计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.250200121

利用融合 2-opt 的强化学习算法求解 TSP 问题

彭俊龙[1]　范　静[2]
1 上海第二工业大学计算机与信息工程学院　上海 201209
2 上海第二工业大学数理与统计学院　上海 201209
(pjl201209@163.com)

摘　要　旅行售货商问题(Traveling Salesman Problem,TSP)是运筹学中经典的组合优化问题,属于 NP 难问题。问题的目标是求解旅行商的环游路径,使其在经过每个城市一次后返回起点且路径长度最短。为求解此问题,提出基于指针网络的深度强化学习算法(2+HRL),融合了 2-opt 算法和图注意力模型。使用图注意力网络提取城市节点的局部和全局结构信息,运用双向 LSTM 进行路径信息提取,期间利用 2-opt 策略,通过局部交换改进路径;进而使用 REINFORCE 算法进行策略网络的梯度优化,结合熵奖励函数避免陷入局部最优解,使用值函数对评价网络参数进行改进。实验结果证明,2+HRL 优于传统启发式算法和精确算法,而且与一些深度强化学习算法相比较时,在较低的训练次数下,2+HRL 具有更快的计算速度,更准确的计算精度;在增加训练次数后,模型的优化效果也超越了相比较的其他深度强化学习算法。

关键词:图注意力网络,旅行售货商问题,深度强化学习,2-opt,组合最优化
中图分类号　TP181

Hybrid Reinforcement Learning Algorithm Combined with 2-opt for Solving Traveling Salesman Problem

PENG Junlong[1] and FAN Jing[2]
1 College of Computer and Information Engineering,Shanghai Polytechnic University,Shanghai 201209,China
2 School of Mathematics,Physic and Statistics,Shanghai Polytechnic University,Shanghai 201209,China

Abstract　TSP is a classic NP-hard combinatorial optimization problem in operations research,aimed at the shortest possible cycle that visits each city exactly once from the origin point and returns back. For solving TSP,this paper presents a hybrid deep reinforcement learning approach(2+HRL) based on a pointer network,integrating graph attention mechanisms and the 2-opt heuristic. Specifically,graph attention networks(GATs) capture both local and global structural information of cities,while bidirectional LSTMs dynamically encode path dependencies for context-aware state representation. During training stage,the 2-opt strategy iteratively improves paths by local edge swaps to enhance solution quality. The policy gradient optimization via the REINFORCE algorithm is combined with an entropy reward function to avoid local optima,while a value network enhances parameter estimation accuracy. Experimental results show that 2+HRL algorithm performs better than traditional heuristics and exact algorithms,and it has faster computational speed and higher precision if limited with fewer training iterations,and its performance exceeds other deep reinforcement learning approaches as training progresses increment.

Keywords　Graph attention network,Traveling salesman problem,Deep reinforcement learning,2-opt,Combinatorial optimization

1　引言

旅行售货商问题(TSP)要求找到经过每个城市一次并返回起点的最短路径。在二维对称 TSP 问题中,城市节点 s_i ($i\in\{1,\cdots,n\}$)的坐标以数组(x_i,y_i)的形式给出,目标是需要找到一条节点不重复的路径 $S=(\boldsymbol{s}_{k_1},\boldsymbol{s}_{k_2}\cdots,\boldsymbol{s}_{k_n})$,使路径的长度最短。若用 $\|s_i-s_j\|_2$ 表示 s_i 与 s_j 的欧氏距离,则 TSP 的目标可写为:

$$\min L(S)=\|\boldsymbol{s}_{k_n}-\boldsymbol{s}_{k_1}\|_2+\sum_{i=1}^{n-1}\|\boldsymbol{s}_{k_i}-\boldsymbol{s}_{k_{i+1}}\|_2 \tag{1}$$

多年来,众多学者针对 TSP 提出不同的求解方法。这些方法可以分为精确算法和启发式算法。精确算法,如 Concorde 求解器[1],在小规模 TSP 问题上表现优异,但在大规模问题上耗时过长,难以扩展。启发式算法,如最近邻算法[2]、LKH 算法[3]、k-opt[4]、模拟退火算法[5]等,在较短时间内可以得到可接受的解,但其性能依赖于初始解,并易陷入局部最优解。这限制了其在更复杂场景中的适用性。总之,由于 TSP 问题的 NP 难性质,传统方法在求解大规模实例时,所需时间呈指数级增长,难以推广使用。

于是,人们便将目光转向了计算机领域。其实,早在 1985 年 Hopfield 等[6]就尝试用神经网络求解 TSP 问题,但收效甚微。近年来,深度学习和强化学习的发展为解决 TSP 问题提供了新思路,可以在无需手工设计启发式规则的情况下优化解的质量并提高效率。之后,基于监督学习、强化学习以及相互结合的新型算法引入了自动学习机制[7],能够扩展到各式组合优化问题中。Vinyals 等[7]提出了基于监督学习的模型,但依赖于大量标注数据,只适用于规模较小且结构简

通信作者:范静(fanjing@sspu.edu.cn)

单的场景。Nowak 等[8]基于图神经网络的监督学习框架，通过束搜索生成可行解。Bello 等[9]采用基于强化学习的方法，通过与环境的交互学习求解策略，避免了对标注数据的依赖，具备更强的扩展性。Khalil 等[10]结合图卷积网络和强化学习，提出一种新的 TSP 求解方法，通过学习图节点的局部和全局结构特性提升求解效率。Kool 等[11]利用注意力机制设计了一种基于 Transformer 的模型，增强了算法处理复杂图结构问题的能力。Joshi 等[12]用基于图卷积网络的模型来解决旅行商问题，但该方法作为非自回归模型，面临着输出条件受限以及束搜索带来的高昂计算成本这两方面的挑战。尽管这些方法在局部优化和路径生成上取得了显著进展，但存在依赖额外程序或技术的局限性。

为提升启发式算法性能，研究者开始探索自主学习改进策略的方法。例如，Wu 等[13]结合 Transformer 架构和图注意力网络，使用 2-opt 方法进行节点交换，但固定输出嵌入限制了其扩展性。Deudon 等[14]改进神经组合最优化模型并在推理阶段加入 2-opt 操作以提高解的质量。Da Costa 等[15]采用了图卷积网络(GCN)来编码图信息，使得算法更具扩展性和效率。然而，GCN 的表达能力有限，可能无法充分捕捉所有关键信息。

因此，为进一步提升求解效率和模型的自适应性，开发无需额外程序支持且能够自动优化的解决方案，是一个需要持续研究的方向。本文在这方面进行探索，提出了一种基于策略梯度的深度强化学习算法，通过 GAT 图注意力网络提取城市节点信息，结合指针网络指导节点选择，并通过 2-opt 移动逐步优化路径，力争在样本信息提取和收敛速度上更为高效，更快地接近最优解。

