计算机科学与探索
Journal of Frontiers of Computer Science and Technology
ISSN 1673-9418,CN 11-5602/TP

《计算机科学与探索》网络首发论文

题目： 求解旅行商问题的 GCN-Pointransformer 模型
作者： 邱云飞，刘一菲，于智龙，金海波
网络首发日期： 2024-06-25
引用格式： 邱云飞，刘一菲，于智龙，金海波．求解旅行商问题的 GCN-Pointransformer 模型[J/OL]．计算机科学与探索．
https://link.cnki.net/urlid/11.5602.tp.20240624.1713.002

计算机科学与探索
Journal of Frontiers of Computer Science and Technology doi:10.3778/j.issn.1673-9418.2404030

求解旅行商问题的 GCN-Pointransformer 模型

邱云飞[1]，刘一菲[1+]，于智龙[2]，金海波[1]

1. 辽宁工程技术大学 软件学院，辽宁 葫芦岛 125105
2. 辽宁工程技术大学 工商管理学院，辽宁 葫芦岛 125105
+ 通信作者 E-mail: lyf131499@126.com

摘要：由于 Transformer 模型基于全连接注意力机制，导致在求解经典旅行商问题（Traveling Salesman Problem，TSP）时，计算复杂度较高并且 GPU 内存使用量过大。针对此问题，提出了一种基于图卷积嵌入层和多头局部自注意力机制的 GCN-Pointransformer 模型。首先，使用图卷积嵌入方式从输入数据中进行空间特征学习，图卷积嵌入层包含多个可以提取输入数据局部特征的卷积核；其次，使用多头局部自注意力机制 MHLSA（Multi Head Local Self Attention，MHLSA），删除冗余信息并提取有用的特征；此外，在编码器中使用可逆残差网络，在反向传播过程中只存储输入和输出嵌入特征对；最后，模型在解码器中增加了 Pointer 指针层，使用注意力权重作为概率分布，确定要访问的下一个节点。在 TSP 随机数据集上进行对比实验，优化间隙减少 12%、GPU 内存减少约 11%、推理时间减少约 25%，结果表明，该方法优于求解 TSP 的标准 Transformer 模型。

关键词：TSP；GCN-Pointransformer；MHLSA；可逆残差；指针层
文献标志码：A　　中图分类号：TP399

GCN-Pointransformer model for solving the Traveling Salesman Problem

QIU Yunfei[1], LIU Yifei[1+], YU Zhilong[2], JIN Haibo[1]

1. School of Software, Liaoning Technology University, Huludao, Liaoning 125105, China
2. School of Business Administration, Liaoning Technology University, Huludao, Liaoning 125105, China

Abstract: Because the Transformer model is based on the fully connected attention mechanism, the computational complexity is high and the GPU memory usage is too large when solving the classic Traveling Salesman Problem. To solve this problem, a GCN-Pointransformer model based on graph convolutional embedding layer and multi-head local self-attention mechanism was proposed. Firstly, the graph convolutional embedding method is used to learn spatial features from the input data, and the graph convolutional embedding layer contains multiple convolution kernels that can extract the local features of the input data, and secondly, the Multi Head Local Self is used Attention,

基金项目：国家自然科学基金（62173171）。
This work was supported by the National Natural Science Foundation of China (62173171).

which removes redundant information and extracts useful features, in addition, uses a reversible residual network in the encoder to store only input and output embedding feature pairs during backpropagation, and finally, the model adds a Pointer layer in the decoder, using attention weights as probability distributions to determine the next node to be visited. Comparative experiments on the TSP random datasets show that the optimization gap is reduced by 12%, the GPU memory is reduced by about 11%, and the inference time is reduced by about 25%, and the results show that the proposed method is superior to the standard Transformer model for solving TSP.

Key words: TSP;GCN-Pointransformer;MHLSA;Reversible residual;Pointer layer

旅行商问题(Travelling Salesman Problem，TSP ）是一种经典的组合优化问题，它的历史可以追溯到19 世纪。然而，这个问题的名称和现代形式是在 20 世纪 50 年代由美国的数学家和运筹学家们正式提出和定义的。它的目标是找到一个最短的旅行路线，使得一个旅行商可以从一个城市出发，访问所有其他城市一次且仅一次，然后返回到出发城市。TSP 属于仓库、运输、供应链、硬件设计、制造等行业使用的路由类问题。

1954 年，Dantzig、Fulkerson 和 Johnson 首次使用线性规划方法解决了一个 49 座城市的 TSP。后来，研究者们开发了精确算法（如分支定界[1,2]、切割平面方法[3]等）和各种启发式算法（如模拟退火[4]、遗传算法、局部搜索[5]等）来求解 TSP。1962 年，Michael Held 等人提出一种时间复杂度为 $O(n^2 2^n)$ 的动态规划算法[5]。随后，Z Gu、E Rothberg 等人引入了具有切割平面(Cutting Planes，CP)和称为 Gurobi 的分支定界(Branch-and-Bound，BB)通用整数规划(Integer programming，IP)求解器[6]。2007 年，David L Applegate 和 Robert E Bixby 等人设计了具有线性整数规划、切割平面和分支定界的 Concorde[7]。TSP 的一个著名的启发式方法是 Christofides 算法[8]。Google-OR 工具[9]是另一种著名的启发式方法，它通过执行局部搜索和元启发式来寻找 TSP 的近似解。然而，启发式方法通常以牺牲最优性来换取计算效率，并且这些方法通常以规则的形式进行表达，启发式算法的性能往往依赖于所选用的启发式规则，如果规则选择不当，算法可能无法找到一个好的解，或者需要很长时间才能找到一个解。许多启发式算法，特别是那些基于梯度或者贪心策略的算法，可能会陷入局部最优解，而无法达到全局最优。

后来逐渐出现了一些有前景的技术，在研究中，最具开创性的工作是指针网络[10]，利用神经注意力机制创建指针，用于选择输入序列中的元素作为输出。这使得模型能够处理输出字典大小可变的问题，使用基于 RNN 的编码器和解码器。Nazari 等人直接使用嵌入输入而不是使用指针网络的 RNN 编码器的隐藏状态，以降低计算复杂性而不影响性能[11]。随着技术的进一步发展，研究者利用图神经网络求解 TSP，图神经网络在求解 TSP 问题时能够充分利用图结构数据，学习全局信息，实现端到端学习，具有较强的泛化能力和可解释性。Joshi 等人提出了一种通过图卷积神经网络模型和基于监督学习的方法[12]预测整个图边缘概率矩阵的方法。Stohy 等人提出了一种用于 TSP 的混合指针网络模型，该模型在大规模 TSP 实例[13]中表现出良好的性能。然而，与基线图指针网络[14]相比，它的推理时间较长，导致模型结构更复杂。研究人员引入卷积神经网络(Convolutional neural networks，CNN)来处理 TSP[15,16]，但效果不佳，将 TSP 转换为图像表示可能会导致信息丢失，特别是当图像分辨率较低的时候。同时，在大规模实例中，计算时间仍然过长。Sultana 等人提出了一个将 1D-CNN 与 LSTM 相结合的新模型，但仍然是一个基于 RNN 的模型[17]。虽然，该方法在一定程度上能够学习更复杂的组合优化问题，但在实际应用中可能需要进一步改进以应对更大规模的问题和实际问题。Kool 等人提出了

