上海交通大學

SHANGHAI JIAO TONG UNIVERSITY

学士学位论文

THESIS OF BACHELOR

论文题目： 双向深度学习辅助 TSP 问题求解

学生姓名:	李铭飞
学生学号:	516021910105
专　　业:	计算机科学与技术
指导教师:	涂仕奎副教授
学院 (系):	电子信息与电气工程学院

双向深度学习辅助 TSP 问题求解

摘　要

本文提出一种双向深度学习系统，将图卷积神经网络（GCN）和传统组合优化算法结合，从学习的角度来求解旅行商问题（TSP 问题）。该系统一方面使用图卷积神经网络学习图像的边特征和结点特征，给出对边 e_{ij} 出现在 TSP 路径的概率预测 p_{ij}，从而将输入的 2D 图像映射到邻接概率矩阵 P；另一方面，使用传统的组合优化迭代算法或启发式搜索算法来将概率矩阵转化为合法的 TSP 路径, 进而将该路径矩阵回传给图卷积神经网络与其输出邻接概率矩阵计算交叉熵进行反向传播，使用梯度下降法减小交叉熵来更新图卷积神经网络的参数。图卷积神经网络和传统算法在多次循环迭代中互相更新参数或输入以提高整个模型的表现。传统组合优化迭代算法在与该双向深度学习系统结合后表现有明显提升。在所有尝试的组合中，启发式搜索算法中的集束搜索与该双向深度学习系统结合后表现最佳。

关键词：双向深度学习; 旅行商问题; 图卷积神经网络; 贪婪搜索; 集束搜索; 迭代算法

A BIDIRECTIONAL DEEP LEARNING APPROACH FOR THE TRAVELLING SALESMAN PROBLEM

ABSTRACT

In this paper we propose a bidirectional deep learning system which combines graph convolutional network(GCN) and traditional combinatorial optimization algorithms, solving the travelling salesman problem from the aspect of learning. On one hand, we use GCN to extract edge and node features from the input 2D graph, and embed edge features to an adjacency matrix P(p_{ij} denotes the probability that the corresponding edge e_{ij} appears in the TSP tour). On the other hand, we use traditional iterative algorithms or traversal algorithms to derive a valid TSP tour from the adjacency matrix P, and pass the TSP tour back to GCN in order to compute the cross entropy loss for back propagation. We train the GCN model by using gradient decent to minimize the cross entropy loss. During each iteration, GCN and traditional algorithms update each other's parameters or input to enhance the performance of this system. The traditional combinatorial optimization algorithm we use has a significant performance improvement when working with this bidirectional deep learning system. Among all the iterative or heuristic search algorithms we use, beam search works the best with our bidirectional deep learning system.

Key words: bidirectional deep learning; travelling salesman problem; graph convolutional network; greedy search; beam search; iterative algorithm

目 录

第一章　绪论

1.1　TSP 问题简介

组合优化问题，即具有离散变量的最优化问题在日常生活中有着大量的实际应用，被广泛地用于交通运输、物资供应、事务调度等各个领域。组合优化问题的许多典型代表均为 NP 难问题，如集合覆盖问题，生产调度问题，图着色问题等等。这类问题往往易于理解，有着很强的工程代表性。然而这些问题目前还没有能给出最优解的多项式时间空间复杂度的算法，当问题规模较大时精确的最优化求解需要极长的运算时间和极大的储存空间（即“组合爆炸”）以至于目前无法在计算机上实现，只能使用有效的近似算法。

旅行商问题（Travelling Salesman Problem, 即 TSP 问题）是组合优化领域里的最著名的 NP 难问题[1]之一，其内容可以简要叙述为给定一系列的城市和两两之间的距离，找出经过所有城市仅一次且最终回到出发城市的最短回路（即最短哈密尔顿回路）。TSP 问题实际上是一个序列的决策问题，即给出一张图 G，我们需要在节点的排列中找出一个最优序列，使得其对应的路径总长度（权重之和）最小。

假设输入图 G 的形式为图的距离矩阵 D，每个元素 d_{ij} 表示城市 i 与城市 j 之间的距离，则 TSP 问题用数学语言可描述为：

$$
\begin{aligned}
\min \quad & \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} d_{ij}v_{ik}v_{j,k+1} \\
\text{subject to} \quad & \sum_{j=1}^{n} v_{ij} = 1, \quad i = 1,2,\cdots,n \qquad (1) \\
& \sum_{i=1}^{n} v_{ij} = 1, \quad j = 1,2,\cdots,n \qquad (2) \\
& v_{ij} \in \{0,1\}, \quad i = 1,2,\cdots,n, \quad j = 1,2,\cdots,n
\end{aligned}
$$

其中，

$$
v_{ik} = \begin{cases} 1 & \text{城市 i 是 TSP 路径中第 k 个到访的城市} \\ 0 & \text{其他情况} \end{cases}
$$

由于 v_{ij} 的定义和式 (1)(2)，显然 TSP 路径中不存在子回路。若用 v_{ij} 表示边 e_{ij} 在 TSP 路径中的存在性，则需另加一个约束条件 $\sum_i \sum_j v_{ij} \le |V-1|$, 其中 $|V|$ 为 G 的顶点个数。以上约束条件可以根据不同近似算法的需要写成其他等价形式，以便将 v_{ij} 从 01 离散变量放松到区间 [0, 1] 之间的连续变量，具体方式见章节3.1。

TSP 问题作为一个著名的数学问题，历史悠久，其雏形是欧拉 1759 年研究的骑士环游问题，即如何遍历国际象棋棋盘中的每个方格且仅遍历一次，最终返回起点。由于其在交通运输、电路设计、施工方案等领域的广泛应用，国内外学者对其进行了长期的、大量的研究。它是诸多领域内多种复杂问题的抽象与集中，集合覆盖、图着色、子集和等问题都可以在多项式时间内归约为 TSP 问题。因此，快速有效地解决 TSP 问题有着重要的理论意义和高度的实用价值。

1.2　深度神经网络的发展现状

当前，深度神经网络在图像识别、语音识别等任务上，取得了巨大的成功。深度学习通过深度神经网络来提取样本特征，将现实世界中的事物使用更加抽象的概念来定义。近几

[1]这里研究的 TSP 问题指的是寻找图中最短哈密尔顿回路，这不是 NP 问题，所以是 NP 难问题而非 NPC 问题。判定图中是否存在长度小于 k 的哈密尔顿回路则是 NP 完全问题。

年来深度学习的快速发展对机器学习领域，乃至更广泛的科研领域，都产生了深远的影响。深度学习在多个领域的成功主要归功于 GPU 等计算资源的高速发展、训练数据集的丰富与公开，以及深度学习从欧几里得数据（如图像、文本和视频）中提取特征信息的高效性。例如卷积神经网络（Convolutional Neural Network，CNN）可以利用平移不变性、局部连通性和图像数据语意合成性，提取出与整个数据集共享的局部有意义的特征，用于各种图像分析任务。深度神经网络的最新进展推进了模式识别和数据挖掘领域的研究。目标检测、机器翻译、语音识别等许多机器学习任务曾高度依赖手工特征工程来提取信息特征集合，但多种端到端深度学习方式，比如卷积神经网络、长短期记忆网络和自编码器等改变了这种状况。

虽然深度学习在学习与处理欧几里得数据的任务中取得了巨大成功，但现实中大量存在的非欧几里得数据目前来看存在着更广阔的应用范围。对于非欧几里得数据，节点的链接方式可以是千差万别的，比如社交网络，蛋白质结构等等。图数据作为一种非欧几里得形数据，其复杂性对传统机器学习算法提出了重大挑战。由于图是不规则的，每张图大小不同、节点无序，一张图中的每个节点都有不同数目的邻近节点，使得一些在图像中容易计算的重要运算（如卷积）不能再直接应用于图。此外，现有机器学习算法的核心假设 IID，即独立同分布假设，假设输入数据是互相独立的。然而，图数据中的每个实例都与周围的其它实例相关，含有一些复杂的连接信息，用于捕获数据之间的依赖关系，包括引用、邻接关系和相互作用等。

最近，越来越多的研究开始将深度学习方法应用到图数据领域。受到卷积神经网络在计算机视觉领域所获巨大成功的激励，近来出现了很多为抽象图重新定义卷积概念的方法。这些方法属于图卷积神经网络（GCN）的范畴。Bruna et al. (2013) 提出了关于图卷积的第一项重要研究，他们基于谱图论（spectral graph theory）开发了一种图卷积的变体。自此，基于谱的图卷积神经网络不断改进、拓展、进阶。由于谱方法通常同时处理整个图，并且难以并行或扩展到大图上，基于空间的图卷积神经网络开始快速发展。这些方法通过聚集近邻节点的信息，直接在图结构上执行卷积。结合采样策略，计算可以在一个批量的节点而不是整个图中执行，这种做法有望提高效率。除了图卷积神经网络，近几年还开发出了很多替代的图神经网络。这些方法包括图注意力网络、图自编码器、图生成网络以及图时空网络等。

深度神经网络在很多任务上都已比肩人类甚至超越人类，但是还存在如下问题：其一，这些识别任务，边界界定清晰、所需带标签样本量巨大；其二，认知层次较低，远远没有达到产生理解、直觉、顿悟、自我意识的阶段；其三，学习能效还不高，与之相比，人脑能够在功耗为 20 瓦的情况下执行非常复杂的功能。认识到现有神经网络的这些不足，研究者已经开始向减少标签样本数量的方向进行努力，取得了一些成果。例如，AlphaZero 已经实现在棋类的特定场景里，不需要人类棋谱进行自主学习。尽管如此，AlphaZero 框架目前也只适用于棋类的平稳场景，其输赢规律是不变的，要拓展到开放环境、多变场景中，也是一个很大的挑战。

1.3 相关工作

深度神经网络不仅能完成标准图推理任务，如节点分类或图分类等，也能广泛地应用于其他领域，比如求解基于图的组合优化问题。图中的组合优化问题是 NP 难问题的集合，引起了各个领域科学家的广泛关注。组合优化问题大多数情况下都涉及到决策顺序，该领域的许多 NP 难问题本质上可以总结为在输入的拓扑图上进行一系列的决策。深度学习可以完成序列到序列的映射问题，因此使用深度学习来求解组合优化问题是一种可行方案。最

近，使用深度神经网络来解决这些问题已成为热点，并且由于这类问题的图形结构，一些解决方案进一步利用图形神经网络。近年来由 Scarselli et al. (2009) 首次提出并被 Bruna et al. (2013), Defferrard et al. (2016) 等研究者不断改进的图神经网络（Graph Neural Network,GNN）由于其可以处理图这一类非欧几里得数据的能力，已被广泛用于解决一些基于图的组合优化问题，如 Job shop 问题 (Weckman et al. 2008), 最小顶点覆盖问题与最大割问题 (Dai et al. 2017; Venkatakrishnan et al. 2018) 等。

就使用神经网络解决 TSP 问题而言，目前已有不少相关研究。序列到序列学习被较早用来解决 TSP 问题。Vinyals et al. (2015) 提出了一种序列到序列 (seq2seq) 的指针网络 (PtrNet)。该网络使用有监督学习，需要使用求解器给出的最优路径来进行训练。它使用注意力机制输出输入序列的一种排列。在测试阶段，需要使用集束搜索来进行解码，将输出序列转化为合法的 TSP 路径。基于前者的工作，Bello et al. (2016) 提出了一种解决 TSP 的深度学习方法。该方案训练 PtrNet 时是无监督的，不需要求解器提供最优解，而是使用强化学习算法来进行训练，用计算出的 TSP 解的开销（即路径长度）作为策略梯度的无偏差蒙特卡罗估计。

Dai et al. (2017) 使用图神经网络来编码 TSP 问题实例，这种编码方式是节点顺序无关的，比起序列到序列模型能更好的反应 TSP 问题的组合结构。他们通过训练一个结构到向量 (structure2vec) 图映射模型将输入图映射到一个节点序列，使用 DQN 训练方法 (Mnih et al. 2013) 按照这个序列将节点逐个插入到 TSP 部分路径中最可能的位置。这项工作比使用序列到序列学习的方法表现更佳，证明了图神经网络的表示能力。

Kool et al. (2019) 提出了一种基于注意力的编码器-解码器算法。这种方法用 Veličković et al. (2017) 提出的图注意力网络替代 structure2vec 模型，其读出阶段由使用强化学习训练的基于注意力的解码器完成。目前，这个模型在基于学习的 TSP 算法中表现是最优的。