2 *k*-opt 启发式算法

k-opt 是一种常用的局部搜索算法，常用于求解 TSP 等组合优化问题。通过移除当前路径中的 k 条边，并以不同方式重新连接路径片段，试图找到更优解。在实际应用中，k 的取值通常为 2 或 3，即 2-opt 和 3-opt 算法。

Sui 等[16]将 3-opt 融入深度强化学习和特征调制网络中，以提升求解 TSP 问题的效率。然而，就如 Uddin 等[17]所说，3-opt 的计算复杂度较高，尤其在大规模问题中，重连组合过多导致运行时间变长。相比之下，2-opt 由于仅移除两条边进行重新连接，计算复杂度较低，更适合大规模问题或需要快速生成可行解的场景。

在 2-opt 算法中，选择一个初始路径 $\boldsymbol{S}_0=(\boldsymbol{s}_1,\boldsymbol{s}_2,\cdots,\boldsymbol{s}_n)$，该路径访问每个城市一次并返回起点形成闭合回路。接着，在当前路径中选择两条不相邻的边 $(\boldsymbol{s}_i,\boldsymbol{s}_{i+1})$ 和 $(\boldsymbol{s}_j,\boldsymbol{s}_{j+1})$ $(i<j)$，将这两条边移除，并连接 $(\boldsymbol{s}_i,\boldsymbol{s}_j)$ 和 $(\boldsymbol{s}_{i+1},\boldsymbol{s}_{j+1})$，形成新路径 $S_i=(\boldsymbol{s}_1,\cdots,\boldsymbol{s}_i,\boldsymbol{s}_j,\boldsymbol{s}_{j-1},\cdots,\boldsymbol{s}_{i+1},\boldsymbol{s}_{j+1},\cdots,\boldsymbol{s}_n)$(见图 1)。路径长度变化量 ΔM 定义为：

$$\Delta M=\|\boldsymbol{s}_i-\boldsymbol{s}_{i+1}\|_2+\|\boldsymbol{s}_j-\boldsymbol{s}_{j+1}\|_2-\|\boldsymbol{s}_i-\boldsymbol{s}_j\|_2-\|\boldsymbol{s}_{i+1}-\boldsymbol{s}_{j+1}\|_2 \tag{2}$$

若 $\Delta M>0$，即新路径长度短于原路径长度，则认为 2-opt 操作为“理想的”；若 $\Delta M<0$，则认为 2-opt 操作为“不理想的”。重复执行上述操作，直到达到迭代次数或无法找到任何改进为止。

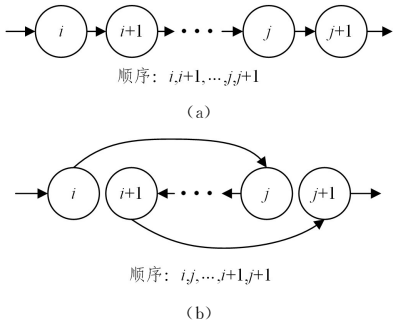

图 1 (b)为(a)进行 2-opt 中所得路径

Fig. 1 (b) is the path after performing 2-opt form (a)

尽管 2-opt 算法在提高解的质量、优化求解时间和简化操作过程中取得了良好平衡，但它仍然容易陷入局部最优解。为了克服局部最优的限制，本文将机器学习算法与 2-opt 结合，提出一种基于强化学习的策略来寻找 TSP 的最优解。

3 TSP 建模转换

将 TSP 问题建模成马尔可夫决策过程(MDP)，定义状态 S，动作 A，状态转移，奖励函数 R 和折扣因子 λ。Wang 等[18]证明参数能全面描述旅行售货商与城市网络之间的交互过程，帮助做出最优决策并最大化长期回报。参数的具体设置为：状态 S 定义为 (S_t,S_t')，其中 S_t 为时间步 t 的当前状态，S_t' 为历史搜索过程中的最优状态，即：

$$S_t'=\min_{S\in(S_0,S_1,\cdots,S_{t-1})}L(S)$$

针对 TSP 问题，状态 S 与路径 S 等同。

1)动作 A 定义为 (A_1,A_2)，其中 A_1，A_2 分别代表 2-opt 移动中的起点和终点。例如图 1 中的 2-opt 移动由动作 (i,j) 得到。

2)奖励函数 R_t 定义为，在当前状态 S 执行动作 A 后获得的奖励的期望值，即 $R_t=L(S_t')-L(S_{t+1})$。其中，$L(S_t')$ 是直到时间步 t 为止找到的最短路径。当 R_t 为负时认为奖励为正，反之奖励为负。

3)状态转移函数定义为，若奖励函数的值为正，则更新历史最优状态 S' 为当前状态 S；反之，则不进行任何改动。

4)最大期望回报定义为 $G_t=\sum_{i=t}^{T-1}\lambda^{i-t}R_i$，表示从当前时间步 t 开始，直到最大时间步 T 的累计期望回报，其中 λ 代表折扣因子，R_i 代表在时间步 i 获得的即时奖励。

4 强化学习算法

本文使用强化学习算法，通过累计奖励函数，不仅考虑当前步骤的即时改进，还评估这些决策对未来产生的长期影响。相比于关注短期优化的贪婪算法，强化学习算法可以根据路径的全局特性选择对全局解更有利的路径。

本文基于指针网络，整合了评论家-行动家模型和编码-解码模型，简记为 2+HRL。本文将模型划分为两部分：策略网络和评价网络(见图 2)。策略网络接受当前状态 S 和图的拓扑结构作为输入，输出最优策略 $\pi_\theta(A|S|)$(见 4.3.2 节)，然后进行 2-opt 移动，从而生成一个新的候选路径 S_i。评价网络专注于处理历史最优状态 S_t'，输出状态价值估计 $\boldsymbol{V}_\phi(\bar{\boldsymbol{S}})$(见 4.4 节)来评估当前路径的质量。在每次路径调整后，算法更新策略网络和评价网络的参数。