一个基于 Transformer 的模型，该模型由纯注意力模块组成，并使用 REINFORCE 训练模型来解决 TSP 和车辆路径问题[18]等各种路由问题。Wu 等人提出了一种基于 Transformer 的深度强化学习框架，该框架训练一个改进启发式，迭代地改进初始解[19]。最近，Bresson 等人提出了一个 TSP Transformer 模型[20]，该模型具有多头注意力和残差连接的标准 Transformer 编码器，它使用批量归一化和自回归解码，并在解码器部分引入了一个自注意力模块，在部分旅行上使用自注意力模块准备查询。尽管 TSP Transformer 模型显示了许多 TSP 实例的 SOTA（state-of-the-art，SOTA），但它具有基于全连接注意力的复杂模型结构[21]，并且还涉及较大的 GPU 使用。而且，标准 Transformer 模型中的线性嵌入没有考虑局部空间信息，在学习局部组合性方面存在局限性，因此，它的计算复杂度和内存消耗巨大。

针对 TSP Transformer 模型中出现的问题，本文提出了一种新颖的 GCN-Pointransformer 模型，将图卷积嵌入层添加到标准 Transformer 模型中，将输入的特征向量映射到一个更高维度的空间，以便在高维空间中捕获更丰富的特征信息提取输入数据的局部空间特征。本文还通过提出多头局部自注意力来改进注意力机制，该部分的自注意力只关注解码器中最近访问的节点，显著减少了在使用 Transformer 模型求解 TSP 时全连接注意力的冗余，通过除去过多的注意力连接来提高 TSP 解决方案的质量。

1 相关介绍

1.1 TSP 介绍

定义对旅行商问题的数学模型描述如下：假定 N 个城市的集合 $C=\{1,2,\cdots,i,j,\cdots,n\}$（$n$ 是一个有限的正自然数）；其中两城市间的距离为 $D_{ij}\in Z+$，其中的 i，$j\in C\,(1\leqslant i$，$j\leqslant n)$。

由以上分析可得出 TSP 问题的一个数学模型为：

$$Min\sum_{i\neq j}D_{ij}x_{ij} \tag{1}$$

$$s.t.\sum_{i\neq j}^{n}x_{ij}=1,i=1,2,\cdots,n \tag{2}$$

$$\sum_{j=1}^{n}x_{ij}=1,j=1,2,\cdots,n \tag{3}$$

$$\sum_{i,j\in C}x_{ij}\leqslant|C|-1,2\leqslant|C|\leqslant n-2,C\subset\{1,2,\cdots,n\} \tag{4}$$

$$x_{ij}\in\{0,1\},i=j=1,2,\cdots,n;i\neq j \tag{5}$$

在式（1）中，设置 x_{ij} 作为一个决定因素，如果没有从城市 i 直接到达城市 j 的路径，那么就设置 $x_{ij}=0$；若商人从城市 i 直接到达 j，则 $x_{ij}=1$；在以上所建立的 5 个方程式中，方程式（1）保证了路径的和最小，而方程式（2）和（3）则保证了从一个城市进出一次。而式（4）则保障了走过的所有路径都没有回路。其中 $|C|$ 表示集合 $|C|$ 中元素个数[22]。

为使 TSP 的时间复杂性更加直观，这里使用一种回溯方法，对其进行简要的解释，如图 1 所示。在 3 个城市中，旅行者可以选择 6 条旅行路线。TSP 问题的计算复杂程度随城市数量的增加而增加。当一个城市的数目是 n 的时候，搜索最少路径的时间复杂性是 O(n!)。如果 n 越大，则需要更多的计算。

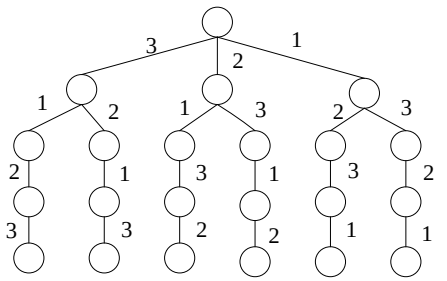


图 1　回溯法求 TSP，n=4

Fig.1　TSP calculation by backtracking method，n = 4

1.2 图卷积神经网络介绍

图卷积神经网络（Graph Convolutional Network，GCN）是一种用于处理图结构数据的深度学习模型，可以有效地学习节点之间的关系和图的全局特征。

GCN 通过聚合每个节点的邻居节点特征来更新节点的表示。对于每个节点，GCN 会将其邻居节点的特征进行加权平均，得到一个新的表示，从而考虑了节点与其邻居节点之间的关系。GCN 中的邻

居聚合操作是通过共享权重矩阵来实现的，这意味着对于图中的每个节点，都使用相同的权重矩阵来更新其表示，从而在整个图上实现参数共享。在GCN中通常会使用非线性激活函数（如ReLU）来引入非线性，增加模型的表达能力。

1.3 注意力机制介绍

注意力机制用于在深度学习模型中学习不同部分之间的关联性，从而能够更加灵活地对输入数据进行加权处理。注意力机制可以类比为人类在处理信息时的注意力分配方式。一个人在阅读一篇文章时，当他试图理解文章中的某个概念或者句子时，会将注意力集中在那个部分，而忽略其他不相关的内容。这种集中注意力的过程就可以类比为注意力机制。

在深度学习中，注意力机制的作用类似于人类的注意力。当模型需要处理一个序列（比如一段文本或一段语音）时，它可能需要更多地关注序列中的某些部分，而忽略其他部分。注意力机制允许模型根据输入的不同部分的重要性来调整自己的处理方式。注意力机制中通常用Query表示查询、Key表示键、Value表示值，用它们之间的关联性来确定不同部分的权重。通过计算Query和Key之间的相似度，然后将相似度进行归一化处理得到注意力权重，最后将注意力权重与对应的Value相乘并加权求和，得到最终的输出结果。

2 GCN-Pointransformer模型

本文提出的GCN-Pointransformer模型具有编码器和解码器两大块，该模型将图卷积嵌入层与标准Transformer模型结合，使用可逆残差网络和局部多头自注意力机制，并在解码器中添加一个指针层。图卷积嵌入层用来提取局部空间信息，在GCN-Pointransformer模型编码器中使用可逆残差网络减少内存消耗。解码器中基于全连接的自注意力机制改进为局部自注意力机制，消除了不必要的注意力连接。解码器中添加了一个指针层，根据给定的查询顺序生成下一个节点，能够节约时间。

模型的整体结构如下图2所示。该模型的输入是平面点集 $M=\{m_1,\cdots,m_n\}$ 有 n 个节点（城市），其中 $m_i\in R^2$ 表示点的2D笛卡尔坐标。模型的输出表示为 $R=\{r_1,\cdots r_n\}$，是一个序列，表示最优的预测行程，其中 r_t 是解码器第 t 个解码步骤中选择的节点索引。