此外，Nowak et al. (2017) 使用 Scarselli et al. (2009) 提出的 GNN 模型进行有监督学习，GNN 用邻接概率矩阵的形式输出其预测的 TSP 路径，然后使用集束搜索 Medress et al. (1977) 来将这个邻接矩阵转化为合法的 TSP 路径。该论文仅提出了适用于 TSP20 这一固定规模问题的模型，并未进行进一步研究，改变节点数量后这个模型表现很差。Bresson et al. (2017) 在 Sainbayar et al. (2016) 提出的 vanilla 图卷积神经网络上整合了 Marcheggiani et al. (2017) 提出的边门控机制，并进一步引入了残差机制，提出了一种残差门控图卷积神经网络模型。之后 Joshi et al. (2019) 在 Nowak et al. (2017) 的工作基础上对 TSP 问题进行进一步探索，通过有监督地训练这个残差门控图卷积神经网络来解决不同规模的 TSP 问题（TSP20,TSP50,TSP100），并沿用了 GNN 给出邻接概率矩阵，用集束搜索将其转化为合法 TSP 路径的方式。该模型对于固定规模、固定大小的输入图像能够给出很好的结果，但是对于不同规模的 TSP 问题，训练出的 GCN 模型并不能通用。本文的双向深度学习系统同样使用了残差门控图卷积神经网络模型 (Bresson et al. 2017)。

第二章　图卷积神经网络

2.1　GCN 简介

图卷积神经网络 (GCN) 由 Kipf et al. (2016) 首次提出，是卷积神经网络 (CNN) 在图这个领域的推广与延伸。它能同时对图的空间信息与拓扑信息进行端到端学习，是目前图数据学习任务的最佳选择，适用于任意拓扑结构的图。Kipf et al. (2016) 的工作展现了 GCN 在应用于节点分类时的惊人表现，在进行空手道俱乐部社交网络图的节点分类时，双层 GCN 网络无需训练就能给出较好的节点分类。GCN 是目前最主流的图神经网络，其余四种图神经网络（图注意力网络、图生成网络、图自编码器和图时空网络）都由 GCN 演变而来。

CNN 作为深度学习的最成功的应用之一，其主要限制在于欧几里得数据, 即有规则空间结构的数据。CNN 能够处理的图像数据中像素点排列成矩阵形式，像素点之间有明确的上下左右的位置关系，故而可以直接应用传统的离散卷积，如图2–1左侧所示。但是在拓扑图里，节点之间的连接与空间上的位置关系无关，每个节点的相邻节点数不同，也就没办法找到一个同样尺寸的卷积核，故无法通过传统卷积进行计算。为了在拓扑图上有效提取结构信息进行机器学习，研究者们对离散卷积进行推广，提出了图卷积这一概念，如图2–1右侧所示。 根据 GCN 模型在图卷积过程中是否使用了傅里叶变换，GCN 可分为基于谱

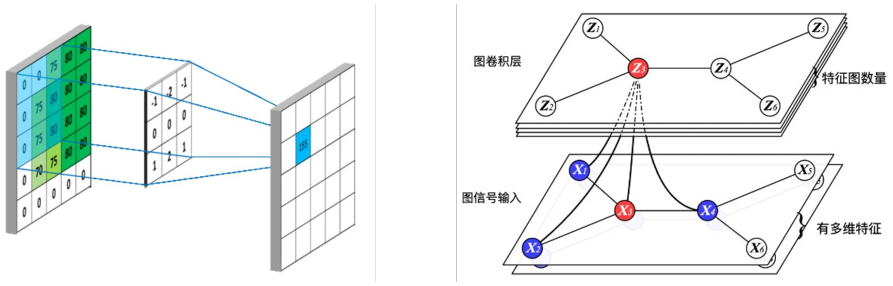

图 2–1 传统卷积[2]（左）与图卷积[3]（右）

(Spectral-based) 和基于空间 (Spatial-based) 两个类别。谱方法起初对图的拉普拉斯矩阵进行谱分解，用得到的特征向量进行傅里叶变换。后来 Defferrard et al. (2016) 对谱方法的卷积核进行改进，不再需要进行谱分解，而是直接使用图的拉普拉斯矩阵进行变换，比如 Li et al. (2018) 使用的 GCN 模型。基于空间的方法如 Dai et al. (2017) 则直接利用图的拓扑结构，根据每个节点的邻接情况进行信息收集。核心思想是对每个节点确定其感受域（即找出其所有“邻居”），通过对不同邻接点设置权值，对邻接点信息取加权平均来更新本节点的信息。目前这种方法占据主流，相关工作占比 80% 以上。本文所使用的残差门控图卷积神经网络也使用基于空间的方法。

基于空间的图卷积的过程可以大致分为以下三步。第一步，发送: 每个节点把自身的节点信息发送给其一阶邻居。第二步，接收与聚合：每个节点接收邻节点的节点信息，按照一定的权重对所有邻居信息取平均并对结果进行归一化处理。第三步，变换：将聚集之后的信息使用 Relu 等激活函数进行非线性变换，使模型表达能力更强。用 h_i^l 表示节点 i 第 l 层的

[2]图源：https://mlnotebook.github.io/post/CNN1
[3]图源：https://www.jianshu.com/p/2fd5a2454781

信息，N_i 表示节点 i 的邻域（所有邻居包括自身），w_j^l 表示节点 j 的权重，n_{ij} 表示归一化因子，Relu 表示线性整流函数，则节点信息的更新可用下式表示：

$$h_i^{l+1} = \mathrm{ReLU}\left(\sum_{j\in N_i} \frac{1}{n_{ij}} h_j^l w_j^l\right)$$

从上式中不难发现，感受域大小正比于层数。计算第一层时，每个节点使用直接邻居的信息更新自身；计算第二层时就能把二阶邻居的信息包含进来，参与运算的信息就更为丰富。图卷积层数越多，节点的感受域就更广，参与运算的信息就更多。

2.2 残差门控图卷积神经网络模型

具体到 TSP 问题这一任务，本文使用了 Bresson et al. (2017) 提出的残差门控图卷积神经网络模型。Sainbayar et al. (2016) 提出的 vanilla 图卷积神经网络的信息更新如下式所示 ($j \to i$ 表示节点 j 为 i 的直接邻居且 $j \neq i$，W_1, W_2 为参数，下同)：

$$h_i^{l+1} = \mathrm{ReLU}\left(W_1^l h_i^l + W_2^l \sum_{j\to i} h_j^l\right)$$

Marcheggiani et al. (2017) 介绍了一种边门控单元 η_{ij}，在使用邻居节点 j 的信息更新 i 时，将边 e_{ij} 也考虑在内：

$$h_i^{l+1} = \mathrm{ReLU}\left(\sum_{j\in N_i} \eta_{ij} \odot W^l h_j^l\right)$$

其中

$$\eta_{ij} = \sigma\left(A^l h_i^l + B^l h_j^l\right)$$

σ 表示 sigmoid 函数，A, B 为参数。综合考虑二者的优点，可以得到以下门控图卷积神经网络的模型：

$$h_i^{l+1} = \mathrm{ReLU}\left(W_1^l h_i^l + \sum_{j\to i} \eta_{ij} \odot W_2^l h_j^l\right)$$

相对于使用多层神经网络来拟合一个隐藏的非线性映射，让神经网络来学习该映射的残差会更容易一些，训练残差比训练原始函数更容易。引入残差机制，可以降低深度神经网络的训练难度。更重要的是，由于图卷积本质上在推动相邻节点的信息彼此接近，理论上经过无数图卷积层信息传播之后所有节点的信息将收敛到同一个值，即出现过度平滑。深层卷积神经网络的退化问题——梯度发散与此类似，卷积神经网络层数越深，梯度越容易发散，深度模型越难以优化并收敛到较优的解。而 ResNet 的残差机制 (即加入 shortcut) 便较好地解决了深度模型的梯度发散问题。受此启发，为解决堆叠多层图神经网络进行多次图卷积导致的信息过度平滑，一些工作在图神经网络中使用了残差连接。Bresson et al. (2017) 便在门控图卷积神经网络基础上引入残差机制，提出了残差门控图卷积神经网络模型：

$$h_i^{l+1} = h_i^l + \mathrm{ReLU}\left(W_1^l h_i^l + \sum_{j\to i} \eta_{ij} \odot W_2^l h_j^l\right)$$

在解决 TSP 问题的实际操作中，我们需要 GCN 用邻接概率矩阵的形式给出每条边出现在 TSP 路径中的可能性预测。这就需要使用图的边信息，使其参与信息传播 (Joshi et al. 2019)。因此实际使用的残差门控图卷积神经网络输入层中需要对节点和边都进行编码。

输入层将节点坐标编码成 k 维的特征向量。记输入的节点坐标矩阵为 $X = \{x_1, \cdots, x_n\}, x_i \in \mathbb{R}^2, i \in 1, \cdots, n$，系数矩阵 $A_1 \in \mathbb{R}^{k\times 2}$，则节点信息的编码可表示为：

$$v_i^0 = A_1 x_i + B_1$$

编码边信息时，将边的长度 d_{ij} 和邻接情况 δ_{ij} 分别进行编码，并拼接为 k 维边特征向量。当节点 ij 互为邻居时 $\delta_{ij} = 1$，否则 $\delta_{ij} = 0$。系数矩阵 $A_2, A_3 \in \mathbb{R}^{\frac{k}{2}}$。输入层编码的边信息为：

$$e_{ij}^0 = A_2 d_{ij} + B_2 || A_3 \delta_{ij} + B_3$$

图卷积层使用基于空间的方法更新每个节点的信息：

$$v_i^{l+1} = v_i^l + \mathrm{ReLU}\left(W_1^l v_i^l + \sum_{j\to i} \eta_{ij}^l \odot W_2^l v_j^l\right)$$

其中边门控单元

$$\eta_{ij}^l = \frac{\sigma\left(e_{ij}^l\right)}{\sum_{j\to i} \sigma\left(e_{ij}^l\right)}$$

边信息使用上一层的对应边信息和两个端点的节点信息更新：

$$e_{ij}^{l+1} = e_{ij}^l + \mathrm{ReLU}\left(W_3^l e_{ij}^l + W_4^l v_i^l + W_5^l v_j^l\right)$$

最后一个图卷积层第 l_{max} 层输出的边信息特征向量使用多层感知机（MLP）来进行解码，将其映射到区间 [0, 1] 上，表示对应边出现在 TSP 路径中的概率：

$$p_{ij} = MLP(e_{ij}^{l_{max}})$$

MLP 的输出即为我们所需的邻接概率矩阵。

值得注意的是，GCN 通常不会堆叠过多的层数，在处理节点分类等任务时仅使用 2 3 层图卷积层时结果最好，随着 GCN 层数的增加结果反而会变差。直观上讲，GCN 的最佳图卷积层数是和所处理图相关的，图的邻接矩阵越稀疏，所需的图卷积层越少，越容易因为堆叠过多图卷积层出现过度平滑现象。因为对于稀疏图，图中连接较少，每个节点的邻居也较少，能够形成连通分量的很少几个节点在多次图卷积中很容易因为节点信息的加权平均收敛到相同点。然而，在处理 TSP 问题时，对于 TSP20 等规模较小的问题，我们一般将输入作为全连接图处理。对于 TSP50，TSP100 等规模较大的问题，我们使用 KNN 的思想，视每个节点和其前 K 个近邻是互相连通的，因为多数情况下最优 TSP 路径会优先连接较近的点。可见，对于 TSP 问题而言，图是较为稠密的，因此堆叠图卷积层并不会很快造成过度平滑。同时，为了能找出更佳的 TSP 路径，我们需要保证节点的感受域足够大，能够感知到足够多节点的信息，因此需要堆叠较多的图卷积层。5.1的实验也验证了这一想法。当图卷积层数较少时，由于模型简单，参数也较少，训练过程需要的样本数量也较少，较为节省训练时间。然而参数较少的简单图卷积神经网络模型表达能力也弱于深层图卷积神经网络，就最终结果而言深层图卷积神经网络表现更好，更接近求解器给出的最优解。

2.3 本章小结

本章对图卷积神经网络的概念、类别及用途进行了简要介绍，并进一步重点介绍了所使用的残差门控图卷积神经网络模型，对该网络处理 TSP 问题时的信息编码与信息传播更新方式进行了分析。最后对处理 TSP 问题时 GCN 需要加深层数的原因和可行性进行了讨论。