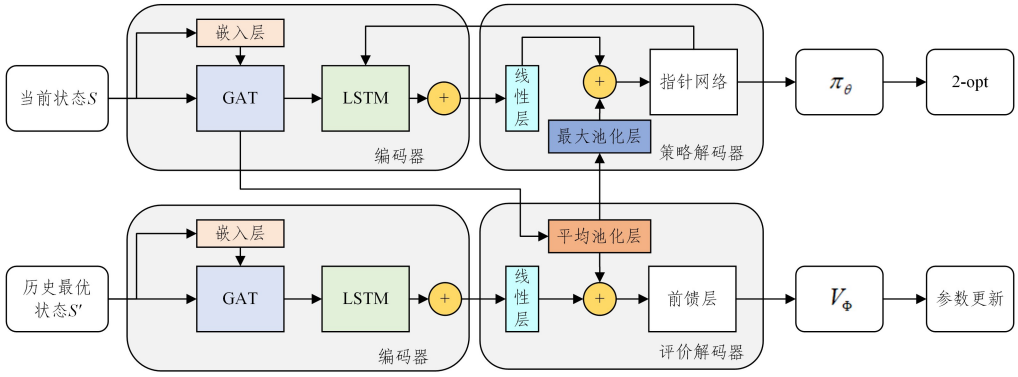

图 2　模型整体架构

Fig. 2　Overall architecture of the model

4.1　嵌入层

在模型的嵌入层，首先对初始随机路径S_0进行处理。将二维坐标通过嵌入操作转换为$\boldsymbol{x}_i^0=\boldsymbol{W}_x\boldsymbol{x}_i+\boldsymbol{b}_x$，$\boldsymbol{W}_x$和$\boldsymbol{b}_x$分别为映射矩阵和偏置，且$\boldsymbol{W}_x\in\mathbb{R}^{d\times d}$，$\boldsymbol{b}_x\in\mathbb{R}^d$。然后计算任意节点$\boldsymbol{s}_i$与节点$\boldsymbol{s}_j$间的欧氏距离$\boldsymbol{e}_{ij}=\|\boldsymbol{s}_i-\boldsymbol{s}_j\|_2$。考虑到本文聚焦于对称 TSP 问题，所得出的欧氏距离矩阵必须为对称矩阵。为确保其在计算过程中保持对称性，进行对称归一化处理，计算公式为：

$$\bar{\boldsymbol{e}}_{ij}=\frac{\boldsymbol{e}_{ij}}{\sqrt{\sum_{i=1}^{n}\boldsymbol{e}_{ij}\sum_{j=1}^{n}\boldsymbol{e}_{ij}}} \tag{3}$$

4.2　图注意力网络(GAT)

本文针对 TSP 问题采用图注意力网络(GAT)(见图 3)。其基本思想是，通过计算邻居节点特征分数，使每个节点获得不同的权重。然后利用训练得出的自适应权重矩阵，计算各节点与邻居节点间的注意力分数。

本文使用层归一化方法对注意力机制进行处理，以避免信息丢失，并增强训练的稳定性。因此，注意力分数计算式为：

$$\boldsymbol{K}=LayerNorm(\mathrm{softmax}(\boldsymbol{W}_a\boldsymbol{x}_i+\boldsymbol{b}_a)) \tag{4}$$

其中，$\boldsymbol{W}_a\in\mathbb{R}^{d\times d}$为权重矩阵，$\boldsymbol{b}_a\in\mathbb{R}^d$为偏置项，$LayerNorm$为层归一化。

如图 3 所示的 GAT 结构图中，节点特征通过线性层映射为与节点数相同的维度。例如，对于有 20 个城市节点的 TSP 实例而言，将第 l 层节点坐标$\boldsymbol{x}_i^l$映射为 20×20 的矩阵。然后，通过 softmax 函数得到初始注意力分数，并使用层归一化进行标准化。在获得注意力分数 $\boldsymbol{K}$ 后，利用整合公式$\boldsymbol{m}_i^l=\boldsymbol{x}_i^l\times\boldsymbol{K}$与各个节点进行特征整合。因此，$\boldsymbol{m}_i^l$不仅可以反映自身属性，也综合了与其他节点的关系强度，能够增强模型对节点重要性的感知能力。

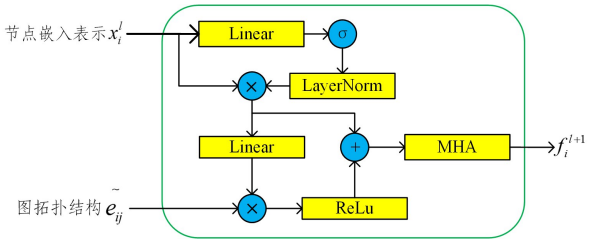

图 3　GAT 网络结构

Fig. 3　Architecture of GAT network

本文选择卷积操作体现每个节点及其邻域内其他节点的特征。卷积操作的计算式为：

$$\boldsymbol{m}_i^{l+1}=\boldsymbol{m}_i^l+ReLu\Big(\sum_{j\in N(i)}\bar{e}_{ij}(\boldsymbol{W}_g^l\boldsymbol{m}_j^l+\boldsymbol{b}_g^l)\Big) \tag{5}$$

其中，$\boldsymbol{W}_g^l\in\mathbb{R}^{d\times d}$，$\boldsymbol{b}_g^l\in\mathbb{R}^d$，$\boldsymbol{W}_g^l$和$\boldsymbol{b}_g^l$分别是 l 层的权重和偏置；$N(i)$表示节点 i 的邻接节点集合；$\bar{e}_{ij}$为对称归一化后的欧氏距离，ReLu 为线性整流函数，用于引入非线性机制，有助于捕捉更复杂的模式和特征组合，提高模型的表达能力。

对 L 层卷积层处理后得到的卷积输出$\boldsymbol{m}_i^L$，进一步运用多头注意力机制(Multi-Head Attention，MHA)，增强模型的特征提取能力。将注意力计算分为多个独立的头，每个头专注于数据的不同部分，计算出不同的注意力权重。MHA 的计算式为：

$$\boldsymbol{f}_i^L=\sum_{n=1}^{Q}ReLu\Big(\sum_{j\in N(i)}\bar{e}_{ij}(\boldsymbol{W}_z^L\boldsymbol{m}_j^L+\boldsymbol{b}_z^L)\Big) \tag{6}$$

其中，$\bar{e}_{ij}$为城市间的欧氏距离，$\boldsymbol{W}_z^L\in\mathbb{R}^{d\times d}$为权重，$\boldsymbol{b}_z^L\in\mathbb{R}^d$为偏置，$Q$为 MHA 中的头数。最终，使用$\boldsymbol{Z}_i=\boldsymbol{f}_i^L$来表示每个节点的嵌入表示。通过将卷积输出与 MHA 进行结合，可以从不同角度理解邻居节点间的关系，有利于模型捕捉全局信息。此外，通过并行学习的方式，还可以提高模型的鲁棒性和泛化能力。

4.3　策略网络

4.3.1　策略编码器

为表示学习状态 S 的节点序列，本文采用 LSTM 网络。考虑到研究对象为对称 TSP 问题，即 TSP 图为无向完全图，故设置正向和反向两个 LSTM(见图 4)。