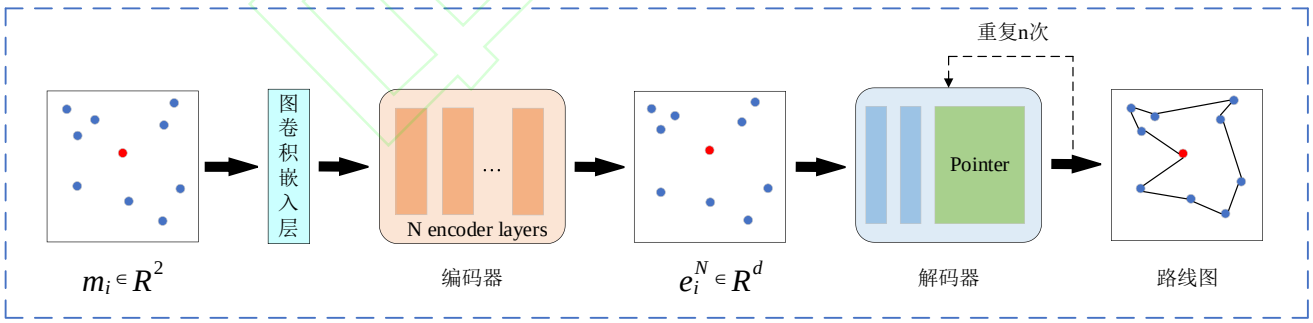


图2 模型整体结构

Fig.2 The overall structure of the model

图2中 $\boldsymbol{e_i^N}$ 是第 N 个编码器层的第 i 个节点的编码器输出，Pointer是模型在解码器中添加的指针层。每个节点首先进入到图卷积嵌入层，将它们映射到一个更高维度的空间，以便在高维空间中捕获更丰富的特征信息，用来提取局部空间特征。图卷积嵌入层的输出结果再输入到编码器中（N层），再将第 N 个编码器层的第 i 个节点的编码器输出输入到解码器中，Pointer层的作用是使用注意力权重作为概率分布，确定要访问的下一个节点。这样的过程重复 n 次，最终输出行程路线图。设 $D(m_i,m_j)$ 为节点 m_i 和 m_j 之间的距离，目标是最小化总行程长度，同时恰好访问每个节点一次，然后返回到起始

节点，见公式（6）。

$$\sum_{t=1}^{n-1} D(m_{r_t}, m_{r_{t+1}}) + D(m_{r_n}, m_{r_1}) \tag{6}$$

2.1 GCN-Pointransformer 模型编码器

GCN-Pointransformer 模型编码器由图卷积嵌入层和 N 个相同的编码器层组成，如图 3 所示。$\boldsymbol{m}_f, \boldsymbol{m}_1, \boldsymbol{m}_2, \cdots \boldsymbol{m}_n$ 分别是输入的原始特征向量，$\boldsymbol{m}_f$ 是虚拟节点特征向量，学习与其他节点特征的依赖关系，在解码时可以在尽可能好的位置开始。(y_1, y_2) 是输出的嵌入特征。$\boldsymbol{e}_f^N, \boldsymbol{e}_1^N, \boldsymbol{e}_2^N, \cdots \boldsymbol{e}_n^N$ 是编码器层的最终输出。

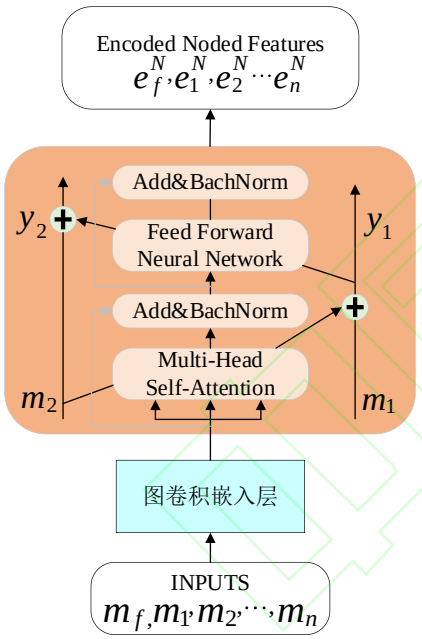


图 3　GCN-Pointransformer 模型编码器整体结构

Fig.3　Overall structure of the GCN-Pointransformer encoder

图卷积嵌入层通过从输入的数据点提取局部空间信息来生成嵌入向量，该局部空间信息被传递到编码器层，编码器由 N 个相同层组成，每一层都有两个子层。第一层是多头自注意力子层（Multi Head Self Attention，MHSA），第二层是简单的、位置全连接的前馈神经网络（Feed Forward Neural Network，FFNN）。本文在两个子层中的每一个周围使用可逆残差连接，可逆残差连接通过设计一种结构，在反向传播阶段可以根据后续层的梯度重新计算前向传播阶段的激活值，避免在反向传播阶段保存激活值，可以减少计算开销并降低计算复杂度提高了模型的效率从而节省了内存消耗。

在编码器前面添加了一个图卷积嵌入层，用于从输入节点提取空间信息，在图卷积过程中，嵌入层不仅考虑节点自身的特征，还会综合其邻居节点的特征，通过这种邻域聚合机制，每个节点的嵌入能够捕获其在图结构中的局部邻域信息。嵌入层不仅处理单个城市的特征，还考虑了城市之间的相互关系，见公式（7）。该公式描述了将输入特征向量 $\boldsymbol{m}_i$ 进行嵌入操作的过程，其中，$\boldsymbol{m}_i^{emb}$ 表示嵌入后的特征向量，$\boldsymbol{m}_i$ 是原始输入特征向量，$\boldsymbol{W}_{emb}$ 是一个线性变换矩阵，$\mathrm{Conv}(\boldsymbol{M}_i^k)$ 是一个卷积操作，用于提取局部邻近信息，设 $\boldsymbol{M}_i^k$ 为第 i 个节点的特征向量及其在距离上最接近第 i 个结点的 k 个最近的相邻结点特征向量的级联。公式（7）的含义是将输入特征向量 $\boldsymbol{m}_i$ 与线性变换矩阵 $\boldsymbol{W}_{emb}$ 进行点积操作，得到一个新的特征向量，然后将这个新的特征向量与局部邻近信息（由卷积操作提取）进行连接，形成最终的嵌入特征向量 $\boldsymbol{m}_i^{emb}$。通过将原始特征与局部邻近信息结合起来，嵌入特征可以捕捉到更多的结构信息，从而提高神经网络的性能。其中，$\boldsymbol{I}^0 = \{\boldsymbol{m}_f, \boldsymbol{m}_1^{emb}, \cdots, \boldsymbol{m}_n^{emb}\} \in R^{(n+1)\times d}$。这里，第一个编码器层的输入 $\boldsymbol{I}^0$，是通过将开始标记 $\boldsymbol{m}_f$ 与 $\{\boldsymbol{m}_1^{emb}, \cdots \boldsymbol{m}_n^{emb}\}$ 连接而创建的，本文是添加 $\boldsymbol{m}_f$ 来创建一个虚拟节点特征向量，该向量学习与其他节点特征的依赖关系，以便解码时可以在尽可能好的位置开始。

$$\boldsymbol{m}_i^{emb} = \boldsymbol{m}_i \boldsymbol{W}_{emb} + \mathrm{Conv}(\boldsymbol{M}_i^k) \tag{7}$$

每个编码器层有两个子层：MHSA 子层和 FFNN 子层。MHSA 子层输出的公式见（8）。这个公式描述了在第 N 层编码器中，输入矩阵 $\boldsymbol{I}^{N-1}$ 经过多头自注意力机制和可逆残差连接处理后得到的输出矩阵 $\boldsymbol{Out}^N$。$\boldsymbol{I}^{N-1}$ 是 GCN-Pointransformer 模型编码器的第 $N\text{-}1$ 层的输入矩阵，$\boldsymbol{Out}^N$ 是 GCN-Pointransformer 模型编码器的第 N 层输出矩阵，Batch^N 是第 N 层编码器中的批量归一化操作，