第三章　传统算法

图卷积神经网络的输出是一个邻接概率矩阵，其每个元素 p_{ij} 表示 GCN 预测的对应边 e_{ij} 在 TSP 路径中出现的概率，而合法的 TSP 路径矩阵 V 中的元素应为非 0 即 1 的离散变量。再者，GCN 对于各边出现在 TSP 路径中的概率估计是互相独立的，而实际寻找 TSP 路径过程中下一条边的寻找与已有部分路径相关，GCN 不能承担寻找最终路径的工作。所以，我们需要合适的解码算法承担整数化工作，将该邻接概率矩阵转化为符合 TSP 条件的合法 TSP 路径。事实上，大量传统组合优化算法均可与 GCN 结合，利用 GCN 给出的边概率信息找出较优的 TSP 路径。在实验中我们采用了 Dang et al. (2002) 提出的一种迭代算法以及贪婪搜索、集束搜索两种启发式搜索算法。

3.1　迭代算法

Dang et al. (2002) 提出了一种用于解决 TSP 问题的基于拉格朗日乘子和障碍函数的迭代算法。首先，该算法通过引入一个松弛变量 $\rho > 0$，将 TSP 问题的约束条件改写为以下形式：

$$\begin{aligned} \min \quad & e_0(v) = \sum_{i=1}^{n} \sum_{j=1}^{n} \left(\sum_{k=1}^{n} d_{ij} v_{ik} v_{j,k+1} - \frac{1}{2} \rho v_{ij}^2 \right) \\ \text{subject to} \quad & \sum_{j=1}^{n} v_{ij} = 1, \quad i = 1, 2, \cdots, n \\ & \sum_{i=1}^{n} v_{ij} = 1, \quad j = 1, 2, \cdots, n \\ & 0 \le v_{ij} \le 1, \quad i = 1, 2, \cdots, n, \quad j = 1, 2, \cdots, n \end{aligned} \tag{3–1}$$

显然当松弛变量 ρ 足够大时，上式的最优解一定是整数解。ρ 越大，在后续的迭代中该算法就能越快收敛到一个整数解，但解的质量也随之下降，可能会给出更差的 TSP 路径。因此实验中需要对计算开销和解的质量进行折中。引入 Xu(1995) 提出的障碍函数项：

$$d\left(v_{ij}\right) = v_{ij} \ln v_{ij} + \left(1 - v_{ij}\right) \ln \left(1 - v_{ij}\right)$$

该问题可以进一步扩充为：

$$\begin{aligned} \min \quad & e(v;\beta) = e_0(v) + \beta \sum_{i=1}^{n} \sum_{j=1}^{n} d\left(v_{ij}\right) \\ \text{subject to} \quad & \sum_{j=1}^{n} v_{ij} = 1, \quad i = 1, 2, \cdots, n \\ & \sum_{i=1}^{n} v_{ij} = 1, \quad j = 1, 2, \cdots, n \end{aligned} \tag{3–2}$$

其中 $\beta > 0$ 是障碍函数项的参数，当 $\beta \to 0$ 时上式的解即离散变量松弛化后 TSP 问题的解。使用拉格朗日乘子将约束条件合并到目标函数中：

$$L\left(v, \lambda^r, \lambda^c\right) = e(v;\beta) + \sum_{i=1}^{n} \lambda_i^r \left(\sum_{j=1}^{n} v_{ij} - 1 \right) + \sum_{j=1}^{n} \lambda_j^c \left(\sum_{i=1}^{n} v_{ij} - 1 \right)$$

考虑 $L\left(v, \lambda^r, \lambda^c\right)$ 取到极值时的必要条件，我们有：

$$\frac{\partial L\left(v, \lambda^r, \lambda^c\right)}{\partial v_{ij}} = \frac{\partial e_0(v)}{\partial v_{ij}} + \lambda_i^r + \lambda_j^c + \beta \ln \frac{v_{ij}}{1 - v_{ij}} = 0 \quad ,$$

$$v_{ij} = \frac{1}{1 + \exp\left(\left(\frac{\partial e_i(v)}{\partial v_{ij}} + \lambda_i^r + \lambda_j^c \right) / \beta \right)}$$

记 $r_i = \exp\left(\frac{\lambda_j'}{\beta}\right), c_j = \exp\left(\frac{\lambda_j^e}{\beta}\right), \alpha_{ij}(v) = \exp\left(\frac{\partial e_0(v)}{\partial v_i}/\beta\right)$，那么有 $v_{ij} = \frac{1}{1+r_i c_j \alpha_{ij}(v)}$。考虑到 TSP 问题约束条件 $\sum_{j=1}^n v_{ij} = 1, i = 1, 2, \cdots, n$且 $\sum_{i=1}^n v_{ij} = 1, j = 1, 2, \cdots, n$，可以得到

$$\begin{aligned} \sum_{j=1}^n \frac{1}{1+r_i c_j \alpha_{ij}(v)} = 1, \quad i = 1, 2, \cdots, n \\ \sum_{i=1}^n \frac{1}{1+r_i c_j \alpha_{ij}(v)} = 1, \quad j = 1, 2, \cdots, n \end{aligned} \tag{3–3}$$

固定 v，则可根据上式解出 r 和 c；固定 r 和 c 则可根据 $v_{ij} = \frac{1}{1+r_i c_j \alpha_{ij}(v)}$ 解出 v。令

$$\begin{aligned} h_{ij}(v, r, c) = & \quad \frac{1}{1+r_i c_j \alpha_{ij}(v)} \\ h(v, r, c) = & \quad \left(h_{11}(v, r, c), h_{12}(v, r, c), \cdots, h_{1n}(v, r, c) \cdots, h_{n1}(v, r, c), h_{n2}(v, r, c), \cdots, h_{nn}(v, r, c)\right)^\top \end{aligned}$$

可以证明，$h(v, r, c) - v$ 是 $L\left(v, \lambda^r, \lambda^c\right)$ 的梯度下降方向。

对于给定的 v，要确保 $h(v, r, c) - v$ 是式 4-2 可行的下降方向，需要求出式 4-3 值为正的解 $r(v), c(v)$。考虑函数

$$f(r, c) = \frac{1}{2}\left(\sum_{i=1}^n \left(\sum_{j=1}^n \frac{1}{1 + r_i c_j \alpha_{ij}(v)} - 1\right)^2 + \sum_{j=1}^n \left(\sum_{i=1}^n \frac{1}{1 + r_i c_j \alpha_{jj}(v)} - 1\right)^2\right)$$

对于 $i, j = 1, 2, \cdots, n$, 令

$$\begin{aligned} x_i(r, c) = r_i \left(\sum_{j=1}^n \frac{1}{1+r_i c_j \alpha_{ij}(v)} - 1\right) \\ y_j(r, c) = c_j \left(\sum_{i=1}^n \frac{1}{1+r_i c_j \alpha_{ij}(v)} - 1\right) \end{aligned}$$

通过以下方式对 r, c 进行迭代：

$$\begin{aligned} r^{k+1} = r^k + \mu_k x\left(r^k, c^k\right) \\ c^{k+1} = c^k + \mu_k y\left(r^k, c^k\right) \end{aligned} \tag{3–4}$$

其中 μ_k 是 $[0, 1]$ 之间的实数，且满足

$$f\left(r^{k+1}, c^{k+1}\right) = \min_{\mu \in [0,1]} f\left(r^k + \mu_k x\left(r^k, c^k\right), c^k + \mu_k y\left(r^k, c^k\right)\right)$$

在实际操作中，取 $\mu_k = 0.95$ 且 $\sqrt{f\left(r^{k+1}, c^{k+1}\right)} < 0.001$ 时立即终止迭代即可保证给出正值解 r, c。

该迭代算法事实上与模拟退火算法类似，β 可视为退火温度。记初始退火温度为 β_0，终止温度为 β'，温度衰减系数为 η，判断向量 v 是否进行方向更新的门限为 ϵ，并将 v 随机初始化为元素取值在 $[0, 1]$ 之间的正向量 v_0，则该算法流程如算法3–1所示。该算法可以分为两个阶段，第一阶段找出松弛化后 TSP 问题（式 4-1）的解（行 1-16），第二阶段对第一阶段的解进行整数化：

$$v_{ij}' = \begin{cases} 1 & \text{当} v_{ij}^k \geq 0.9 \\ 0 & \text{当} v_{ij}^k < 0.9 \end{cases}$$

然后判断 v_k' 是否为 TSP 问题的合法解，若是则算法结束，否则增大松弛变量 ρ 并回到算法第一阶段, 使用当前结果 v_k' 初始化 v_k 进行下一轮迭代。

算法执行过程中，随着松弛变量 ρ 不断增大，最终寻找到的整数解的精确度将由于条件放宽而不断下降，但该算法也更容易收敛到整数解。松弛变量 ρ 足够大时，式 4-1 的最优解一定是整数解，因此该算法最终必定收敛到符合 TSP 问题约束条件的整数解。

该算法优点是不需要任何额外信息就可直接找出 TSP 路径，因此可与 GCN 结合进行无监督学习，训练样本不需要提供最优解。而缺点是计算较繁，且受初始化影响严重，较差的初始化可能 ρ 很大时算法仍未收敛到整数解，导致最终给出的解误差较大。此外，使用类似的梯度下降思路来寻找最优 TSP 路径，局部最优问题的难以避免的，难以保证 $L\left(v, \lambda^r, \lambda^c\right)$ 一直是全局凸函数。

算法 3–1 迭代算法 Dang et al. (2002)

Input: 输入图各个点的坐标
Output: TSP 路径

初始化 $k = 0$;
while $\beta_k \geq \beta'$ **do**
  根据 v_k，使用式 4-4 不断迭代，计算出式 4-3 的正值解 r_k, c_k;
  使用 v_k, r_k, c_k 计算出 $h(v_k, r_k, c_k)$;
  if $h(v_k, r_k, c_k) - v_k < \epsilon$ **then**
    if $\beta_k < \beta'$ **then**
      break;
    else
      $\beta_{k+1} = \eta\beta_k$;
      $v_{k+1} = v_k$ $k = k + 1$;
    end
  else
    $v^{k+1} = v^k + \theta_k \left(h\left(v^k, r\left(v^k\right), c\left(v^k\right)\right) - v^k\right)$, 其中 $\theta_k \in [0, 1]$;
    $k = k + 1$;
  end
end
令 $\beta_k = \beta'$;
对上一阶段给出的 v 进行整数化：若 $v_{ij} > 0.9$，则 $v'_{ij} = 1$，否则 $v'_{ij} = 0$;
if v' 满足 *TSP* 问题约束条件 **then**
  输出 v';
else
  $\rho = \rho + 2$;
  $k = k + 1$;
  $v_k = v'$ 回到算法第 2 行;
end

3.2 启发式搜索算法

将 GCN 给出的邻接概率矩阵映射到 TSP 路径与序列到序列 (seq2seq) 模型的解码十分类似，而启发式搜索算法常用于 seq2seq 模型的解码阶段。同时，GCN 给出的邻接概率矩阵可以作为很强的启发信息来引导搜索的进行，所以使用启发式搜索算法进行这一映射是一种可行的思路。启发式搜索算法可以利用问题已有的启发信息来引导搜索向最有希望产生最优解的方向前进，从而减小搜索范围并降低问题复杂度。通过在搜索过程中进行剪枝，删除某些状态及其延伸，启发式搜索可以避免解空间较大时的组合爆炸现象，得到较好的合法解 (不一定是最优解)。贪婪搜索和集束搜索是启发式搜索的常用方法。

3.2.1 贪婪搜索

贪婪搜索的思想十分简洁明了。利用已有的启发信息，从起始状态开始，下一步选取该节点所有未遍历的邻接点中概率最大的点，然后将出发点设为已遍历。将这一步选中的节点作为新的起始点，继续进行上一步，直至遍历完所有节点。

贪婪搜索的优点是易于实现，计算代价非常低，但是其缺点也十分明显，即容易困在较差的局部最优情况，如图3–1所示。从起始点 1 出发，遍历到节点 11 时，贪婪搜索会选中距 11 最近的 12 号节点，即局部最优。节点 21 成为遍历至节点 20 时最后一个未被遍历的节点，要花费较大的开销从 20 到 21 再从 21 回到 1。因此，贪婪搜索通常不能给出很好的解。