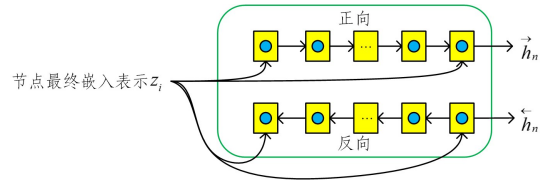

图 4　双向 LSTM 简略图

Fig. 4　Simplified diagram of bidirectional LSTM

图 4 描述了处理和学习节点序列特征的过程，其中，$\boldsymbol{Z}_i$为节点最终嵌入表示，$\overrightarrow{\boldsymbol{h}}_n$和$\overleftarrow{\boldsymbol{h}}_n$分别对应正向和反向 LSTM 的输出隐藏状态。正向 LSTM 从序列的开始到结束处理节点表示$\overrightarrow{\boldsymbol{z}}_i$，并生成最终隐藏状态$\overrightarrow{\boldsymbol{h}}_n$；反向 LSTM 则从序列的结束到开始处理，生成最终隐藏状态$\overleftarrow{\boldsymbol{h}}_n$。这种双向结构允许模型不仅捕捉从前到后的依赖性，而且也能从后到前的捕捉依赖性，

从而提供一个更全面的序列表示。

图 5 展示了正向 LSTM 和反向 LSTM 的详细过程，以左侧的正向 LSTM 为例，节点最终嵌入表示$\vec{z}_i$与前一刻的隐藏状态$\vec{h}_{i-1}$输入到遗忘门，以决定需要遗忘的记忆。同时，输入门确定哪些新信息将被存储到细胞状态，并将新信息添加到细胞状态中，从而更新其内容。接着，新的细胞状态$\vec{c}_i$，上一个时间步的隐藏状态$\vec{h}_{i-1}$以及当前节点嵌入表示$\vec{Z}_i$被输入到输出门中，输出门根据这些信息生成当前的隐藏状态$\vec{h}_i$，并基于当前的细胞状态$\vec{c}_i$产生输出。

由于 TSP 问题中路径是一条环路，因此在学习节点路径表示时，需要将正向路径和反向路径的初始值分别设置为路径的末端和起点：

$$\begin{cases}(\vec{\boldsymbol{h}}_0,\vec{\boldsymbol{c}}_0)=LSTM(\vec{\boldsymbol{Z}}_n,\boldsymbol{0})\\(\overleftarrow{\boldsymbol{h}}_{n+1},\overleftarrow{\boldsymbol{c}}_{n+1})=LSTM(\overleftarrow{\boldsymbol{Z}}_1,\boldsymbol{0})\end{cases}\tag{7}$$

进而，将正反方向的节点路径整合为一个通用的节点表示：

$$\boldsymbol{o}_i=\tanh((\boldsymbol{W}_f\vec{\boldsymbol{h}}_i+\boldsymbol{b}_f)+(\boldsymbol{W}_b\overleftarrow{\boldsymbol{h}}_i+\boldsymbol{b}_b))\tag{8}$$

其中，$\boldsymbol{W}_f,\boldsymbol{W}_b\in\mathbb{R}^{d\times d}$，$\boldsymbol{b}_f,\boldsymbol{b}_b\in\mathbb{R}^d$。通过将正向隐藏状态$\vec{\boldsymbol{h}}_n$和反向隐藏状态$\overleftarrow{\boldsymbol{h}}_n$进行整合，最终得到 TSP 路径整体表示$\boldsymbol{h}_n=\vec{\boldsymbol{h}}_n+\overleftarrow{\boldsymbol{h}}_n$。

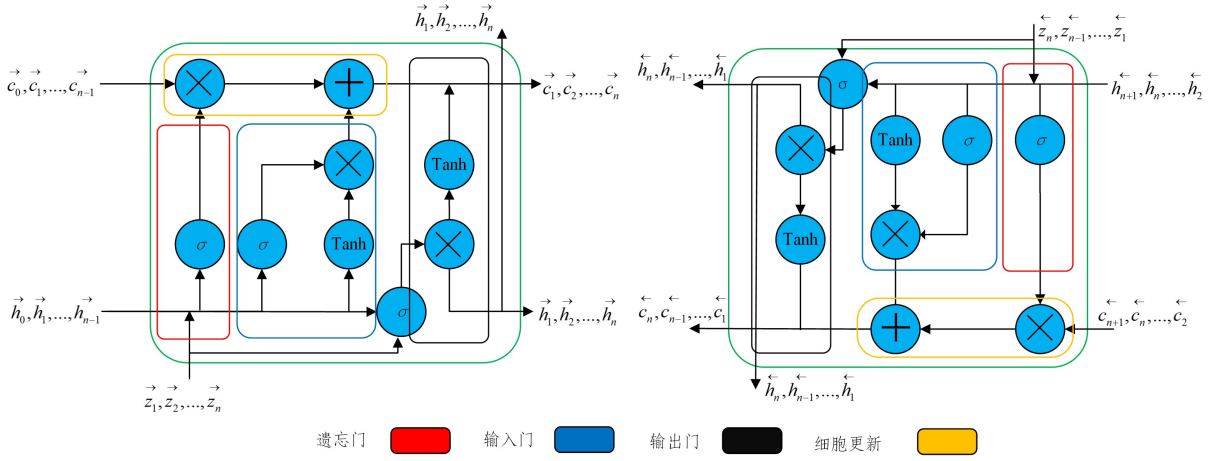

图 5 正向 LSTM 图(左)，反向 LSTM 图(右)

Fig. 5 Forward LSTM diagram(left), backward LSTM diagram(right)

4.3.2 策略解码器

这部分的目标是学习一个随机策略$\pi_\theta(A|S|)$，即在给定状态 S 的情况下，为能够减少路径长度的节点分配更高的选择概率。其计算式如下：

$$\pi_\theta(A|S|)=\prod_{i=1}^{k}\boldsymbol{P}_\theta(A_i|A_{<i},S|)\tag{9}$$

式(9)表示在给定状态 S 和动作$A_{<i}$的前提下，序列 A 中每个元素A_i的条件概率乘积。其中A_i对应于动作 A 中的第 i 个节点，$A_{<i}$表示已经采样过的点，θ 为网络参数。因使用 2-opt，故将 k 设置为 2。

为计算每个元素A_i的条件概率，需要将整合后的路径序列表示转化为查询向量 $\boldsymbol{q}_i$：

$$\boldsymbol{q}_i=\tanh((\boldsymbol{W}_q\boldsymbol{q}_{i-1}+\boldsymbol{b}_q)+(\boldsymbol{W}_o\boldsymbol{o}_{i-1}+\boldsymbol{b}_o))\tag{10}$$

其中，$\boldsymbol{b}_q,\boldsymbol{b}_o\in\mathbb{R}^{d\times d}$，$\boldsymbol{b}_q,\boldsymbol{b}_o\in\mathbb{R}^d$。并且，$\boldsymbol{o}_i$的初始值$\boldsymbol{o}_0$从均匀分布$U\left(-\frac{1}{\sqrt{d}},\frac{1}{\sqrt{d}}\right)$中随机初始化，而 $\boldsymbol{q}_i$的初始值$\boldsymbol{q}_0$为：

$$\boldsymbol{q}_0=\boldsymbol{W}_s\boldsymbol{h}_n+\boldsymbol{b}_s\parallel\boldsymbol{W}_{S'}\boldsymbol{h}_n'+\boldsymbol{b}_{S'}+\max(\boldsymbol{Z}_1,\boldsymbol{Z}_2,\cdots,\boldsymbol{Z}_n)\tag{11}$$