SA^{N} 是第 N 层编码器中的多头自注意力机制。函数 $\mathrm{SA}^{\mathrm{N}}(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V})$ 接受三个输入：$\boldsymbol{Q}$，$\boldsymbol{K}$，$\boldsymbol{V}$，分别表示查询、键和值向量，在第 N 个编码器层执行多头自注意力机制。在这个公式中，首先将输入矩阵 $\boldsymbol{I}^{N-1}$ 与多头自注意力机制处理后的矩阵求和，再添加上可逆残差连接的输入矩阵 $\boldsymbol{I}^{N-1}$。最后，将这个结果送入批量归一化操作 $\mathrm{Batch}^{\mathrm{N}}$ 中，得到第 N 层编码器的 MHSA 子层输出矩阵 $\boldsymbol{Out}^{N}$。

$$\boldsymbol{Out}^{N}=\mathrm{Batch}^{\mathrm{N}}(\boldsymbol{I}^{N-1}+\mathrm{SA}^{\mathrm{N}}(\boldsymbol{I}^{N-1},\boldsymbol{I}^{N-1},\boldsymbol{I}^{N-1})) \tag{8}$$

由两个线性投影和一个 ReLU 激活组成的第 N 个编码器层中的 FFNN 的输入是 $\boldsymbol{Out}^{N}$。它执行非线性激活，然后进行可逆残差连接和批量归一化，并产生输出 $\boldsymbol{I}^{N}$，见公式（9）。其中 FF^{N} 是第 N 个编码器层的 FFNN 子层。

$$\boldsymbol{I}^{N}=\mathrm{Batch}^{\mathrm{N}}(\boldsymbol{Out}^{N}+\mathrm{FF}^{\mathrm{N}}(\boldsymbol{Out}^{N})) \tag{9}$$

在可逆残差网络中[23-25]，MHSA 和 FFNN 维护一对输入和输出嵌入特征，假设 (m_1,m_2) 和 (y_1,y_2)，见公式（10）和（11）。该公式描述了在模型中使用的可逆残差网络的计算过程。在可逆残差网络中，输入特征首先经过一个 MHSA 子层，然后加上原始输入特征 m_1 得到 y_1。接下来，y_1 经过一个 FFNN 子层，然后加上原始输入特征 m_2 得到 y_2 这个过程保证了在反向传播过程中，梯度可以直接计算，从而节省了内存和计算时间。

$$y_1=m_1+\mathrm{MHSA}(m_2) \tag{10}$$

$$y_2=m_2+\mathrm{FFNN}(y_1) \tag{11}$$

因此，输入嵌入特征 (m_1,m_2) 可以在反向传播期间轻松从输出嵌入 (y_1,y_2)，见公式（12）和（13）。

$$m_2=y_2-\mathrm{FFNN}(y_1) \tag{12}$$

$$m_1=y_1-\mathrm{MHSA}(m_2) \tag{13}$$

GCN-Pointransformer 模型编码器的最后一步是编码器的输出，图 3 中 $\boldsymbol{e}_i^{N}$ 是第 N 个编码器层的第 i 个节点的编码器输出，f 是开始标记的索引，第 N 个编码器层的最终输出 $\boldsymbol{I}^{N}=\{\boldsymbol{e}_f^{N},\boldsymbol{e}_1^{N},\cdots,\boldsymbol{e}_n^{N}\}$ 产生并被送到解码器中。

2.2 GCN-Pointransformer 模型解码器

这里一次执行一个节点的自动回归解码，解码器由四个层组成，分别是：多头局部自注意力（Multi Head Local Self Attention，MHLSA）层、掩码多头注意力（Masked Multi-Head Attention，MMHA）层、前馈神经网络（Feed Forward Neural Network，FFNN）层、Pointer 指针层。每个层后面都有残差连接和层归一化。如图 4 所示，该图列举了当前时间步长 TS 取 10，参考向量数 v 取 3 时的解码过程，$\boldsymbol{e}_{r_7}^{N},\boldsymbol{e}_{r_8}^{N},\boldsymbol{e}_{r_9}^{N}$ 是选取的参考向量，r_7,r_8,r_9 是第 7、8、9 个解码步骤中选择的索引。选取的 3 个参考向量通过位置编码后得到：$\boldsymbol{D_7},\boldsymbol{D_8},\boldsymbol{D_9}$。

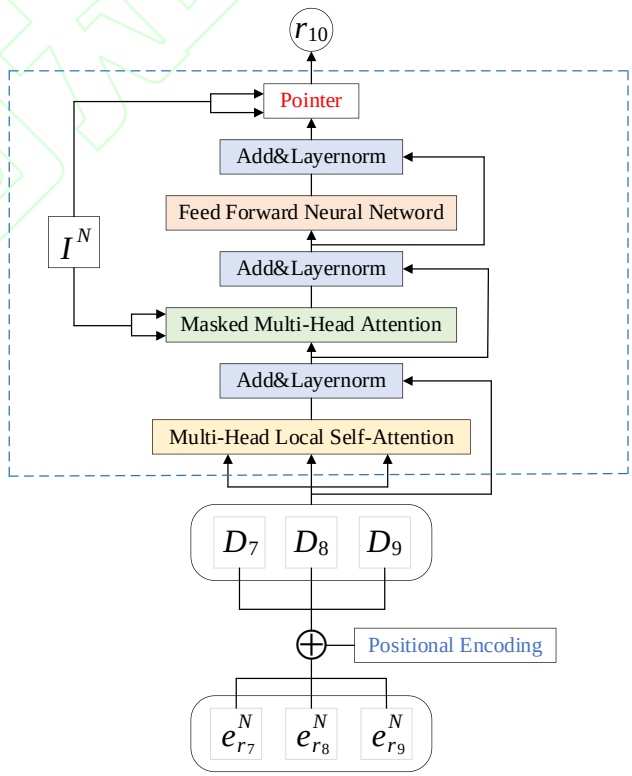


图 4 解码器整体结构

Fig.4 Overall structure of the decoder

在 MHLSA 层中，通过对先前步骤中已经访问过的节点的编码器输出进行注意力操作来提取之前的信息，与标准的 Transformer 模型不同，这里使用部分参考向量来提高性能和计算效率。最近的访问的节点与当前步骤中要选择的节点更相关，所以

不必访问更早的节点，本文提出的 MHLSA 仅在最近访问的节点上使用注意力机制，能够更好的学习局部性组合，降低计算复杂度。

公式（14）是 MHLSA 取当前时间步长为 *TS* 的解码器输入，解码器的输入作为查询用 $\boldsymbol{d}_{TS}$ 表示，其中 $r_0 = f$，$\boldsymbol{PE}_{TS}$ 表示时间步长 *TS* 的位置编码。

$$\boldsymbol{d}_{TS} = \boldsymbol{e}^{N}_{\boldsymbol{r}_{TS-1}} + \boldsymbol{PE}_{TS} \quad (14)$$