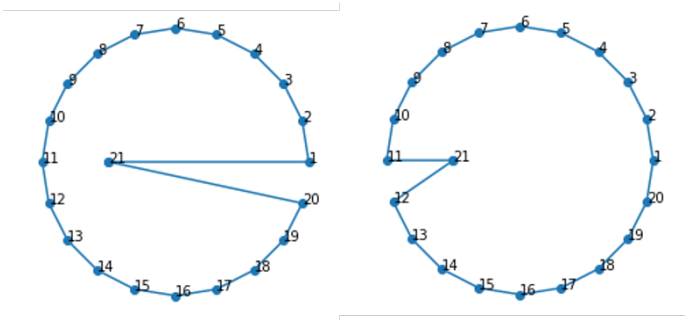

图 3–1 以节点 1 为起点，左侧为贪婪搜索给出的 TSP 路径，右侧为最优 TSP 路径。

用于和 GCN 结合寻找 TSP 路径的贪婪搜索算法可修改为3–2的形式。

算法 3–2 用于 TSP 问题的贪婪搜索算法

Input: GCN 输出的邻接概率矩阵，输入图 G
Output: TSP 路径

起始节点（1 号节点）加入路径;
while 路径经过的节点数 < 图 G 节点数 n **do**
    遮罩路径中所有节点;
    根据 GCN 给出的邻接概率矩阵对路径末尾节点和其余未被遮罩节点之间的边出现在 TSP 路径中的概率进行排序;
    选取概率最大的边对应的未在路径中的节点加入路径;
end
将产生的无环路径首尾相连进行闭合作为输出的 TSP 路径;

3.2.2 集束搜索

集束搜索 Medress et al. (1977) 是一种启发式图搜索算法，可被视为贪婪搜索和广度优先搜索的折中。为了减少搜索所占用的空间和时间，在每一次扩展深度时，依据启发信息，删掉排名靠后的状态，仅保留下当前部分解较优的状态。这样不仅节约了储存空间，也降低了时间开销，但缺点是潜在最优解可能提前被剪枝。集束搜索不会对解空间进行完全遍历，一般用于解空间较大的系统中，对解的质量和计算代价进行折中，给出可以接受的解（往往并非最优解）。集束搜索仅有一个参数即集束宽度 b。如果集束宽度趋于无穷，那么集束搜索就成为广度优先搜索，如果集束宽度为 1，那么集束搜索就变为贪婪搜索。

集束搜索使用广度优先策略建立搜索树。在树的每一层，对当前的 b 种状态进行广度优先扩展，对于每种已有状态列举其所有可能转移到的 n 种状态，然后使用启发信息评价函

数对扩展后的 $b \times n$ 状态进行排序，仅留保前 b 种状态。保留的状态在下一层继续扩展，其他状态被剪枝。

在应用于结合 GCN 给出的邻接概率信息解决 TSP 问题时，集束搜索可以修改为算法3–3的形式。

算法 3–3 用于 TSP 问题的集束搜索算法

Input: GCN 输出的邻接概率矩阵，输入图 G，集束宽度 b

Output: TSP 路径

起始节点（1 号节点）入队;
while 队列中第一条路径经过的节点数 < 图 G 节点数 n **do**
    队头前 b 条路径出队;
    按广度优先策略从出队的 b 条路径的末端节点扩展每条路径。每条路径分别维护已访问节点的信息避免产生回路，产生不超过 $b \times n$ 条无环待选路径;
    根据 GCN 给出的邻接概率矩阵计算每条待选路径出现的概率，从中选取可能性最大的前 b 条路径入队;
end
将队列中的 b 条无环路径各自首尾相连进行闭合;
根据图 G 的坐标信息从 b 条环路中选出长度最短的一条路径作为最终的 TSP 路径;

3.3 本章小结

GCN 的输出是邻接概率矩阵，其对于各边出现概率的预测是互相独立的，因而并不能直接给出 TSP 路径，需要使用传统算法利用 GCN 的输出找到合法的 TSP 路径。本章举例介绍了三种可以与双向深度学习系统结合的传统算法，即迭代算法 (Dang et al. 2002)、贪婪搜索与集束搜索 (Medress et al. 1977)，并对其各自的优劣势进行了分析。

第四章　双向深度学习系统

4.1　双向深度学习系统结构

Xu (2019a) 描绘了一种双向深度学习策略，解决具有双随机矩阵（double stochastic matrix，即满足行和列加和均为 1 的非负矩阵）特征的组合优化问题，最典型的此类问题即 TSP 问题。该双向深度学习策略一方面考虑对输入拓扑图进行特征丰富化，将其转化为一种类似棋盘格的图形 (Xu 2019b)，以便使用 CNN 学习其特征，将输入图映射到松弛化后 TSP 问题（式 4-1）的解，另一方面使用传统算法将松弛化后的解转化为 TSP 路径，计算 Loss 更新 CNN 参数。

随着 GCN 近几年的发展，使用 GCN 来直接学习图的拓扑信息成为一种很好的选择，无需再对图进行编码然后使用传统 CNN 学习。GCN 提取出的图特征可以作为一种启发信息，有效地指导传统算法的优化方向，节省其计算开销，提高最终结果的准确度。同时传统算法找出的 TSP 问题的解也可以作为 GCN 训练时的标签，协助 GCN 的训练过程。二者可以在训练过程中互相更新，互相提高。目前有关 GCN 解决 TSP 问题的研究中，几乎都是有监督地训练 GCN，然后将 GCN 输出解码为 TSP 路径的方式，尚无使用双向深度学习来解决 TSP 问题的实例。于是，参考 Xu (2019a) 提出的利用双向深度学习解决 TSP 问题的构想，我们提出一种双向深度学习系统。该双向深度学习系统一方面使用 GCN 自下而上抽取输入拓扑图的特征，将其转化为有效的启发信息指导传统算法进行自上而下的 TSP 优化，另一方面将优化后的 TSP 路径传递给 GCN 更新其参数，再进行下一轮循环，直至前后两次循环的最终结果差距小于预设的门限。该深度学习系统的结构可以形象地表示为下图：

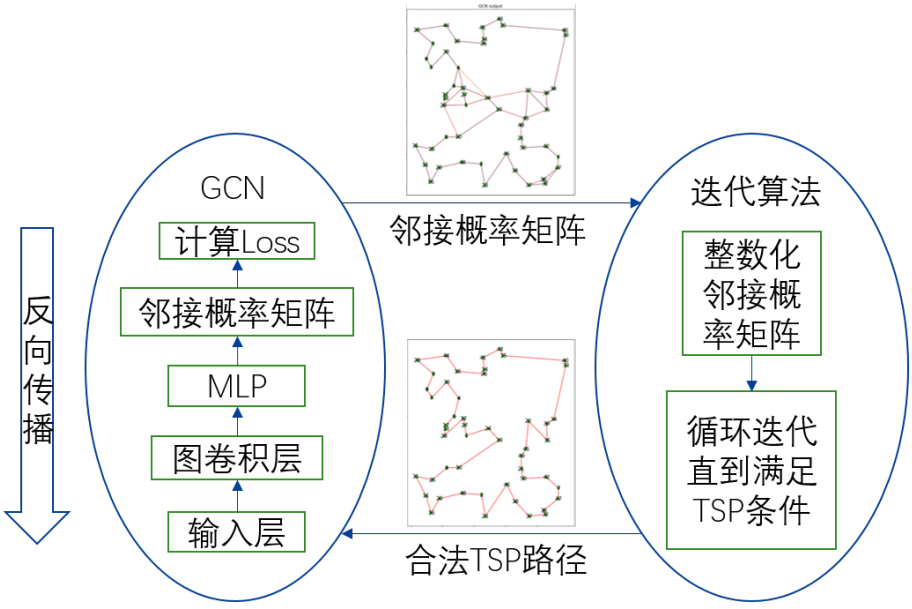

图 4–1 双向深度学习系统，以迭代算法结合 GCN 为例。GCN 自下而上抽取输入图 $\mathcal{G}$ 的特征转化为启发信息指导迭代算法自上而下的 TSP 优化，优化后的路径再反馈给 GCN 做反向传播，开始下一轮循环。

具体而言，我们一方面使用 GCN 学习输入图的拓扑信息，完成以下映射（θ 代表 GCN

的参数)：

$$f(\mathcal{G},\theta)=P, P=\left[p_{ij}\right], i,j=1,\cdots,n$$
$$\text{subject to } 0\le p_{ij}\le 1, \sum_j p_{ij}=1, \sum_i p_{ij}=1.$$

另一方面传统算法 S 接收矩阵 P 进行整数化，计算出 TSP 路径 v，并回传给 GCN:

$$v=S(\mathcal{G},P)$$

GCN 需要通过梯度下降最小化 Loss 来更新参数 θ。首先考虑交叉熵 Loss:

$$\min_\theta l\left(f\left(\mathcal{G},\theta\right),v\right)=-\sum_i^N\sum_j^N v_{ij}\log\left(f_{ij}\left(\mathcal{G},\theta\right)\right)+\left(1-v_{ij}\right)\log\left(1-f_{ij}\left(\mathcal{G},\theta\right)\right)$$

考虑到当问题规模逐渐增大时，TSP 问题最终的解路径矩阵是稀疏的，其绝大部分元素都是 0，即绝大部分边都未被 TSP 路径选中。因为解路径矩阵中共有 $n\times n$ 个非 0 即 1 的离散元素，而仅有 n 个取 1，其他 n^2-n 个全为 0。可见，一条路径最终被预测为未选中，即对应项 $e_{ij}=0$ 的概率要远大于 $e_{ij}=1$ 的概率。这样，上面的交叉熵 Loss 中，因子 $(1-v_{ij})\log\left(1-f_{ij}\left(\mathcal{G},\theta\right)\right)$，即未选中边的影响要远大于因子 $v_{ij}\log\left(f_{ij}\left(\mathcal{G},\theta\right)\right)$，即 TSP 路径选中的边的影响。这将会严重影响 GCN 对 TSP 路径的学习。因此，要权衡一条边被划分到选中和未选中两类在最终 Loss 函数中的影响程度。为使 GCN 更好地学习最终找出的 TSP 路径信息，参考 Joshi et al. (2019) 提出的两种划分为类别的权重因子：

$$w_0=\frac{n}{2n-4}, w_1=\frac{n}{4}$$

w_0, w_1 分别表示未被选中边和被选中边的权重。进而我们可以将损失函数改写为

$$\min_\theta l\left(f\left(\mathcal{G},\theta\right),v\right)=-\sum_i^N\sum_j^N w_1 v_{ij}\log\left(f_{ij}\left(\mathcal{G},\theta\right)\right)+w_0(1-v_{ij})\log\left(1-f_{ij}\left(\mathcal{G},\theta\right)\right)$$

整体来看，GCN 所需的输入包括图 $\mathcal{G}$，以及用于计算交叉熵 Loss 的一条 TSP 路径 v。因此完整的 GCN 可以用映射 $f_{GCN}\left(\mathcal{G},v,\theta\right)$ 来表示。于是整个双向学习系统可以简要地表示为：

$$f_{GCN}\left(\mathcal{G},v,\theta\right)\underset{v}{\overset{P}{\rightleftharpoons}}S(\mathcal{G},P)$$

4.2 实现细节

在实际训练过程中，GCN 和传统算法都需要进行一些调整以实现互相结合，方便实际操作。首先考虑 GCN 的调整。对输入层而言，由于训练样本由 DIMACS TSP challenge 生成，并且数据分布类型有两种，Random 类数据是随机生成的点，较为均匀地分布在 $[0,1000000]^2$ 区间内，而 Clustered 类数据较前一种更为集中和密集，大致分布在 $[-300000,300000]^2$ 区间内，具体表现在内部节点更多，最终的 TSP 路径“凹陷”或“褶皱”更多。训练时我们将两种数据各 50000 组混合在一起，每个 epoch 随机抽取 100 组数据划分为 10 个 mini-batch 进行训练。然而，我们的 GCN 模型并不具备迁移学习能力，其训练和测试立足于机器学习领域的 IID 假设，至少训练数据和测试数据在图片大小（即节点坐标分布区间大小）上需保持一致，否则 GCN 模型难以稳定地学习规律。考虑到 GCN 对输入图片大小和分布区间的敏感性，加之图片大小变化以及整体的平移旋转并不改变其拓扑性质，TSP 路径长度也会进行等比缩放，为降低输入的协方差偏移 (Covariate shift)，使 GCN 加快训练速度，提高训练质

量，我们在输入层对输入图进行类似于白化 (whiten) 的等比缩放操作，即对输入样本进行归一化，使之分布于 $[0,1]^2$ 区间内。需要注意的是，我们对整张图进行长宽等比缩放，这并没有改变 Random 和 Clustered 两种分布类型各自的性质，缩放后的数据仍是两类分布，只不过大小统一在一个规模，方便 GCN 进行学习。