其中，$\boldsymbol{W}_S,\boldsymbol{W}_{S'}\in\mathbb{R}^{\frac{d}{2}\times d}$；$\boldsymbol{b}_S,\boldsymbol{b}_{S'}\in\mathbb{R}^{\frac{d}{2}}$；$\max(\boldsymbol{Z}_1,\boldsymbol{Z}_2,\cdots,\boldsymbol{Z}_n)$是当前状态 S 的节点嵌入表示的最大池化结果。

模型通过整合路径表示 $\boldsymbol{o}_i$和查询向量$\boldsymbol{q}_i$来预测节点在动作空间上的输出分布，以实现节点的选择机制，即：

$$\boldsymbol{u}_j^i=\begin{cases}\boldsymbol{v}^{\mathrm{T}}\tanh(\boldsymbol{K}\boldsymbol{o}_j+\boldsymbol{Q}\boldsymbol{q}_i), & j>A_{i-1}\\-\infty, & \text{其他}\end{cases}\tag{12}$$

其中，$\boldsymbol{K}$ 和 $\boldsymbol{Q}$ 为注意力矩阵，$\boldsymbol{K},\boldsymbol{Q}\in\mathbb{R}^{d\times d}$；$\boldsymbol{v}$ 为一个向量，$\boldsymbol{v}\in\mathbb{R}^d$；$A_{i-1}$为上一步骤选择的城市。设置 $j>A_{i-1}$，即当前选择的节点 j 必须位于前一个节点A_{i-1}之后。这样可以保证节点选择的过程中满足顺序约束。同时还可以过滤掉不满足顺序的点以减少搜索空间。得到节点的输出分布后，通过式(13)来计算节点的概率分布：

$$p_\theta(A_i|A_{<i},S)=\mathrm{softmax}(C\tanh(\boldsymbol{u}^i))\tag{13}$$

其中，$\boldsymbol{u}^i$为节点的输出分布，超参数 C 用于将$\boldsymbol{u}^i$限制在$[-C,C]$之间，从而确保输出在合理的范围内，使模型能够在小动作空间中高效地进行节点选择。最终，根据概率分布值选择的两个城市进行 2-opt 移动。

4.4 评价网络

评价网络与策略网络使用相同的编码器结构，均通过 GAT 网络提取节点特征，并使用双向 LSTM 来整合节点路径表示。两者的区别在于，评价网络中 GAT 网络输入的是历史最优状态S'，而非当前状态 S。

评价网络的目标是对当前状态进行价值估计。通过整合当前状态 S 和历史最优状态S'的路径表示$\boldsymbol{h}_n$和$\boldsymbol{h}'_n$，以及状态 S 的节点嵌入表示$\boldsymbol{Z}_i$，来计算状态的价值估计。路径表示提供了整体信息，反映了当前以及历史最优的路径特征，而节点嵌入则捕捉每个节点的局部特性。通过结合这些信息，评价网络能够全面分析当前状态，精确估算其价值，从而帮助模型判断路径优化方向。其计算式如下：

$$\boldsymbol{V}_\phi(\bar{\boldsymbol{S}})=\boldsymbol{W}_r ReLU(\boldsymbol{W}_z\left(\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{Z}_i+\boldsymbol{h}_v\right)+\boldsymbol{b}_z)+\boldsymbol{b}_r\tag{14}$$

其中，$\boldsymbol{W}_z\in\mathbb{R}^{d\times d}$，$\boldsymbol{W}_r\in\mathbb{R}^{1\times d}$，$\boldsymbol{b}_z\in\mathbb{R}^d$，$\boldsymbol{b}_r\in\mathbb{R}$，$\boldsymbol{h}_v=\boldsymbol{W}_v\boldsymbol{h}_n+\boldsymbol{b}_v\parallel\boldsymbol{W}_{v'}\boldsymbol{h}'_n+\boldsymbol{b}_{v'}$，且$\boldsymbol{W}_v,\boldsymbol{W}_{v'}\in\mathbb{R}^{\frac{d}{2}\times d}$。

5 策略梯度优化

在策略梯度优化过程中，本文采用 REINFORCE 算法。给定状态 S，定义总目标函数为 $\boldsymbol{J}(\theta)=\boldsymbol{E}_{\bar{S}\sim S}[\boldsymbol{J}(\boldsymbol{\theta}|\bar{\boldsymbol{S}})]$。进行梯度展开后，得到式(15)：

$$\nabla_\theta \boldsymbol{J}(\boldsymbol{\theta})\approx\frac{1}{B}\frac{1}{T}[\sum_{b=1}^{B}\sum_{t=0}^{T-1}\nabla_\theta\log\boldsymbol{\pi}_\theta(\boldsymbol{A}_t^b|\bar{\boldsymbol{S}}_t^b)(\boldsymbol{G}_t^b-\boldsymbol{V}_\phi(\bar{\boldsymbol{S}}_t^b))] \tag{15}$$

其中，优势函数定义为 $\boldsymbol{A}_t^b=\boldsymbol{G}_t^b-\boldsymbol{V}_\varphi(\bar{\boldsymbol{S}}_t^b)$，表示实际累计奖励 $\boldsymbol{G}_t^b$ 与状态价值估计 $V_\phi(\bar{S}_t^b)$ 的差异，用于衡量当前策略与状态估值的偏差；$\nabla_\theta\log\boldsymbol{\pi}_\theta(\boldsymbol{A}_t^b|\bar{\boldsymbol{S}}_t^b)$ 为策略的对数梯度，决定策略参数 θ 的更新方向。

传统的策略梯度更新方法容易出现两种情况：一种是“超调”，即更新幅度过大，错过奖励峰值；另一种情况为“失速”，即更新停留在梯度趋近零的区域，陷入局部最优解。为解决这些问题，本文引入熵奖励函数。熵奖励函数的计算式为：

$$\boldsymbol{H}(\boldsymbol{\theta})=-\frac{1}{B}\sum_{b=1}^{B}\sum_{t=0}^{T-1}\boldsymbol{E}_{\pi_\theta}[\log\pi_\theta(\cdot|\bar{\boldsymbol{S}}_t^b)] \tag{16}$$

使用熵奖励函数可以增加策略的不确定性，从而尝试更多的动作，还可以防止模型过拟合，并提高模型的泛化性能。

评价网络参数更新，将从时间 t 开始到结束的总回报 G 与状态 S 的价值函数的差的平方作累加：

$$\boldsymbol{L}(\boldsymbol{\phi})=\frac{1}{B}\frac{1}{T}[\sum_{b=1}^{B}\sum_{t=0}^{T-1}\|\boldsymbol{G}_t^b-\boldsymbol{V}_\varphi(\bar{\boldsymbol{S}}_t^b)\|_2^2] \tag{17}$$

