et al*. (2018) Reinforcement Learning for Solving the Vehicle Routing Problem. *Advances in Neural Information Processing Systems*, **31**, 1-11.

[13] Kool, W., Van Hoof, H. and Welling, M. (2018) Attention, Learn to Solve Routing Problems! arXiv: 1803.08475.

[14] Vaswani, A., Shazeer, N., Parmar, N., *et al*. (2017) Attention Is All You Need. *Advances in Neural Information Processing Systems*, **30**, 1-11.