此外，由于我们实际使用的 GCN 层数较深，需要考虑到内部协方差偏移 (Internal Covariate Shift) 问题。对于包含很多隐含层的深层神经网络结构，由于训练时各隐含层的参数都在不断发生变化，每个隐含层都会面临和输入层类似的协方差偏移问题，即其激活函数 ReLU 的输入分布可能不断发生变化。激活函数输入的分布会随着神经网络的加深，逐渐向激活函数取值范围的上下限两端靠近，这可能导致反向传播时低层神经网络的梯度消失。这也是深层神经网络的训练过程收敛速度慢的根本原因。批归一化 (Batch Normalization) 是解决这个问题的较好途径。对于每个隐层神经元，批归一化把逐渐向非线性激活函数的取值区间上下极限靠拢的输入分布通过改变方差大小和均值位置强行拉回到比较标准的正态分布，使得非线性激活函数的输入落入该函数对输入较敏感的区域，以避免梯度弥散。因此，为保证 GCN 模型的非线性表达能力，降低内部协方差偏移，使梯度更新方向更准确，加快训练速度，参考 Joshi et al. (2019)，我们对每个图卷积层在非线性变换前进行批归一化 (BN) 操作。从而，每个图卷积层输出的节点信息可表示为：

$$v_i^{l+1} = v_i^l + \mathrm{ReLU}\left(\mathrm{BN}\left(W_1^l v_i^l + \sum_{j\to i} \eta_{ij}^l \odot W_2^l v_j^l\right)\right), \quad \eta_{ij}^l = \frac{\sigma\left(e_{ij}^l\right)}{\sum_{j\to i}\sigma\left(e_{ij}^l\right)}$$

每个图卷积层的输出边信息：

$$e_{ij}^{l+1} = e_{ij}^l + \mathrm{ReLU}(\mathrm{BN}\left(\left(W_3^l e_{ij}^l + W_4^l v_i^l + W_5^l v_j^l\right)\right)$$

除了 GCN 的调整外，传统算法也需要一定的调整。以 GCN+ 迭代算法3–1为例，双向深度学习系统的无监督训练方式如图4–1所示。对于每一个输入样例，GCN 将预测结果以邻接概率矩阵的形式传递给迭代算法3.1来指导其进行优化。该算法将邻接概率矩阵作为松弛化后 TSP 问题（式 4-1）的解，直接进入第二阶段（从算法3–1第 17 行开始），不断迭代直到给出合法 TSP 路径，将其回传给 GCN。GCN 根据 TSP 问题解的信息计算交叉熵 Loss 进行反向传播，通过梯度下降最小化交叉熵 Loss 来更新参数，之后进行下一轮循环，直到连续两次循环的输出路径长度差距小于 1%。测试过程和验证过程则不进行反向传播完成循环，直接取整个系统第一次计算出的 TSP 路径作为结果。

此外，贪婪搜索和集束搜索也需要在与该双向深度学习系统结合时进行调整。两种算法的内在思想不变，但形式上与使用在 seq2seq 模型的解码阶段时并不相同。贪婪搜索和集束搜索用于结合 GCN 解决 TSP 问题的变体形式如算法3–2和算法3–3所示。值得注意的是，GCN 在训练阶段的首个 epoch，参数是随机初始化的，因此一开始并不能提供较为有效的启发信息。由于贪婪搜索本身并不是像迭代算法3–1一样用于解决 TSP 问题的算法，其本身不借助启发信息并不能给出一个较好的 TSP 路径，使用无监督训练会导致 GCN 在前几个 epoch 无法进行有效的初始化，即 GCN+ 贪婪搜索的双向深度学习系统自身无法“启动”。而集束搜索在处理 TSP20 等规模较小的问题时，以边长度倒数为启发信息就可以独立找出可以接受的 TSP 路径。而对于 TSP50，TSP100 这种规模的问题，集束搜索在集束宽度的大小并未导致计算开销和储存空间使用过大时，也不能独立找出较好的 TSP 路径（集束宽度过大则相当于穷举，这是没有意义的），如图4–2所示。

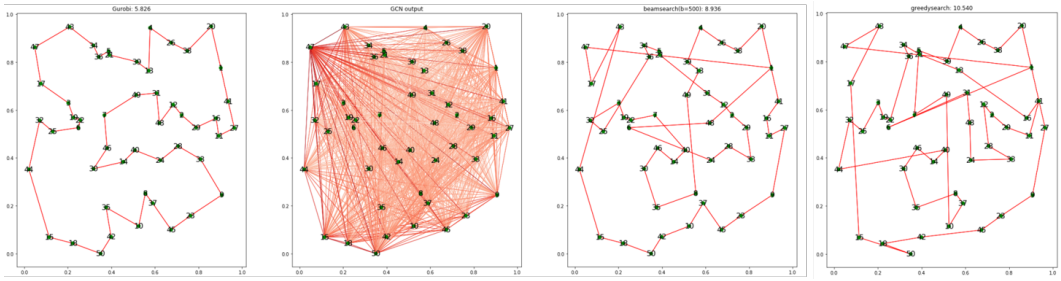

图 4–2 TSP50 问题的初始化阶段，集束搜索与贪婪搜索的表现。

从图中可见，此时 GCN 给出的邻接概率矩阵并不能作为有效的启发信息，因为其各层参数仅进行了随机初始化。无有效启发信息时集束搜索（集束宽度 b=500）给出的路径长度是 8.936，Gurobi 给出的最优路径长度为 5.826，相对误差超过 50%，贪婪搜索表现更差，其路径长度为 10.540，相对误差超过 90%，显然这样的结果不能指导 GCN 的反向传播，因此需要部分有标签数据用于 GCN 的初始化，进行半监督学习。对于贪婪搜索和集束搜索，实验中我们进行半监督训练，训练集中 30% 的数据附带有最优路径作为标签。训练过程中前 300 个 epoch 需要提供求解器 Gurobi 给出的最优 TSP 路径用作交叉熵 Loss 计算。

4.3 本章小结

本章对双向深度学习系统的结构和实现细节进行了介绍，即如何通过双向深度学习结合图卷积神经网络与传统迭代算法或启发式搜索算法，使二者可以互相更新，互相提高，达到无监督学习或半监督学习的目的。

第五章　实验与结果

5.1　超参数设置

在实验过程中，首先需要考虑 GCN 的超参数设置。其中最重要的是 GCN 网络的结构参数，即堆叠多少图卷积层，以及将节点信息和边信息编码为多长的特征向量。这些参数决定了 GCN 模型的学习能力与训练难度。Kipf et al. (2016) 的研究指出，GCN 的层数不宜多，一般使用 2~3 层。而且 GCN 在处理节点分类等问题时，一般仅需要两层，甚至不需要训练，仅凭随机初始化的参数就足以给出惊人的结果。因此在实验初期我们并未考虑加深图卷积神经网络。而后续实验表明处理 TSP 这类较为复杂、且输入图较为稠密的问题时 GCN 需要足够的层数来保证其学习能力和非线性表达能力。是否需要在处理 TSP 问题时加深网络以及加深到什么程度是一个值得考虑的问题。参考 Joshi et al. (2019) 的工作，我们设计了类似的实验来进行探究。在实验过程中我们使用了一些不同的图卷积层层数和特征向量维数的搭配组合对于 Random 数据分类的 TSP50 问题进行测试，仅使用该数据分类的 50000 个样例进行训练 GCN+ 集束搜索模型，使用测试集中的 100 个 Random 类样本计算相对误差，避免其他变量的影响。实验结果见表5–1：

图卷积层数	特征向量长度	相对误差
2	20	19.65%
5	50	12.10%
10	100	5.54%
20	200	2.63%
30	300	2.59%

表 5–1 GCN+ 集束搜索模型，不同 GCN 层数和特征向量长度在处理 TSP50 问题时的相对误差。评价基准由求解器 Gurobi 的结果充当。

从实验结果中可见，随着 GCN 层数的逐渐加深，对节点信息和边信息编码形成的特征向量长度逐渐增加，GCN 模型逐渐复杂化，但最终结果在不断变好。从两层图卷积层到 30 层，相对误差减小了超过 700%。然而从两层图卷积层到 20 层图卷积层，相对误差已经减小到了 2.63%，增加到 30 层后也只不过降低到了 2.59%，这样的表现提升程度比起训练难度的大幅增加而言是较低的。因此我们没有尝试继续加深网络。可见，一味地加深图卷积网络也是不可取的，超过一定限度之后结果并不会继续变好，甚至可能因为出现梯度消失问题，大幅增加训练难度，继续加深网络最终会对 GCN 模型的表现起到副作用。对于以上结果，我们认为，过浅的图卷积神经网络和过于简单的特征编码不足以支持其充分学习输入图的边信息和节点信息，并完成从输入图到邻接概率矩阵的映射。图卷积神经网络的层数足够深才能表示较为复杂的非线性映射，提取出的边信息节点信息的特征向量维数足够多才能提取出足够的结构信息，用于支持后续对每条边出现在 TSP 路径中概率的预测。至于堆叠较多 GCN 层可能导致信息平滑的问题，这是与输入图的性质相关联的。对于 TSP20 问题我们对于输入图按全联通处理，对于 TSP50 和 TSP100，我们视某节点与最近的 19 个邻居节点互相联通，因此输入图是较为稠密的，在一定限度内堆叠图卷积层不容易出现信息平滑。

而且，每个节点的感受域也将随着卷积层的加深而不断扩大，第一图卷积只能利用到一阶邻居的信息，第二次就能接收到二阶邻居的信息，依次类推。这更有利于在全局范围内寻找最佳 TSP 路径，因此我们也需要堆叠一定层数的图卷积层来保证每个节点的感受域足够大。总而言之，加深图卷积网络，其层数应有一个最优范围，层数不能过浅导致模型过于简单导致模型表达能力不足，层数也不能过深导致模型难以训练，同时也容易导致过拟合。

此外，加深图卷积神经网络自然会面临深度学习在加深神经网络时容易出现的致命问题，即梯度消失。于是在普通的图卷积神经网络基础上，我们使用了 Bresson et al. (2017) 提出的残差门控图卷积神经网络，并参考 Joshi et al. (2019) 的工作，对每个隐含层的激活函数输入进行批归一化。采用残差连接机制、做批归一化以及使用 ReLU 而非 Sigmoid 做激活函数都是为了应对图卷积神经网络层数加深时出现的梯度消失问题，使其更易于训练。实验中我们使用 30 层图卷积层，300 维特征向量的配置。

其次还需要考虑学习率的设置。参考 Joshi et al. (2019) 和 Nowak et al. (2017) 的工作，我们将初始学习率设置为 0.001。随后使用预留的 10000 组验证（validation）集数据，在训练过程中，每个 epoch 结束时随机抽取一个 batch 进行一次验证。如果此次验证集上的 Loss 与上次验证相比变化较小（小于 1%），我们便认为此时比原先更接近最优状态，然后缩小学习率，$lr_{new} = 0.99 \times lr$。使用初始较大而逐渐缩小的学习率，在训练刚开始的几个 epoch 可以快速地将 GCN 参数更新到一个较为合适的状态（远非最佳状态，但是已经可以给出一部分有效的启发信息，而不是像最初参数随机生成时几乎不能给出任何信息），此外也可以避免接近最优状态时的振荡，或者出现损失值爆炸等情况。

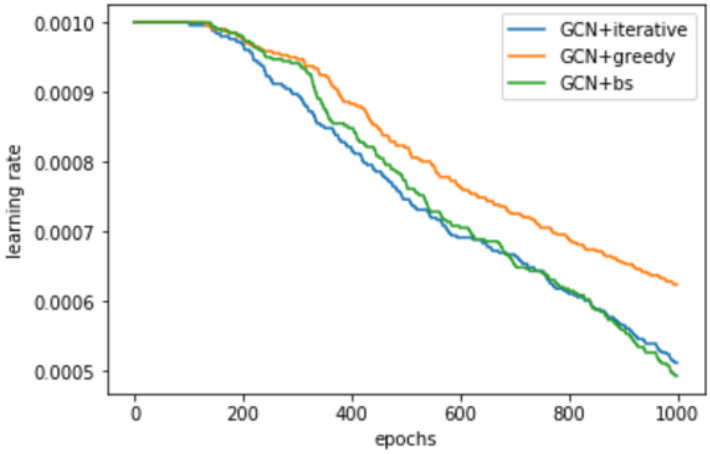

图 5–1 训练过程中三种方法的学习率变化曲线，以 TSP50 问题为例。

从上图中可见，在训练刚开始进行时 GCN+ 贪婪搜索和 GCN+ 集束搜索的学习率下降速度要低于 GCN+ 迭代算法。这是因为 GCN+ 迭代算法形成的双向深度学习系统是无监督学习，而 GCN+ 贪婪搜索和 GCN+ 集束搜索因为启发式搜索本身在没有有效启发信息的情况下，不能独立找出较好的 TSP 路径，需要额外的标签在该系统训练过程中的“启动”阶段提供参数更新的方向，即只能使用半监督学习。在训练过程中开始的 300 个 epoch 为了正确地对 GCN 的参数进行初始化使其能够给出较为有效的启发信息指导启发式搜索算法的 TSP 路径寻找，使用的数据是有标签的，即其包含有最优路径信息，用于交叉熵 Loss 的计算以及通过梯度下降减小 Loss 更新各层的权重参数。因此 GCN+ 贪婪搜索和 GCN+ 集束搜索的学习率一开始比 GCN+ 迭代算法更大，以快速进行学习。较大的学习率可以使模型更快