同时，模型参数 $\boldsymbol{\theta},\boldsymbol{\phi}$ 通过 ADAM 优化器更新：

$$\boldsymbol{\theta},\boldsymbol{\phi}=ADAM(\alpha,-(\nabla_\theta\boldsymbol{J}(\boldsymbol{\theta})+\boldsymbol{Y}\boldsymbol{H}(\boldsymbol{\theta})),\boldsymbol{U}\boldsymbol{L}(\boldsymbol{\phi})) \tag{18}$$

其中，α 代表学习率，$\boldsymbol{Y}$ 和 $\boldsymbol{U}$ 两个参数分别代表熵奖励函数和评价网络参数更新的权重。ADAM 默认沿着负梯度方向更新，策略网络需要最大化回报，所以将 $\nabla_\theta\boldsymbol{J}(\boldsymbol{\theta})+\boldsymbol{Y}\boldsymbol{H}(\boldsymbol{\theta})$ 取负，沿着正梯度方向更新；而 $\boldsymbol{U}\boldsymbol{L}(\boldsymbol{\phi})$ 代表值函数的梯度，用于最小化误差，所以不需要进行取负。

6 实验和结果

本文模型及实验均使用 Python 编写，使用的 TSP 数据集与 Joshi 等[12]，Da Costa 等[15]一致，每个数据集包含了 10000 个实例。为方便起见，将包括 20，50 和 100 个城市节点的数据集分别记作 TSP20，TSP50 和 TSP100。在模型训练的过程中，将节点的坐标范围限制在单位正方形 $[0,1]^2$ 之间。

6.1 参数设置

在强化学习中，参数设置具有至关重要的作用，它直接影响到模型的学习效率和性能。下面就折扣因子、学习率等参数的选择与设置进行分析。

折扣因子 λ 可决定模型对未来奖励的关注程度。λ 的取值不仅可以平衡收敛速度与最终效果，还能权衡短期和长期回报的优先级。本文在折扣因子 λ 取不同值后，对比模型的收敛速度(见图 6)。当 λ 为 0.8 及 0.93 时，初始训练误差较低，但训练过程中出现了明显的波动。λ 为 0.88 时，虽然初始误差较大，但在训练过程中波动较小，路径误差更低，为 0.594。因此，本文将 λ 取为 0.88。

在训练过程中，学习率 α 影响训练的收敛速度和稳定性。选择合适的 α 值对于实验至关重要。若 α 过大，初始学习率会导致模型收敛速度过快，使最终结果不稳定；若 α 极小，初始学习率虽然能带来更平稳的训练过程，但会导致较慢的收敛速度。当 λ 取值为 0.88，调整学习率 α 的取值进行收敛速度的对比。如图 7 所示，α 为 0.0003 时，初期收敛速度很快，适合资源有限或需快速初步结果的场景；α 为 0.0001 时，模型初始误差较大，且后续训练过程中波动明显。因此，本文将 α 取为 0.0003。

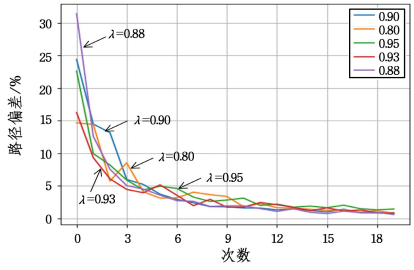

图 6 不同 λ 收敛速度图

Fig. 6 Convergence speed with different λ values

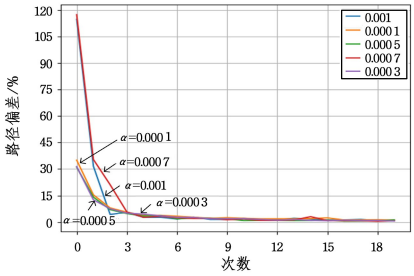

图 7 不同 α 收敛速度图

Fig. 7 Convergence speed with different α values

经过仔细的调整分析，确定在模型训练过程中，各参数的具体设置如表 1 所列。

表 1 参数设置

Table 1 Parameter settings

参数名	值
折扣因子 λ	**0.88**
学习率 α	**0.0003**
更新公式权重 Y	0.0045
更新公式权重 U	0.5
权重 Y 衰减系数	0.9
批次大小 batch_size	512
学习率衰减系数	0.98
节点概率分布参数 C	10

6.2 收敛速度分析

针对数据集 TSP20，运行本文的 2+HRL 模型及 GCN 模型[15]，对比路径偏差率(见图 8)。根据图 8 的折线图，在训练初期，2+HRL 模型相较于 GCN 模型，初期偏差较小，且在后续能够迅速收敛，并最终取得较低的训练误差。这表明 2+HRL 模型能够更迅速地找到 TSP 问题的最优解。

针对数据集 TSP20，TSP50 和 TSP100，运行 2+HRL 模型，得到 200 次迭代训练结果(见图 9)。从图 9 可以看出，2+HRL 模型能够在训练过程中快速减小误差。对于数据集 TSP20，初始误差较小，且误差在前期迅速下降并趋于平稳，最终收敛至 0.029%的训练误差。这表明在处理小规模 TSP 问题时，2+HRL 模型能够以更少的训练轮次快速收敛至高

质量解。对于数据集 TSP50，其初始误差较大，下降速度相对较慢，且在训练过程中波动较大。但最终在 200 次迭代时收敛至 2.47%的最小训练误差。对于数据集 TSP100，初始误差为 84.209%，在 200 次迭代时误差降至 4.897%，训练过程整体平稳。

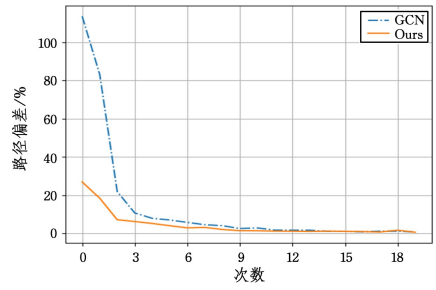

图 8 训练路径前期偏差对比

Fig. 8 Comparison of early-stage path deviation during training

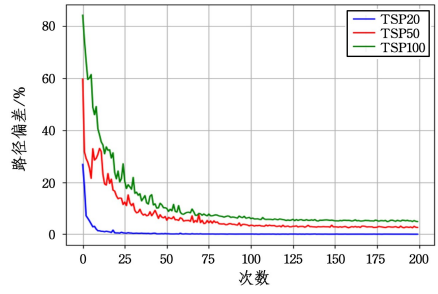

图 9 模型训练误差

Fig. 9 Training errors of the model

6.3 消融实验

将 GAT 模块从 2+HRL 模型中删除，进行消融实验，用于验证 GAT 模块的有效性。实验结果如图 10 所示。

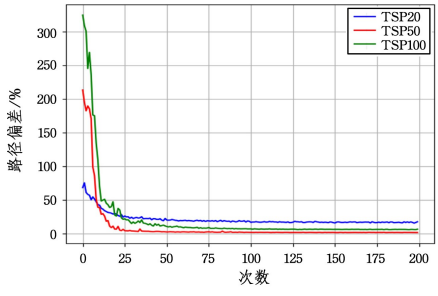

图 10 删除 GAT 模块的训练误差

Fig. 10 Training errors after ablation of GAT

比较图 9 与图 10 的数据，可以看出，将 GAT 模块删除后，模型的初始训练误差变得很大。对于数据集 TSP20，甚至无法收敛到 TSP 问题的优解。这充分说明 GAT 模块能明显提升 2+HRL 模型的性能。