所提出的 MHLSA 仅使用最后访问的 v 个节点的解码器输入作为参考向量，例如，假设当前时间步长为 *TS*，则在时间步长 $TS-v$ 到 *TS* 使用的解码器输入被用作参考向量，表示为 $\boldsymbol{D}_{TS} = \{\boldsymbol{d}_{TS-v}, \cdots, \boldsymbol{d}_{TS}\} \in R^{v \times d}$。因此，显著减少了内存使用和计算时间。MHLSA 层的输出用 $\boldsymbol{d}'_{TS}$ 表示，公式见（15）。

$$\boldsymbol{d}'_{TS} = \mathrm{Layer}(\boldsymbol{d}_{TS} + \mathrm{SA}^{N\text{-}1}(\boldsymbol{d}_{TS}, \boldsymbol{D}_{TS}, \boldsymbol{D}_{TS})) \quad (15)$$

MMHA 层执行注意力机制时，其中查询是 MHLSA 层的输出，即 $\boldsymbol{d}'_{TS}$，参考向量是未访问节点的编码器输出 $\boldsymbol{I}^{N}$。MMHA 的输出为 $\boldsymbol{d}''_{TS}$，见公式（16）。这里设 $\boldsymbol{\zeta}_{TS} \in R^{n+1}$ 是掩码，其中未访问节点的值设为 1，访问节点的值设为 0。$\mathrm{MMHA}(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V},\boldsymbol{\zeta})$ 是 $\mathrm{SA}(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V})$ 的修正函数，它增加了一个输入掩码 $\boldsymbol{\zeta}$。$\mathrm{MMHA}(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V},\boldsymbol{\zeta})$ 函数见公式（17）。

$$\boldsymbol{d}''_{TS} = \mathrm{Layer}(\boldsymbol{d}'_{TS} + \mathrm{MMHA}^{N+1}(\boldsymbol{d}'_{TS}, \boldsymbol{I}^{N}, \boldsymbol{I}^{N}, \boldsymbol{\zeta}_{TS})) \quad (16)$$

$$\mathrm{MMHA}(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V},\boldsymbol{\zeta}) = \mathrm{softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^{T}}{d_k} \odot \boldsymbol{\zeta})\boldsymbol{V} \quad (17)$$

FFNN 层的输入是 $\boldsymbol{d}''_{TS}$，设输出为 $\boldsymbol{d}'''_{TS}$，公式见（18）。

$$\boldsymbol{d}'''_{TS} = \mathrm{Layer}(\boldsymbol{d}''_{TS} + \mathrm{FFNN}^{N+1}(\boldsymbol{d}''_{TS})) \quad (18)$$

Pointer 层的目标是通过计算未访问节点的概率分布来选择下一个要访问的节点。这里使用 $\boldsymbol{d}'''_{TS}$ 作为查询，$\boldsymbol{I}^{N}$ 作为参考向量来执行单头注意力，注意力权重作为概率分布记为 p_{TS}，见公式（19）。$\boldsymbol{\zeta}_{TS}$ 用于避免已经访问过的节点，C 是控制 logits 范围的超参数[26]。

$$p_{TS} = \mathrm{softmax}(C \cdot \tanh(\frac{\boldsymbol{d}'''_{TS} \cdot (\boldsymbol{I}^{N})^{T}}{\sqrt{d}})) \quad (19)$$

在训练阶段，解码器将 p_{TS} 视为采样节点索引的分类分布，在推理阶段选择概率最高的节点索引。这个过程重复 n 次，得到一个节点索引序列 $R = \{r_1, \cdots, r_n\}$，这是模型的最终输出[27]。

2.3 基于强化学习的模型训练

网络的优化目标是最小化路径的期望，把平均的行程长度作为损失函数，设 θ 为训练模型的参数，模型生成路线 R 的概率记作：$p(R;\theta)$，公式见（20）。其中，$p(r_{TS} \mid r_1, \cdots, r_{TS-1}; \theta)$ 是在时间步长为 *TS* 时，从 p_{TS} 中选择 r_{TS} 的概率。

$$p(R;\theta) = \prod_{TS=1}^{n} p(r_{TS} \mid r_1, \cdots, r_{TS-1}; \theta) \quad (20)$$

本文使用 REINFORCE 算法来更新 θ 参数，把 θ' 用作基线，设 R' 是由 θ' 参数化的模型，并以贪婪解码的方式生成的索引序列。损失函数记为 $\mathrm{D}(\theta)$，并使用梯度下降算法优化 $\mathrm{D}(\theta)$，见公式（21），（22）。其中，$\mathrm{b}(R)$ 和 $\mathrm{b}(R')$ 分别表示训练模型和基线模型的节点序列的总路程。当完成一次训练时，比较训练模型和基线模型的平均行程长度，如果训练模型的平均行程长度比基线模型的平均行程长度短，就把 θ 赋值给 θ'。

$$\mathrm{D}(\theta) = E_{R \sim p(R;\theta)}[b(R) - b(R')] \quad (21)$$

$$\nabla_{\theta} \mathrm{D}(\theta) \approx \sum_{R} (\mathrm{b}(R) - \mathrm{b}(R'))\nabla_{\theta} \log p(R;\theta) \quad (22)$$

3 实验

3.1 实验配置

使用镜像 Miniconda conda3，Python3.8(ubuntu 18.04)，在 PyCharm 软件开发平台运行，使用 pytorch 编程框架，pytorch 版本为 2.2.1+CUDA12，GPU 为 NVIDIA GeForce RTX 2080 Ti，内存为 40GB。

3.2 实验数据集

本文使用随机数据集和 TSPLIB 标准数据集。

3.2.1 随机数据集

从$[0,1]^2$的单位正方形中随机采样一定数量的节点用来模型的训练和测试。本实验假设一个二维平面对称 TSP，对节点数量 n=50（TSP50）和 100 个节点（TSP100）的 TSP 问题进行了训练和测试，并为每个问题生成了 10000 个测试实例。

3.2.2 TSPLIB 标准测试数据集

这个库由德国计算机科学家 Gerhard Reinelt 在 1990 年代初期创建，旨在提供一个共享的、标准化的测试集，以便研究者和算法开发者能够在一个公平、一致的基础上评估和比较他们的解决方案。

这里使用一个关键参数值 $\sqrt{\frac{l}{N\cdot A}}$ 来评估 TSPLIB 的难度水平，其中 N 是城市数量，A 是城市覆盖的面积，其中 l 是最佳行程，参数值接近 0.75 表示较高的难度级别[28]。本实验从 TSPLIB 中选择了 10 个被认为是相对困难的二维欧几里得问题实例，并且规范化每个 TSPLIB 实例，使得所有 TSPLIB 例子都在$[0,1]^2$中。

3.3 评估指标及实验参数

实验采用预测的平均行程长度 $\text{avg}(len_i')$、优化间隙 Og（%）、训练时间和 GPU 内存使用作为衡量模型性能的评价标准。平均行程长度见公式（23），len_i'是第 i 个实例的预测行程长度，n 是总测试实例的数量，这里取 10000。通过计算平均旅行长度与理论最优旅行长度之间的比值，再减去 1 来计算优化间隙，见公式（24）。优化间隙用于衡量当前解的质量，较小的优化间隙表示当前解接近最优解，而较大的优化间隙表示当前解与最优解之间存在较大的差距。

$$\text{avg}(len_i')=\frac{1}{n}\sum_{i=1}^{n}len_i' \tag{23}$$

$$\text{Og}=\frac{1}{n}\sum_{i=1}^{n}\left(\frac{len_i'}{len_i}-1\right) \tag{24}$$