地收敛到稳定状态附近。GCN+ 迭代算法形成的双向深度学习系统，在“启动”阶段较另外两种方法更为稳定，因此最先出现收敛，学习率最先发生下降。在后续训练过程中，可以看出 GCN+ 集束搜索模型收敛较快，其学习率逐渐和 GCN+ 迭代算法互相接近。而 GCN+ 贪婪搜索的学习率一直较另外两种双向深度学习系统大，这是因为贪婪搜索给出的解的质量是难以保证的，有时局部最优和全局最优的差距并不大，甚至局部最优就是全局最优情况，而有时则被困在较差的全局最优情况，此时反馈给 GCN 的路径信息是不准确的。贪婪搜索难以保证的解的质量造成了 GCN+ 贪婪搜索在训练过程中的波动，GCN 的在更新参数时的梯度下降方向可能是不稳定、不准确的。因此，在相邻两次验证过程中，验证集上的 Loss 更可能出现波动，其学习率下降也就较慢，GCN 模型收敛也就慢。

此外，贪婪搜索过程对输入图原本的距离矩阵并没有充分利用，其搜寻 TSP 路径的过程是完全依赖 GCN 给出的启发信息，即邻接概率矩阵。而集束搜索最后从 b 条待选路径中选择最短的一条，这需要用到输入图的距离矩阵；迭代算法在寻找解向量的更新方向时也需要利用输入图的距离矩阵。GCN+ 贪婪搜索模型在每次循环的解码阶段对图的拓扑信息与节点位置的空间信息利用不足，这很可能也是该模型不够鲁棒、受较差的局部最优情况影响较大的原因之一。

接下来讨论传统算法的参数设置。迭代算法 (Dang et al. 2002) 比较重要的参数有松弛变量 ρ，该变量直接影响最终整数解的质量和求解时间。参考 Dang et al. (2002) 的实验参数设置，我们对 ρ 的取值在计算时间和解的质量之间进行折中，取 $\rho = 30$ 作为起始值，在后续的迭代中如果当前 ρ 值找不到整数解，就增大 ρ 值，令 $\rho = \rho + 2$。除此之外还需要设置初始退火温度 β_0，终止温度 β_f，温度衰减系数 η，结果更新门限 ϵ。同样参考 Dang et al. (2002) 的参数配置，我们令 $\beta_0 = 200$，$\beta_f = 1$，$\eta = 0.95$，$\epsilon = 0.01$。此外 Dang et al. (2002) 的工作指出在向量 r^k, c^k 的迭代计算阶段，为保证给出一个各维度取值均为正的向量解，只需将更新程度系数 $\mu_k \in [0, 1]$ 设为一个固定值 0.95，而不必通过式

$$f\left(r^{k+1}, c^{k+1}\right) = \min_{\mu \in [0,1]} f\left(r^k + \mu_k x\left(r^k, c^k\right), c^k + \mu_k y\left(r^k, c^k\right)\right)$$

来确定 μ_k 的值，以减小计算难度。

贪婪搜索不需要额外的参数，集束搜索仅需一个参数即集束宽度。对 TSP20，TSP50，TSP100 我们分别改变集束宽度在 Random 类数据的测试集上进行了测试，分别取集束宽度 $b = 1, 5, 10, 20, 50, 100, 200, 300, 400, 500, 600, 800, 1000, 1200, 1500$，不同集束宽度使用 GCN+ 集束搜索模型在 Random 类数据分布的测试集上相对误差如表5–2所示。

从表中可以看出，集束宽度从 1（此时集束搜索即贪婪搜索）刚开始增大时，对最终结果相对误差的影响程度十分巨大。集束宽度仅仅是从 1 增加到 100，TSP20 问题的相对误差骤然就降低了近 600%，对 TSP50 和 TSP100 而言相对误差也下降了超过 100%，可见集束搜索相比贪婪搜索的优越性。集束宽度超过 400 后，对三种问题最终结果的影响就开始快速降低，除了 TSP100 问题的相对误差还有部分影响，TSP20 和 TSP50 问题的相对误差都几乎不再有任何变化。可见，集束宽度对结果的影响程度也是有极限的，虽然理论上无穷大的集束宽度即广度优先搜索必然能找到最优路径，但穷举的时间空间开销目前的计算机都无法承担，这是得不偿失的。在集束宽度处于一个合理范围，不会造成过大计算开销的前提下，我们可以考虑尽可能取较大的集束宽度以换取较好的模型性能。

虽然集束宽度越大计算开销越大，但在搜索过程中提前剪掉潜在的最优路径对应的当前部分状态的可能性也会随着集束宽度的增大而减小。显然最佳的集束宽度和 TSP 问题的规模是相关的。但是考虑到集束宽度在合理范围内如 [1, 1500] 之内取较大的值带来的计算

TSP20		TSP50		TSP100	
集束宽度	相对误差	集束宽度	相对误差	集束宽度	相对误差
1	3.21%	1	9.62%	1	19.02%
5	2.73%	5	9.13%	5	18.39%
10	1.94%	10	8.77%	10	17.13%
20	1.53%	20	8.11%	20	15.98 %
50	0.84%	50	5.46%	50	13.68%
100	0.55%	100	3.27%	100	10.05%
200	0.21%	200	3.06%	200	9.23%
300	0.17%	300	2.82%	300	8.61%
400	0.15%	400	2.61%	400	8.15%
500	0.14%	500	2.60%	500	7.79%
600	0.14%	600	2.60%	600	7.57%
800	0.14%	800	2.55%	800	7.24%
1000	0.14%	1000	2.51%	1000	6.93%
1200	0.14%	1200	2.51%	1200	6.78%
1500	0.14%	1500	2.51%	1500	6.62%

表 5–2 GCN+ 集束搜索模型，不同集束宽度下对应的测试集上的相对误差变化情况。显然问题规模越大所需的最佳集束宽度也越大，适当增大集束宽度可以很好地提升该模型的总体表现。

开销并不算大，以上的几种集束宽度的训练过程消耗时长事实上差别不大，双向深度学习系统训练过程的主要耗时仍在 GCN 的正向传播和反向传播上。与穷举法如广度优先搜索相比，其在第 n 层扩展时的搜索宽度为 a^n 的指数级，总的时间复杂度是 $o\left(2^n\right)$，这种情况才会导致指数爆炸与计算时间储存空间消耗的激增，而集束搜索的每层的搜索宽度是固定不变的，扩展 n 层总的时间复杂度是 $o(n)$，所以略微增大集束宽度的时间开销可以忽略不计。在优先考虑结果精确度的前提下，我们考虑使用较大的集束宽度以降低提前丢弃潜在最优解的可能性，因此统一取 b=1200。

5.2 数据集

实验过程中使用的数据集由 TSP Challenge 通过指定不同的随机数种子生成。对于每种不同规模的问题（TSP20，TSP50，TSP100），Random 和 Clustered 两种分布类型各生成 50000 例作为训练集。由于该生成器产生的数据分布差异较大，不利于 GCN 学习，我们进一步对生成的数据进行等比缩放和平移（详见章节4.2），不改变图的性质。训练时同一规模的问题两种分布类型的数据混合在一起进行训练，本质上这是相同的 TSP 问题，只不过 Clustered 类略难。考虑到计算开销，训练集中仅 30% 计算出了最优路径，且训练集数据中的最优路径信息仅提供给 GCN+ 启发式搜索模型做半监督训练时的标签。验证集每种问题规模、每个数据分布各生成 500 例，每种规模的问题每个 epoch 结束后的验证阶段随机从该规模的 1000 个验证集数据中抽取 10 个作为一个 batch 进行验证。验证集数据均需要带有标签以计

算验证集 Loss。测试集每种问题规模、每个数据分布仅生成 100 例，且测试时对不同的数据分布类型分开进行测试，得出不同问题规模不同数据分布类型下的相对误差。验证集和测试集数据的平均最优路径长度、标准差的统计数据如下表所示：

数据分布类型	问题规模	验证集		测试集	
		路径长度	标准差	路径长度	标准差
Random	TSP20	3.77	0.32	3.78	0.33
	TSP50	5.52	0.27	5.54	0.28
	TSP100	7.82	0.23	7.85	0.26
Clustered	TSP20	2.79	0.28	2.78	0.28
	TSP50	3.73	0.25	3.73	0.25
	TSP100	7.41	0.22	7.40	0.23

表 5–3 实验所用数据集中，不同规模不同数据分布类型的验证集、测试集平均路径长度和标准差。

5.3 结果对比

本节对于几种不同算法 TSP20，TSP50 和 TSP100 三种规模的问题以及 Random 和 Clustered 两种数据分布类型在测试集上的相对误差 (Optimality Gap) 进行了对比。以优化求解器 Gurobi 的结果作为评价基准 OPT，某算法结果 L 的相对误差

$$Gap = \frac{L - OPT}{OPT} \times 100\%$$

各种方法的结果对比表格如表5–4所示。

分布类型	方法	TSP20		TSP50		TSP100	
		路径长度	相对误差	路径长度	相对误差	路径长度	相对误差
Random	Gurobi	3.78	0%	5.54	0%	7.85	0%
	DM	3.95	5.02%	6.23	12.91%	10.20	29.75%
	GCN+DM	3.83	1.85%	5.84	5.82%	8.64	10.11%
	GCN+GS	3.89	3.10%	5.96	7.17%	9.23	17.54%
	GCN+BS	3.78	0.11%	5.66	2.35%	8.36	6.49%
Clustered	Gurobi	2.78	0%	3.73	0%	7.40	0%
	DM	2.93	5.43%	4.22	13.26%	9.73	31.43%
	GCN+DM	2.85	2.36%	3.97	6.34%	8.22	11.03%
	GCN+GS	2.89	3.83%	4.06	8.96%	8.79	18.80%
	GCN+BS	2.78	0.17%	3.84	3.16%	7.91	6.92%

表 5–4 各种算法求解 TSP20，TSP50 和 TSP100 的表现对比。DM 表示 Dang et al. (2002) 的迭代算法，GS 为贪婪搜索，BS 为集束搜索

可以发现，各种算法在 Clustered 类数据分布的测试集上的相对误差比 Random 类数据要大，这是因为 Clustered 类数据分布更为集中，相对而言有更多点分布在输入图较为靠内的位置，最终找出的 TSP 路径，即哈密尔顿回路有较多、较深、接近图中心的“褶皱”，这增大了解决 TSP 问题的难度。Clustered 类数据对贪婪搜索影响尤其大，因为此类数据更容易出现图3–1所示的局部最优情况，而贪婪搜索很容易被困在较差的局部最优情况，无法找出全局最优解。

总体来看，传统迭代算法 (Dang et al. 2002) 在结合了 GCN 形成双向学习系统后，不论问题规模和数据分布类型如何，整体的表现均较其本身有较大提升，可见该系统确实可以提高传统算法的表现，GCN 和迭代算法之间的互相更新提高是比较有效的。三种双向深度学习系统中，GCN+ 集束搜索表现最佳，最为接近 Gurobi 给出的最优解。尤其是在解决规模较小的 TSP 问题时，相对误差分别能达到 0.11%(TSP20) 和 2.35%(TSP50)。这样的结果可能与三种双向深度学习系统的训练方式有关，对于 GCN+ 迭代算法系统，由于迭代算法本身就是 TSP 问题算法，我们考虑使用无监督的训练方式，由迭代算法给出路径供 GCN 学习。而 GCN+ 集束搜索系统则是半监督学习，开始需要由 Gurobi 提供最优路径使该系统能够正常初始化，因此其表现会略好于 GCN+ 迭代算法。而 GCN+ 贪婪搜索系统因为贪婪搜索容易被困在局部最优的缺点，尽管其训练过程和 GCN+ 集束搜索一样也是半监督的，仍在三种双向深度学习系统中表现最差。