值得注意的是，2+HRL 模型为改进式的模型，即 2-opt 模块的引入能够优化初始解，以更快的速度逼近最优解，因此，2-opt 模块也一定能提升模型的性能。

6.4 性能对比

首先，本文进行了 105 次测试，以验证在较少迭代次数下 2+HRL 模型的准确性。图 11 展示了将训练得到的 2+HRL 模型和 GCN 模型应用于同一 TSP20 数据集测试后的对比结果。

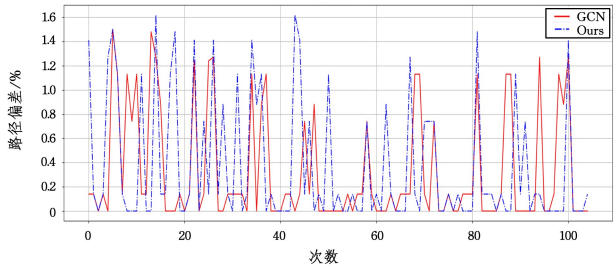

图 11 测试路径偏差百分比

Fig. 11 Percentage deviation of test paths

从图 11 可以看出两次差距最大的测试结果。在第 43 次测试时，2+HRL 模型的预测误差为 0.00%，而 GCN 模型的预测误差为 1.62%；在第 89 次测试时，2+HRL 模型的预测误差为 1.14%，GCN 模型的预测误差为 0.00%。测试结果表明，2+HRL 模型整体更加稳定，波动性小于 GCN 模型。

表 2 给出了 105 次测试数据的统计信息。其中，误差率表示测试得到的 TSP 问题解的长度与最优解的误差，数量代表在 105 次测试中该误差率出现的次数，比例为该误差率在 105 次测试中所占的比重。表 2 的数据显示，2+HRL 模型的最差测试结果比 GCN 模型小 8%。另外，2+HRL 模型在[0.0,0.88]的比例占到 82%，而 GCN 模型占到 75%，这表明 2+HRL 模型的低误差率高于 GCN 模型。

表 2 105 次测试数据

Table 2 Data of 105 test runs

模型	误差率	数量	比例/%	模型	误差率	数量	比例/%
2+HRL	0.0	46	44	GCN	0.0	47	45
	0.14	32	30		0.14	28	27
	0.74	4	4		0.88	3	3
	0.88	4	4		0.74	7	7
	1.13	11	10		1.13	7	7
	1.24	2	2		1.27	2	2
	1.27	4	4		1.41	6	6
	1.48	1	1		1.48	2	2
	1.5	1	1		1.5	1	1
					1.62	2	2
合计	[0.0,1.5] 105		1～44	合计	[0.0,1.62] 105		1～45

另外，从图 12 的箱体图可以看到，2+HRL 模型的平均数为 0.323 低于 GCN 模型的 0.365，表示 2+HRL 模型的整体稳定性优于 GCN 模型。以上均说明，在低训练迭代次数下，2+HRL 模型相比于 GCN 模型能够更为准确的求解 TSP20。

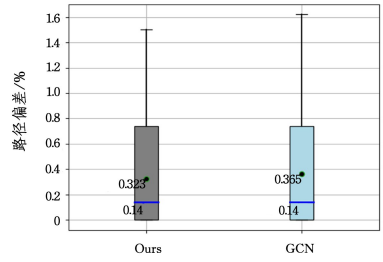

图 12 测试数据箱体图

Fig. 12 Box plot of test data

由于通过增加迭代次数，可以更全面地评估模型在更长时间训练下的收敛情况以及其在处理更复杂问题时的稳定性。并且，随着训练次数的增加，模型能够更好地捕捉到数据的深层次特征，提高解的质量。因此，本文将训练迭代次数设

置为 200，并从两方面进一步探究 2＋HRL 模型的性能。

一方面，运行 2＋HRL 模型与部分传统近似算法，求解 TSP20，TSP50 和 TSP100 的实例，比较各算法所得的目标值、计算误差与计算时间（结果详见表 3）。其中，以 Concorde 求解器得到的最优解作为基准；近似算法选择了最近插入法（Nearest Insertion）、最远插入法（Farthest Insertion）以及随机插入法（Random Insertion）。在最近（或最远）插入法中，首先选择两个距离最近的节点构成初始部分路径，然后从未访问节点中，找到距离当前路径中任意节点最近（或最远）的节点插入当前路径中，直到所有节点都插入路径中为止。而在随机插入法中，选择两个节点构成初始路径，再从未访问节点中随机选择一个节点插入路径中，直到所有节点都被插入路径中。

表 3　与传统算法的数据对比

Table 3　Data comparison with traditional algorithms

模型	TSP20			TSP50			TSP100		
	路径长度	误差/%	时间	路径长度	误差/%	时间	路径长度	误差/%	时间
Concorde[1]	3.84	0.00	1 min	5.700	0.00	2 min	7.76	0.00	3 min
Nearest Insertion	4.33	12.91	1 s	5.800	1.83	2 s	9.46	21.82	6 s
Random Insertion	4.00	4.36	0 s	6.780	19.03	1 s	8.52	9.69	3 s
Farthest Insertion	3.93	2.36	1 s	6.010	5.53	2 s	8.35	7.59	7 s
2＋HRL{500}	3.84	0.00	5 min	5.729	0.52	4 min	7.86	1.32	10 min
2＋HRL{1000}	3.84	0.00	8 min	5.723	0.41	7 min	7.81	0.66	14 min
2＋HRL{2000}	3.84	0.00	12 min	5.700	0.01	12 min	7.80	0.61	16 min

表 3 的计算结果显示，传统近似算法虽然运行时间短，但误差率很高。在时间允许的范围内，如 5 分钟以内，2＋HRL 模型在 TSP20 数据集上能够做到零误差，在 TSP50 数据集上能够做到 0.52%的误差。如果再增加时间步数，2＋HRL 模型在 TSP50 数据集上能将误差降到 0.01%，在 TSP100 数据集中能够将误差降至 0.61%。