实验所需的超参数设置见表 1。分别是嵌入维度、前馈网络维度、编码器层数、解码器层数、KNN 算法的邻居数量、每个批次处理的样本数量、卷积核的大小、多头注意力的头数、训练周期、每个训练周期 *epoch* 中要处理的批次 *batch* 数量、学习率、解码器中的 logits 范围裁剪值 C。

表 1 超参数设置

Table 1 Hyperparameter settings

Parameter	Values
dim_emb	128
dim_ff	512
encoders_layers(N)	6
decoders_layers	2
nb_neighbors(KNN)	10
bsz	512
kernel_size	11
nb_heads	8
nb_epochs	100
nb_batch_per_epoch	2500
learning_rate	1e-4
C	10

3.4 对比实验结果

表 2 展示了解码器中提出的多头局部自注意力是否比求解 TSP 的标准 Transformer[21]解码器中的全连接自注意力更有效地提高模型的性能。这里通过将参考向量的数量 v 从 100 减少到 5 来测试，使用随机数据集。本文模型在 TSP50 和 TSP100 上具

有不同v值的平均行程长度和优化间隙。这里，$v=100$表示完全连接的注意力模型，而其他v值表示部分连接的注意力模型。表 2 显示了本文的模型在不同 v 值上的平均行程长度和优化间隙的数值变化。以 TSP100 为例，参考向量$v=5$时比$v=100$时的平均行程长度减少约 0.250%，优化间隙减少约 7.692%。平均行程长度和优化间隙的数值越小代表解的质量越好。实验结果表明，随着 v 值的减小，模型的性能有所提高，v 值为 5 时得到了最佳输出结果。可得结论：过度的注意力连接会降低模型性能，与求解 TSP 的标准 Transformer 模型中的完全连接注意力相比，本文提出的多头局部自注意力能专注于最近的节点，在提高性能方面十分有效。

表 2　多头局部自注意力与全连接自注意力

Table 2　Multi-head local self-attention and fully connected self-attention

	TSP50		TSP100	
参考向量	平均行程长度	优化间隙	平均行程长度	优化间隙
100	---	---	8.008	3.12%
50	5.752	1.05%	7.999	2.98%
20	5.750	1.02%	7.992	2.92%
5	5.747	0.97%	7.988	2.88%

本文还将所提出的可逆残差网络与标准 TSP Transformer 模型中的残差网络之间的计算复杂性进行了比较，用 10000 个 TSP 实例的训练时间和 100 个 TSP 实例的推理时间，使用随机数据集。表 3 总结了模型的复杂性和运行时间，展示了本文提出的可逆残差网络比标准 TSP Transformer 解码器中的残差网络提升了实验效果。可逆结构可以在执行反向传播时重新计算而不是存储前向传播中的中间激活，能够在编码器中显著降低内存消耗，使得在处理更大规模问题时内存消耗不会急剧增加。这里的“GPU 内存”是指在训练过程中使用批处理大小为 512 的 GPU 内存使用情况。与 TSP Transformer 模型相比，所提出的模型消耗的 GPU 内存减少了大约 11%。本文模型的另一个优势在于其推理时间明显更快，TSP100 的推理时间比 TSP Transformer 模型推理时间减少约 25%，而且优化间隙减少了 12%，说明解的质量更好。

表 4 给出了本文的模型和求解 TSP 的标准 Transformer 模型在 TSPLIB 实例上的输出结果。两种模型均采用 TSP50 进行训练。通过表 4 可以观察到：除了 rd100 之外，GCN-Pointransformer 模型在所有 TSPLIB 实例中都优于 TSP Transformer，即在绝大多数 TSPLIB 实例中,所提出的模型均优于 TSP Transformer。

表 5 将所提出的模型与大部分其他求解 TSP 的模型的性能进行比较，使用随机数据集。Concorde 和 Gurobi 属于混合整数规划求解器；OR-Tools、LKH-3、Farthest Insertion 是启发式方法求解 TSP；Bello et-al 到 Vinyals et-al 方法是利用神经网络求解 TSP；TSP Transformer 是基于标准 Transformer 模型求解 TSP；Wu et-al 和 Costa et-al 是神经改进算法求解 TSP；Jung et-al 和 Xiao et-al 属于端到端模型求解 TSP。以上均使用贪婪解码。实验结果表明，能精确求解 TSP 的 Concorde 显示出最佳性能，其次就是本文提出的 GCN-Pointransformer 模型，在 TSP50 上，优化间隙为 0.31%，在 TSP100 上，优化间隙为 0.42%均取得了优异的结果，优化间隙越小代表所求结果更优。

表 3　模型复杂性与运行时间

Table 3　Model complexity and running time

模型	平均行程长度	优化间隙	GPU 占用	训练时间	推理时间

TSP Transformer	7.860	1.25%	18.16	18.89s	76.96s
GCN-Pointransformer	7.844	1.10%	16.13	19.20s	57.88s

表 4　性能比较

Table 4　Performance comparison

问题实例	关键参数	Concorde	TSP Transformer		GCN-Pointransformer	
			平均行程长度	优化间隙	平均行程长度	优化间隙
berlin52	0.74	7542	7637	1.26%	7588	0.87%
ch130	0.78	6110	6569	7.51%	6550	7.19%
ch150	0.78	6528	7390	13.20%	7046	7.97%
eli51	1.05	426	438	2.82%	424	0.67%
eli76	1.03	538	565	5.02%	562	4.82%
eli101	0.98	629	681	8.27%	670	6.98%
kroA100	0.77	21282	21747	2.18%	21617	1.59%
kroC100	0.75	20749	21788	5.01%	21499	3.69%
rd100	0.81	7910	8078	2.12%	8209	3.89%
St70	0.86	675	710	5.19%	669	0.13%

表 5　与其他求解 TSP 的模型比较

Table 5　Comparison with other models that solve TSP

方法	TSP50		TSP100	
	平均行程长度	优化间隙	平均行程长度	优化间隙
Concorde[7]	5.689	0.00%	7.765	0.00%
Gurobi[16]	---	0.00%	7.765	0.00%
OR-Tools[29]	5.800	1.83%	7.990	2.90%
LKH-3[30]	---	0.00%	7.765	0.00%
Farthest Insertion[31]	6.010	5.53%	8.350	7.59%
Bello et-al[26]	5.950	4.46%	8.300	6.90%
Dai et-al[32]	5.990	5.16%	8.310	7.03%
Deudon et-al[33]	5.810	2.07%	8.850	13.97%
Joshi et-al[12]	5.870	3.10%	8.410	8.38%
Kool et-al[18]	5.800	1.76%	8.120	4.53%
Vinyals et-al[10]	7.660	34.48%	---	---
TSP Transformer[20]	5.750	1.05%	8.015	3.22%
Wu et-al(T=1000)[19]	5.740	0.89%	8.010	3.16%
Jung et-al[27]	5.754	1.13%	7.985	2.83%
Xiao et-al[34]	5.752	1.11%	7.899	1.74%

Costa et-al(T=1000)	5.735	0.81%	7.851	1.10%
GCN-Pointransformer	5.701	0.31%	7.800	0.42%

如图 5 至图 9 分别是在本文提出的模型下，困难级别较高的 TSPLIB 实例路线图，它们的关键参数相对比较接近 0.75，分别是：0.74、0.75、0.77、0.78、0.78。它们均在该文提出的模型上取得了更加理想的结果。