此外，三种双向深度学习系统在处理 TSP100 问题时表现都不是很好，可能是受限于训练数据集的数量不足。深度神经网络需要大量的数据进行训练，因此进一步增加训练数据量，该双向深度学习系统的表现可能会更好。

5.4 结果可视化

为更好地展现双向深度学习系统辅助传统算法进行 TSP 问题求解的过程，对传统算法和 GCN 结合后构成的双向深度学习系统提高原算法表现的原因进行进一步的解释分析，本节我们对不同的问题规模和数据分类各举一例进行可视化展示，以更为直观地观察 GCN 给出的启发信息在双向深度学习系统中起到的作用及其对传统算法的优化过程。可视化过程相当于测试阶段，过程中不再进行双向的信息传播和参数更新，直接使用 GCN+ 迭代算法形成的双向深度学习系统训练好的 GCN 模型。GCN 的输出同样传给贪婪搜索和集束搜索寻找路径，以方便结果对比。

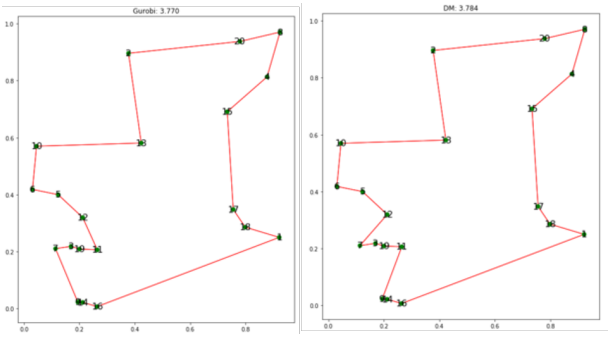

图 5–2 TSP20，Random 类分布，从左到右依次是 Gurobi（路径长 3.770）和迭代算法（路径长 3.784）。

从上图可以看出，问题规模较小且问题较为简单（Random 类分布，节点分布均匀）时，

路径上内折较少且其深度较浅，因此迭代算法的结果与 Gurobi 相差并不算大。二者路径差距主要存在于左下角的一簇节点的部分路径选取。

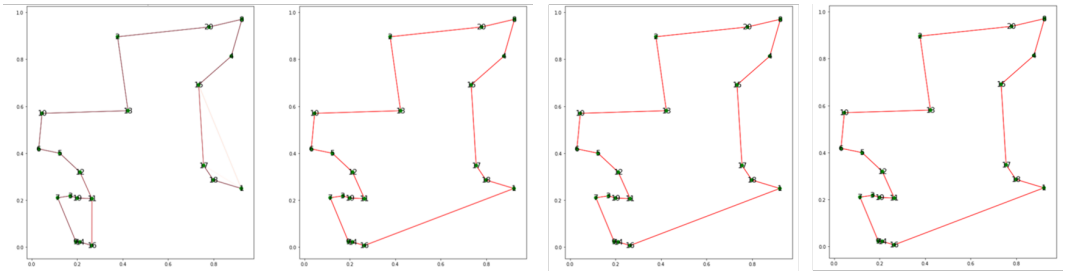

图 5–3 TSP20，Random 类分布，从左到右依次是 GCN 输出的邻接概率矩阵（边颜色越深表示 GCN 预测这条边出现在 TSP 路径中概率越大），GCN+ 迭代算法（3.770），GCN+ 贪婪搜索（3.770），GCN+ 集束搜索（3.770）

可以看出，GCN 给出的邻接概率矩阵已经非常接近最终结果，传统算法只需连接 1 号节点与 16 号节点将路径闭合成回路即可。GCN 给出的信息成功帮助迭代算法进行了路径的优化，左下角选择了 12 号节点连接 11 号、7 号连接 9 号，这和 Gurobi 的最优解是一样的。在 GCN 给出的邻接概率信息下因此三种算法都找到了最优解。

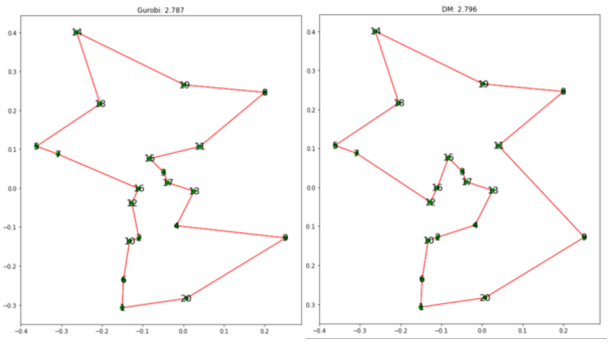

图 5–4 TSP20，Clustered 类分布，左为 Gurobi 的解（2.787），右为迭代算法的解（2.796）

这个 Clustered 类测试样例较 Random 类更为复杂，节点分布更为密集。从图中可见大量节点分布在图的中心区域，这回导致 TSP 路径向内的凹陷较多且程度较深，较难处理。迭代算法解和最优解的区别正在于遍历图中心区域几个节点时的部分路径。

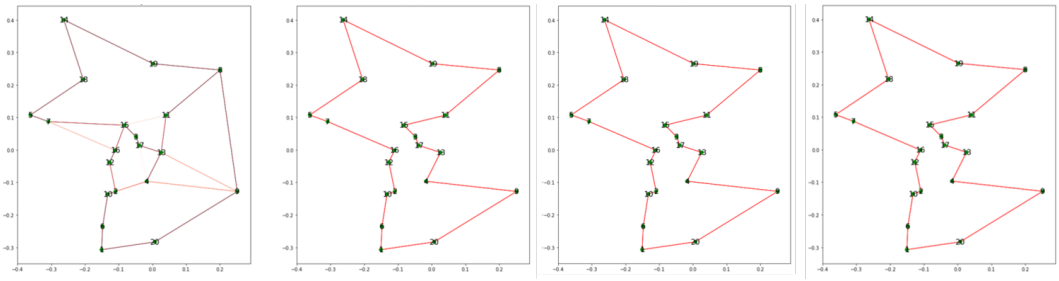

图 5–5 TSP20，Random 类分布，从左到右依次是 GCN，GCN+ 迭代算法（2.787），GCN+ 贪婪搜索（2.787），GCN+ 集束搜索（2.787）

从 GCN 给出的 TSP 路径预测信息来看，16-12-2 这条部分路径的概率是非常大的（颜

色较深），这和最优路径的选择是一模一样的。路径 7-16 也有一定的选中概率，而迭代算法原本选择的 7-12 这条边的选中概率为 0，反应在概率矩阵图中即没有连线（事实上颜色深度为 0，为白色所以不可见）。这样的信息帮助迭代算法优化了图中心区域的部分路径（放弃 7-12，选择 7-16-12-2），迭代算法此时在 GCN 的辅助作用下找到了最优解。由于 GCN 给出了有效的启发信息，贪婪搜索和集束搜索也找到了最优解。

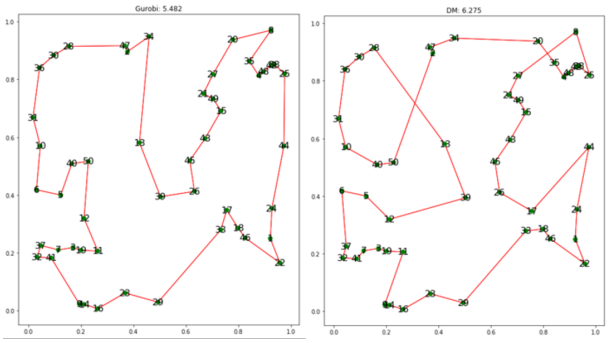

图 5–6 TSP50，Random 类分布，左为 Gurobi 的解（5.482），右为迭代算法的解（6.275）。

从图中可见，对于这个样例，迭代算法给出的解和最优 TSP 路径的差距是比较大的，对于此例相对误差超过的 14%。具体而言，迭代算法找出的这条路径含有大量的交叉。根据章节 4.1 和 4.2 对该迭代算法缺陷和对局部最优情况的分析，我们认为这样的梯度下降策略有可能随着松弛变量的增大最终停止在局部最优而非全局最优的情况，即提前连接一簇节点中部分较近的节点，而被“剩下的”一些节点则只能在其余节点遍历完后花费较大代价从当前部分路径的末尾再去连接，这样就会在路径中出现比较多的交叉路径。

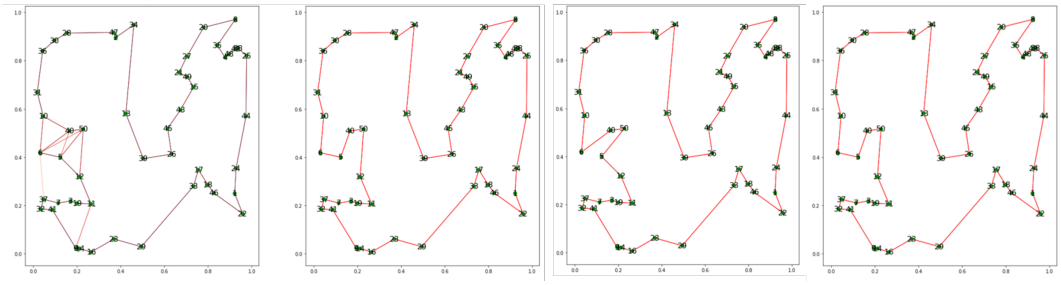

图 5–7 TSP50，Random 类分布，从左到右依次是 GCN，GCN+ 迭代算法（5.482），GCN+ 贪婪搜索（5.512），GCN+ 集束搜索（5.482）。

GCN 对于图5–8的样例给出了非常好的 TSP 路径预测信息。从邻接概率矩阵图来看，GCN 已经几乎给出了完整的 TSP 路径。而且除了图左侧 10、40、6、5、12 这几个节点之外，其余各节点之间的连线几乎没有任何多余路径作为干扰信息，完全和最优路径的选择相同。事实上 GCN 给出的这些节点之间连线对应边的选中概率是非常接近 1 的（超过 0.99），迭代算法直接将该邻接概率矩阵整数化就可以找出除了以上 5 个节点以外的最优部分路径。借助 GCN 给出的信息，迭代算法成功找出了最优路径。集束搜索也同样找出了最优路径，而贪婪搜索因为边 6-40 的概率比 6-5 更大，陷入了非全局最优的局部最优情况，在上述的 5 个节点的部分路径选择中出现了错误。尽管如此，由于 GCN 的预测信息足够好，贪婪搜索找出的其余部分路径都是最优的，因此总的相对误差也不大，仅 $\frac{5.512-5.482}{5.482} = 0.5\%$。

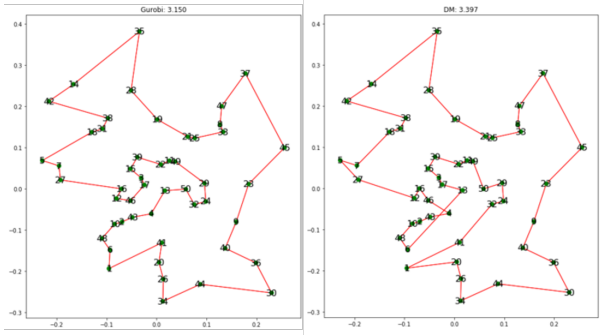

图 5–8 TSP50，Clustered 类分布，左为 Gurobi 的解（3.150），右为迭代算法的解（3.397）。

迭代算法本身在处理这个样例时结果较好，高于其在 TSP50，Clustered 类数据集上的平均表现。图中可见其与最优路径的主要差距在于左下角和图中心两簇节点的部分路径。

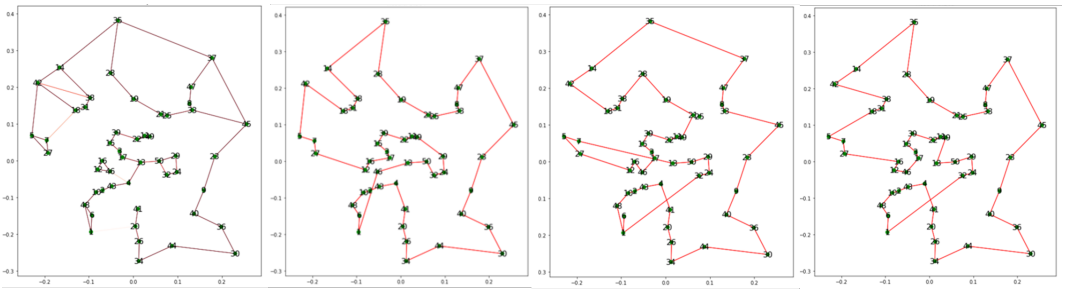

图 5–9 TSP50，Clustered 类分布，从左到右依次是 GCN，GCN+ 迭代算法（3.304），GCN+ 贪婪搜索（3.443），GCN+ 集束搜索（3.263）。