另一方面，运行 2＋HRL 模型与部分强化学习算法，求解 TSP20，TSP50 和 TSP100 的实例，比较各算法所得的目标值、计算误差与计算时间（结果详见表 4）。其中，类型一列中 SL 表示监督学习，RL 表示强化学习，S 表示采样，T 表示 2-opt 局部搜索，B 表示束搜索。从表 4 可以看出，时间步数为 500 时，2＋HRL 模型均比 OR-Tools[19]，GCN（Joshi）[12]，S2V[10]的误差低。与 GCN 模型相比，在时间步数同为 500，1000，2000 时，求解数据集 TSP20 所需时间更短、误差也更低。而对于数据集 TSP50，在时间步数为 500 和 1000 时，2＋HRL 模型虽运行时间更短，但在测试误差上略高于 GCN 模型。当时间步数为 2000 时，2＋HRL 模型的预测误差则明显低于 GCN 模型。对于数据集 TSP100，2＋HRL 模型的预测误差在时间步数为 500，1000，2000 时均低于 GCN 模型。这表明 2＋HRL 模型在求解 TSP 问题时的准确率更高。

表 4　与强化学习方法的数据对比

Table 4　Datacomparison with reinforcement learning methods

模型	类型	TSP20			TSP50			TSP100		
		路径长度	误差/%	时间	路径长度	误差/%	时间	路径长度	误差/%	时间
OR-Tools[19]	S	3.85	0.37	1 min	5.80	1.83	5 min	7.99	2.90	—
GCN(Joshi)[12]	SL，B	3.86	0.60	6 s	5.87	3.10	55 s	7.92	2.11	10 min
GCN{500}[15]	RL	3.84	0.01	5 min	5.72	0.36	7 min	7.91	1.84	10 min
GCN{1000}[15]	RL	3.84	0.00	10 min	5.71	0.21	13 min	7.86	1.26	21 min
GCN{2000}[15]	RL	3.84	0.00	15 min	5.70	0.12	29 min	7.83	0.87	41 min
GAT[14]	RL，T	3.85	0.42	4 min	5.85	2.77	26 min	8.17	5.21	3h
S2V[10]	RL	3.89	1.42	—	5.99	5.16	—	8.31	7.03	—
2＋HRL{500}	RL	3.84	0.00	5 min	5.729	0.52	4 min	7.86	1.32	10 min
2＋HRL{1000}	RL	3.84	0.00	8 min	5.723	0.41	7 min	7.81	0.66	14 min
2＋HRL{2000}	RL	3.84	0.00	12 min	5.700	0.01	12 min	7.80	0.61	16 min

总体而言，通过两方面的对比实验，可以表明，2＋HRL 模型可以在测试初期迅速缩小与最优解的差距；同时相比于其他模型，计算误差随着时间步数的增加也能更快降低。由于 2＋HRL 模型只需随机的初始解和采样就能够找到小规模 TSP 问题的最优解或近似最优解，因此，可以将 2＋HRL 模型看作一种小规模 TSP 问题的求解器。

结束语　本文提出一种融合 2-opt 的深度强化学习模型，在求解完全 TSP 问题上更为精确。将 2-opt 与注意力机制结合，使模型能够迅速捕获节点信息，更适用于训练资源有限或需要快速得到解决方案的情况。对于更复杂的 TSP 问题、如非对称 TSP、时间窗约束 TSP，多目标 TSP 等，情况会更为复杂，如何应用 2＋HRL 模型将是未来的研究重点。

参 考 文 献

[1]　APPLEGATE D L，BIXBY R E，CHVÁTAL V，et al. Certification of an optimal TSP tour through 85900 cities[J]. Operations Research Letters，2009，37(1)：11-15.

[2]　COVER T，HART P. Nearest neighbor pattern classification [J]. IEEE Transactionson Information Theory，1967，13(1)：21-27.

[3]　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.

[4]　HELSGAUN K. General k-opt submoves for the Lin-Kernighan TSP heuristic [J]. Mathematical Programming Computation，2009，1：119-163.

[5]　KIRKPATRICK S，GELATT JR C D，VECCHIM P. Optimization by simulated annealing[J]. Science，1983，220(4598)：671-680.

[6]　HOPFIELD J J，TANK D W. “Neural” computation of decisions

in optimization problems[J]. Biological Cybernetics, 1985, 52(3):141-152.
[7] VINYALS O, FORTUNATO M, JAITLY N. Pointer networks[C]// Advances in Neural Information Processing Systems. 2015.
[8] NOWAK A, VILLAR S, BANDEIRAA S, et al. A note on learning algorithms for quadratic assignment with graph neural networks[J]. Stat, 2017, 1050:22.
[9] BELLO I, PHAM H, LE Q V, et al. Neural combinatorial optimization with reinforcement learning[C]// International Conference on Learning Representations. 2017:1-13.
[10] KHALIL E, DAI H, ZHANG Y, et al. Learning combinatorial optimization algorithms over graphs[C]// Advances in Neural Information Processing Systems. 2017.
[11] KOOL W, VAN HOOF H, WELLING M. Attention, learn to solve routing problems[C]// International Conference on Learning Representations. 2018.
[12] JOSHI C K, LAURENT T, BRESSON X. An efficient graph convolutional network technique for the travelling salesman problem[J]. arXiv:1906.01227, 2019.
[13] 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.
[14] DEUDON M, COURNUT P, LACOSTE A, et al. Learning heuristics for the tsp by policy gradient[C]// 15th International Conference Integration of Constraint Programming, Artificial Intelligence, and Operations Research(CPAIOR 2018). Springer, 2018:170-181.
[15] DA COSTA P R, RHUGGENAATH J, ZHANG Y, et al. Learning 2-opt heuristics for the traveling salesman problem via deep reinforcement learning[C]// Asian Conference on Machine Learning. PMLR, 2020:465-480.
[16] SUI J, DING S, LIU R, et al. Learning 3-opt heuristics for traveling salesman problem via deep reinforcement learning[C]// Asian Conference on Machine Learning. PMLR, 2021: 1301-1316.
[17] UDDIN F, RIAZ N, MANAN A, et al. An improvement to the 2-opt heuristic algorithm for approximation of optimal TSP tour[J]. Applied Sciences, 2023, 13(12):7339.
[18] WANG Y, CHEN Z, YANG X, et al. Solving the TSP Problem with Deep Reinforcement Learning Combined with Graph Attention Model[J]. Journal of Nanjing University(Natural Science Edition), 2022, 58(3):420-429.
[19] PERRON L, FURNON V. Or-tools[EB/OL]. https://developers.google.com/optimization/.

PENG Junlong, born in 2000, postgraduate. His main research interests include deep reinforcement learning and combinatorial optimization.

FAN Jing, born in 1979, associate professor. Her main research interests is optimization algorithms.