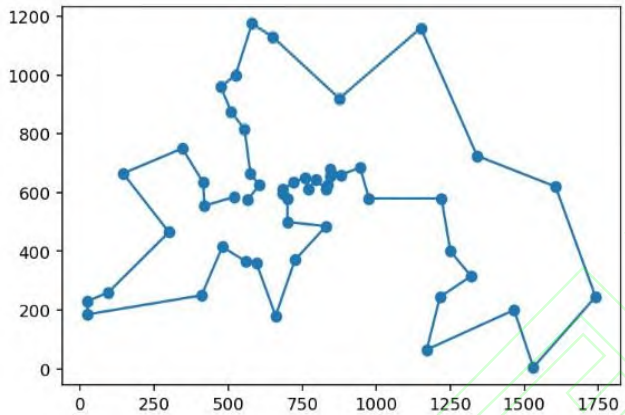


图 5　berlin52 路线图

Fig.5　berlin52 roadmap

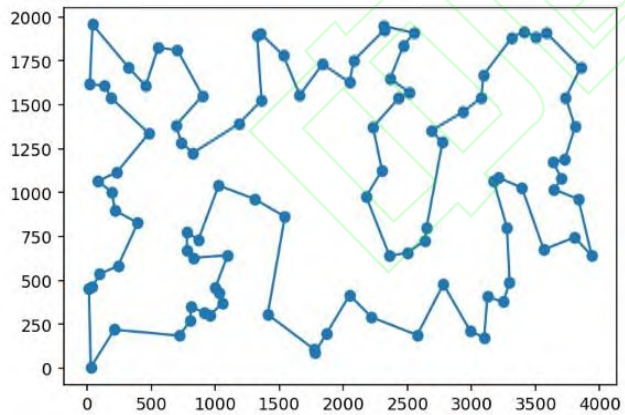


图 6　kroC100 路线图

Fig.6　kroC100 roadmap

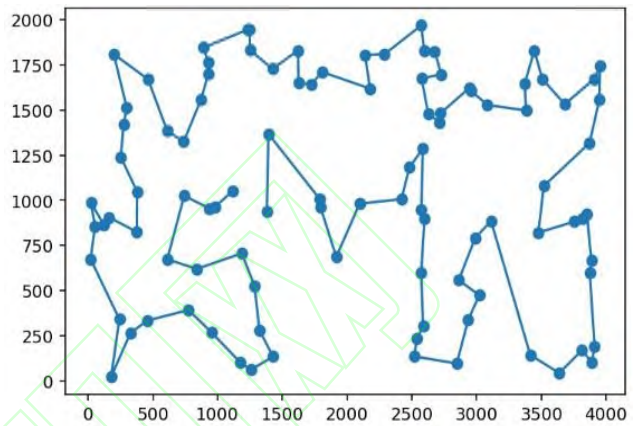


图 7　kroA100 路线图

Fig.7　kroA100 roadmap

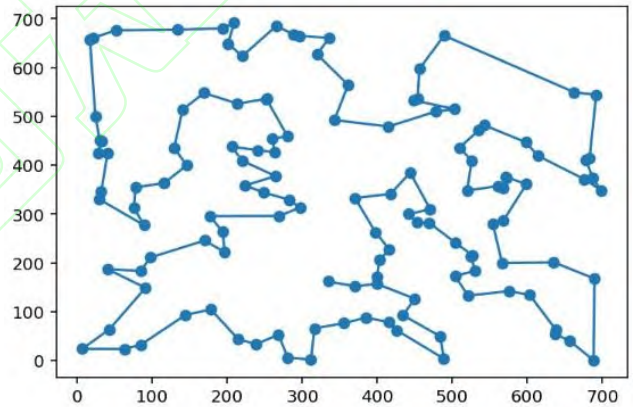


图 8　ch130 路线图

Fig.8　ch130 roadmap

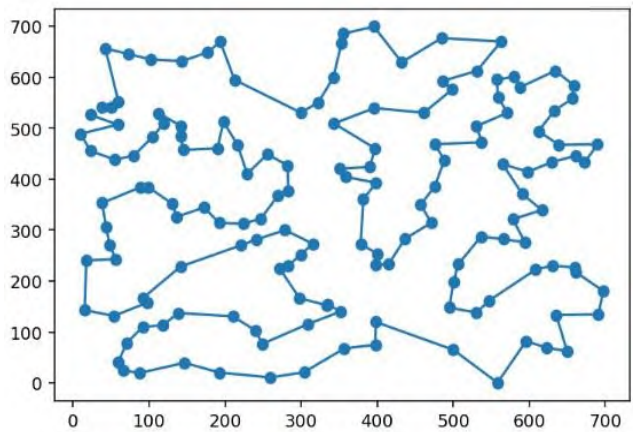


图 9　ch150 路线图

Fig.9　ch150 roadmap

通过使用随机数据集 TSP50 和 TSP100 以及

TSPLIB 数据集进行对比实验，实验结果表明，它们均在该文提出的模型上取得了理想的结果，即平均行程长度和优化间隙在该文提出的模型上获得了更小值。可得出结论：该方法在处理旅行商问题方面具有较高的准确性和质量。

3.5 消融实验结果

分别去除了 GCN-Pointransformer 模型中的图卷积嵌入层、可逆残差连接、Pointer 层，通过进行消融研究来测试它们是否能产生更好的输出结果，这里使用随机数据集。表 6 显示了消融研究的结果，平均行程长度和优化间隙衡量了求解的质量。在 TSP50 上，添加图卷积嵌入层时的平均行程长度比去除图卷积嵌入层时的平均行程长度减少约 11.855%，优化间隙减少 51.556%；使用可逆残差连接时的平均行程长度比去除可逆残差连接时的平均行程长度减少约 7.846%，优化间隙减少 39.106%；不增加 Pointer 层时的平均行程长度比去除 Pointer 层时的平均行程长度减少约 2.978%，优化间隙减少 8.403%；在 TSP100 上，添加除图卷积嵌入层时的平均行程长度比去除图卷积嵌入层时的平均行程长度减少约 7.009%，优化间隙减少 13.897%；使用可逆残差连接时的平均行程长度比去除可逆残差连接时的平均行程长度减少约 6.411%，优化间隙减少 12.577%；增加 Pointer 层时的平均行程长度比去除 Pointer 层时的平均行程长度减少约 2.803%，优化间隙减少 4.362%。实验结果表明，本文提出的 GCN-Pointransformer 模型可以有效地求解 TSP，产生了更佳的结果。

表 6 消融实验结果

Table 6 Comparison with other models that solve TSP in part

比较	TSP50 平均行程长度	优化间隙	TSP100 平均行程长度	优化间隙
去除图卷积嵌入层	6.689	2.25%	9.545	3.31%
去除可逆残差连接	6.398	1.79%	9.484	3.26%
去除 Pointer 层	6.077	1.19%	9.132	2.98%
GCN-Pointransformer	5.896	1.09%	8.876	2.85%

4 结论

本文所提出的 GCN-Pointransformer 模型能够从输入数据中提取和学习空间特征，还能够更好地学习局部组合性，并与基于标准的 Transformer 的模型相比产生了更好的 TSP 解决方案。通过实验还观察到，所提出的模型通过在解码器中应用多头局部自注意力显着减少了 GPU 内存使用和推理时间。在随机数据集和真实世界数据集上，问题的解决方案质量、GPU 内存使用和推理时间方面优于现有的求解 TSP 的标准 Transformer 模型。但是，文章中使用的实验数据和评估指标相对有限，这可能影响到结果的可靠性和推广性，并且在处理更大规模实例时，模型的计算成本可能仍然较高。未来应进一步解决上述问题，提升数据的准确性。

参考文献：

[1] BELLMAN R. Dynamic programming treatment of the travelling salesman problem[J]. Journal of the ACM (JACM), 1962, 9(1): 61-63.