在处理这个 Clustered 类样例时，GCN 给出的路径预测并不如上一例好，因此迭代算法、集束搜索、贪婪搜索都没有找到最优路径。但 GCN 的路径预测信息仍然有效指导了迭代算法的路径优化。迭代算法结合 GCN 后给出的路径较其独立给出的路径长度有一定幅度的减小（从 3.397 到 3.304），主要体现在左下角一簇节点的部分路径优化。

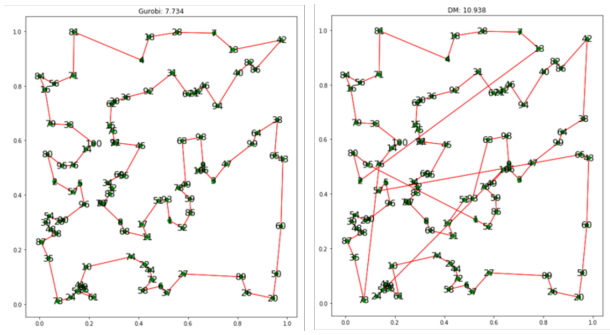

图 5–10 TSP100，Random 类分布，左侧为 Gurobi 结果（7.734），右侧为迭代算法（10.938）

迭代算法对这个样例给出的结果较差。其给出的 TSP 路径中存在多个交叉，这样的错误和前面 Random50 样例的缺点类似，都是为了选择局部最优路径而造成部分节点的遗漏，最后又强行连接。

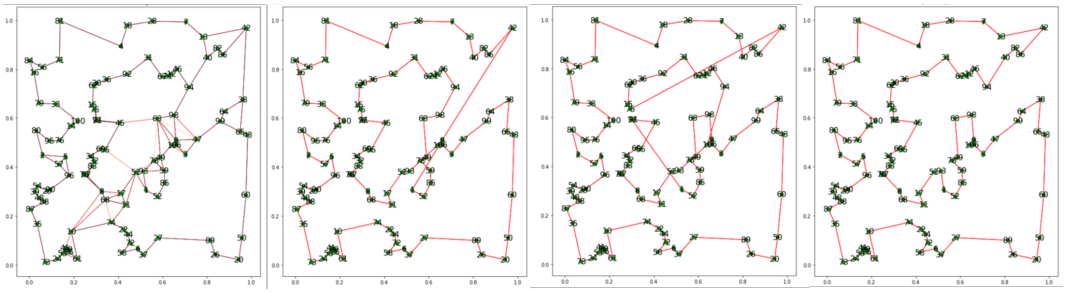

图 5–11 TSP100，Random 类分布，从左到右依次是 GCN，GCN+ 迭代算法（8.365），GCN+ 贪婪搜索（8.852），GCN+ 集束搜索（7.734）。

GCN 对这个样例提供了有效的预测信息。预测的邻接概率矩阵中没有任何交叉路径有较大概率。迭代算法新的路径中不再有大量交叉，仅仅因为路径收尾部分 42 号节点和 1 号相连产生了交叉。虽然较差的局部最优情况依然存在，42 号节点是之前遍历时遗漏的节点，但是这种情况出现的次数比利用 GCN 信息优化前大大减少。GCN+ 贪婪搜索依然由于存在较多较差局部最优情况，表现最差，其路径交叉比 GCN 迭代算法更为严重。而 GCN+ 集束搜索则展现了其优势，通过多条待选路径规避较差的局部最优找到了全局最优路径。

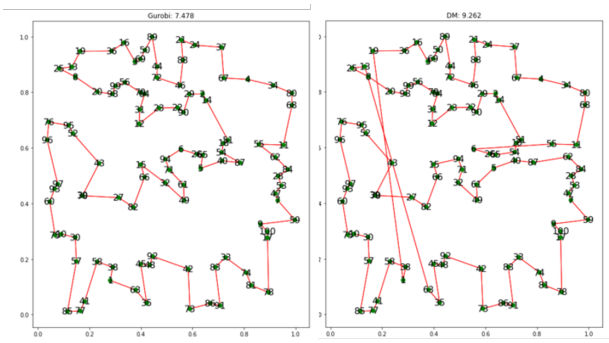

图 5–12 TSP100，Clustered 类分布，左侧为 Gurobi 结果（7.478），右侧为迭代算法（9.262）

对于这个样例，迭代算法给出的结果和最优解的区别主要在于图中的两个大交叉（1-19，13-68），出现这种交叉的原因仍是局部最优，19 号，13 号节点是左上角一簇节点寻找部分路径时遗漏的节点，不得不在最后为完成遍历从远处进行连接。

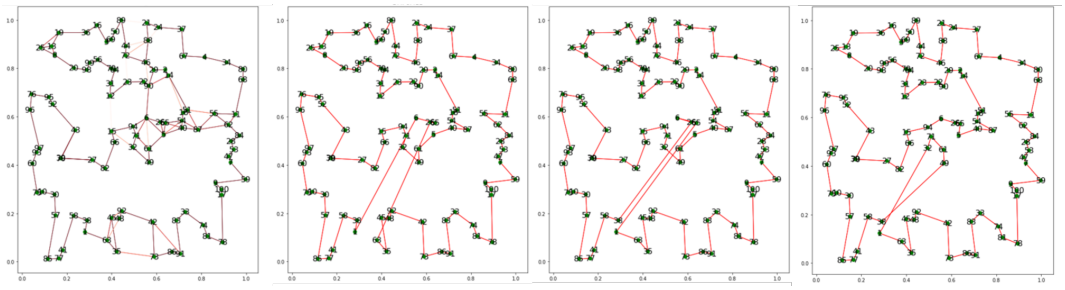

图 5–13 TSP100，Clustered 类分布，从左到右依次是 GCN，GCN+ 迭代算法（8.411），GCN+ 贪婪搜索（8.518），GCN+ 集束搜索（8.100）。

GCN 提供的邻接概率信息使得迭代算法的表现有了明显提升，虽然仍有路径交叉，但原本的两个大交叉不再存在。这个样例三种方法结果差异不大，借助 GCN 的启发信息都没

有找到最优解，因为 GCN 给出的邻接概率矩阵在图的中部靠左和底部附近的两簇节点对部分路径的预测信息不是很明确。

5.5 本章小结

本章对使用双向深度学习系统进行的实验进行了介绍，首先讨论了实验中的超参数设置，如图卷积网络结构参数，学习率，迭代算法的松弛变量，集束搜索的集束宽度等，以便后续实验得到较好的结果。之后对实验的数据集进行概览，并对各种算法在不同问题规模、不同数据分布类型的测试集上的表现进行了对比分析。最后对部分实验结果进行了可视化处理，以便更为直观地观察双向深度学习对传统算法的提升作用。

第六章　结论

本文提出了一种双向深度学习系统，通过结合残差门控图卷积网络与传统迭代算法形成无监督的双向深度学习系统来提高传统迭代算法的表现。一方面通过图卷积网络学习图的结构信息和空间信息，以邻接概率矩阵的形式给出其对每条边出现在 TSP 路径中概率的预测，指导传统算法进行 TSP 路径优化；另一方面将优化后的 TSP 路径反馈给图卷积网络用于反向传播，二者互相更新互相提高。该双向深度学习系统同样可以将 GCN 的输出作为启发信息，结合 GCN 与启发式搜索算法，如贪婪搜索和集束搜索来进行半监督的双向深度学习。实验结果表明，传统迭代算法在与图卷积神经网络结合形成双向深度学习系统后，对于不同的数据分布类型和不同问题规模，表现均有显著提高。同时，结果表明当问题规模一致时，目前使用的几种算法在处理 Clustered 类样例（节点分布更为密集，有较多内部节点）时的表现均比处理 Random 类样例（节点随机分布）时差，这说明对于 TSP 问题而言节点分布密集时较难处理。此外，在所有尝试构建的双向深度学习系统中，图卷积网络 + 集束搜索表现最佳，对于节点随机分布的样例，TSP20,TSP50,TSP100 的测试集上相对误差分别达到 0.11%，2.35% 和 6.49%。

目前工作的不足之处在于，该双向深度学习系统的泛化能力较差，因为对于不同规模的问题训练出的 GCN 模型并不通用，如用 TSP20 对应的 GCN 模型解决 TSP100 问题会得到很差的结果，反之亦然。因此，未来工作可以考虑将该双向深度学习系统与迁移学习或强化学习相结合，以提高该系统的泛化能力与迁移能力。

参考文献

[1] BRUNA J, ZAREMBA W, SZLAM A, et al. Spectral Networks and Locally Connected Networks on Graphs[J]. Computer Science, 2013.

[2] SCARSELLI F, GORI M, TSOI A C, et al. The Graph Neural Network Model[J]. IEEE Transactions on Neural Networks, 2009, 20(1): 61-80.

[3] DEFFERRARD M, BRESSON X, VANDERGHEYNST P. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering[J]., 2016.

[4] WECKMAN G R, GANDURI C V, KOONCE D A. A neural network job-shop scheduler. Journal of Intelligent Manufacturing.[J]., 2008, 19(2): 191-201.

[5] DAI H, KHALIL E B, ZHANG Y, et al. Learning Combinatorial Optimization Algorithms over Graphs[J]., 2017.

[6] VENKATAKRISHNAN S B, ALIZADEH M, VISWANATH P. Graph2Seq: Scalable Learning Dynamics for Graphs[Z]. 2018. arXiv: 1802.04948.

[7] VINYALS O, FORTUNATO M, JAITLY N. Pointer Networks[J]. Advances in Neural Information Processing Systems, 2015, 28.

[8] BELLO I, PHAM H, LE Q V, et al. Neural Combinatorial Optimization with Reinforcement Learning[J]., 2016.

[9] MNIH V, KAVUKCUOGLU K, SILVER D, et al. Playing Atari with Deep Reinforcement Learning[Z]. 2013. arXiv: 1312.5602.

[10] KOOL W, van HOOF H, WELLING M. Attention, Learn to Solve Routing Problems![Z]. 2019. arXiv: 1803.08475.

[11] VELIČKOVIĆ P, CUCURULL G, CASANOVA A, et al. Graph Attention Networks[Z]. 2017. arXiv: 1710.10903.

[12] NOWAK A, VILLAR S, BANDEIRA A S, et al. Revised Note on Learning Algorithms for Quadratic Assignment with Graph Neural Networks[Z]. 2017. arXiv: 1706.07450.

[13] MEDRESS M F, COOPER F S, FORGIE J W, et al. Speech understanding systems: Report of a steering committee[J]. Artificial Intelligence, 1977, 9(3): 307-316.

[14] BRESSON X, LAURENT T. Residual Gated Graph ConvNets[Z]. 2017. arXiv: 1711.07553.

[15] SAINBAYAR S, SZLAM A, FERGUS R. Learning Multiagent Communication with Backpropagation[Z]. 2016. arXiv: 1605.07736.

[16] MARCHEGGIANI D, TITOV I. Encoding Sentences with Graph Convolutional Networks for Semantic Role Labeling[Z]. 2017. arXiv: 1703.04826.

[17] JOSHI C K, LAURENT T, BRESSON X. An Efficient Graph Convolutional Network Technique for the Travelling Salesman Problem[J]., 2019.

[18] KIPF T N, WELLING M. Semi-Supervised Classification with Graph Convolutional Networks[Z]. 2016. arXiv: 1609.02907.

[19] LI Z, CHEN Q, KOLTUN V. Combinatorial Optimization with Graph Convolutional Networks and Guided Tree Search.[J]., 2018.

[20] DANG C, XU L. A Lagrange multiplier and hopfield-type barrier function method for the traveling salesman problem.[J]. Neural Comput, 2002, 14(2): 303-324.

[21] XU L. Deep IA-BI and Five Actions in Circling.[J]. IScIDE, 2019: 1-21.

[22] XU L. An overview and perspectives on bidirectional intelligence: Lmser duality,double ia harmony, and causal computation.[J]. IEEE/CAA J. Autom. Sin., 2019, 6(4): 865-893.

[23] DANG C, XU L. A globally convergent Lagrange and barrier function iterative algorithm for the traveling salesman problem.[J]. Neural Netw., 2001, 14(2): 217-230.

[24] BENGIO Y, LODI A, PROUVOST A . Machine Learning for Combinatorial Optimization: a Methodological Tour d'Horizon.[J]., 2018.

[25] VLASTELICA M, PAULUS A, MUSIL V. Differentiation of Blackbox Combinatorial Solvers.[J]., 2019.