[2] 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.

[3] DANTZIG G, FULKERSON R, Johnson S. Solution of a large-scale traveling-salesman problem[J]. Journal of the operations research society of America, 1954, 2(4): 393-410.

[4] KIRKPATRICK S, GELATT JR C D, VECCHI M P. Optimization by simulated annealing[J]. science, 1983, 220(4598): 671-680.

[5] CROES G A. A method for solving traveling-salesman problems[J]. Operations research, 1958, 6(6): 791-812.

[6] Bixby R, Rothberg E. Progress in computational mixed integer programming--a look back from the other side of the tipping point[J]. Annals of Operations Research, 2007, 149(1): 37.

[7] COOK W J, APPLEGATE D L, BIXBY R E, et al. The traveling salesman problem: a computational study[M]. Princeton university press, 2011.

[8] CHRISTOFIDES N. Worst-case analysis of a new heuristic for the travelling salesman problem[C]// Operations Research Forum. Cham: Springer International Publishing, 2022, 3(1): 20.

[9] PERRON L, FORTUNATO V. Or-tools[J/OL]. 2022-11-2. URL:https://developers.google.com/optimization/VINYAL S O, FORTUNATO M, JAITLY N. Pointer networks[J]. Advances in neural information processing systems, 2015, 28.

[10] NAZARI M, OROOJLOOY A, SNYDER L, et al. Reinforcement learning for solving the vehicle routing problem[J]. Advances in neural information processing systems, 2018, 31.

[11] 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.

[12] STOHY A, ABDELHAKAM H T, ALI S, et al. Hybrid pointer networks for traveling salesman problems optimization[J]. Plos one, 2021, 16(12): e0260995.

[13] MA Q, GE S, HE D, et al. Combinatorial optimization by graph pointer networks and hierarchical reinforcement learning. arXiv 2019[J]. arXiv preprint arXiv:1911.04936, 1911.

[14] MIKI S, EBARA H. Solving traveling salesman problem with image-based classification[C]//2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI). IEEE, 2019: 1118-1123.

[15] LING Z, TAO X, ZhANG Y, et al. Solving optimization problems through fully convolutional networks: An application to the traveling salesman problem[J]. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2020, 51(12): 7475-7485.

[16] SULTANA N, CHAN J, SARWAR T, et al. Learning to optimise general TSP instances[J]. International Journal of Machine Learning and Cybernetics, 2022, 13(8): 2213-2228.

[17] KOOL W, VAN HOOF H, Welling M. Attention, learn to solve routing problems![J]. arXiv preprint arXiv:1803.08475, 2018.

[18] WU Y, SONG W, CAO Z, et al. Learning improvement heuristics for solving routing problems[J]. IEEE transactions on neural networks and learning systems, 2021, 33(9): 5057-5069.

[19] BRESSON X, LAURENT T. The transformer network for the traveling salesman problem[J]. arXiv preprint arXiv:2103.03012, 2021.

[20] GUO Q, QIU X, LIU P, et al. Star-transformer[J]. arXiv preprint arXiv:1902.09113, 2019.

[21] 程荣.遗传算法求解旅行商问题[J].科技风, 2017, (16): 40+51.
CHENG Rong. Genetic algorithm to solve the travel ing salesman problem[J].Science & Technology Wind, 2017,(16):40+51.)

[22] GOMEZ A N, REN M, URTASUN R, et al. The reversible residual network: Backpropagation without storing activations[J]. Advances in neural information processing systems, 2017, 30.

[23] KITAEV N, KAISER Ł, LEVSKAYA A. Reformer: The efficient transformer[J]. arXiv preprint arXiv:2001.04451, 2020.

[24] JIN Y, DING Y, PAN X, et al. Pointerformer: Deep reinforced multi-pointer transformer for the traveling salesman problem[C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2023, 37(7): 8132-8140.

[25] BELLO I, PHAM H, LE Q V, et al. Neural Combinatorial

Optimization with Reinforcement Learning[J]. arXiv: Artificial Intelligence,arXiv: Artificial Intelligence, 2016.

[26] JUNG M, LEE J, KIM J. A Lightweight CNN- Transformer Model for Learning Traveling Salesman Problems[J]. arXiv preprint arXiv:2305.01883, 2023.

[27] SULTANA N, CHAN J, SARWAR T, et al. Learning to optimise general TSP instances[J]. International Journal of Machine Learning and Cybernetics, 2022, 13(8): 2213-2228.

[28] Google. 0R-tools: Google's Operations Research tools [EB/OL]. (2015). Retrieved from https://gitcode.com/google/or-tools.

[29] HELSGAUN K. An extension of the Lin-Kernighan-Helsgaun TSP solver for constrained traveling salesman and vehicle routing problems[J]. Roskilde: Roskilde University, 2017, 12: 966-980.

[30] JOHNSON D S. Local optimization and the traveling salesman problem[C]//International colloquium on automata, languages, and programming. Berlin, Heidelberg: Springer Berlin Heidelberg, 1990: 446-461.

[31] KHALIL E, DAI H, ZHANG Y, et al. Learning combinatorial optimization algorithms over graphs[J]. Advances in neural information processing systems, 2017, 30.

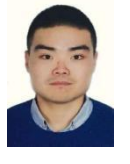
[32] DEUDON M, COURNUT P, LACOSTE A, et al. Learning heuristics for the tsp by policy gradient[C]// Integration of Constraint Programming, Artificial Intelligence, and Operations Research: 15th International Conference, CPAIOR 2018, Delft, The Netherlands, June 26–29, 2018, Proceedings 15. Springer International Publishing, 2018: 170-181.

[33] XIAO Y, WANG D, CHEN H, et al. Reinforcement learning-based non-autoregressive solver for traveling salesman problems[J]. arXiv preprint arXiv:2308.00560, 2023.

[34] DA COSTA P, RHUGGENAATH J, ZHANG Y, et al. Learning 2-opt heuristics for routing problems via deep reinforcement learning[J]. SN Computer Science, 2021, 2: 1-16.

邱云飞(1978—),男,辽宁省阜新市,博士,教授,主要研究方向为数据挖掘等。
QIU Yunfei, born in 1978, Ph.D, professor.His research interests include data mining, etc.

刘一菲(2000—),女,辽宁省开原市,硕士研究生,主要研究方向为旅行商问题、智能算法应用等。
LIU Yifei, born in 2000, Candidate M.S., Her research interests include Traveling Salesman Problem, intelligent algorithm application, etc.



于智龙(1993—),男,山东省荣成市,博士研究生,主要研究方向为智能算法应用等。
YU Zhilong,born in 1993,Candidate Ph.D. His research interests include intelligent algorithm application, etc.

金海波(1983—),男,辽宁省沈阳市,博士,副教授,主要研究方向为复杂系统可靠性分析等。
JIN Haibo, born in 1983, Ph.D, associate professor. His research interests include reliability analysis of complex systems, etc.