中山大學

SUN YAT-SEN UNIVERSITY

# 利用有监督和强化学习联合方法解决VRP问题

唐瑞怡 **18340159**

张琛颐 **18340205**

# 目录

# 摘要

在这次实验中，需要复现论文《Efficiently solving the practical vehicle routing problem: A novel joint learning approach》中的方法来解决 VRP 问题，也就是利用图卷积网络对 VRP 问题中客户坐标和客户距离信息进行编码，然后利用序列解码器和分类解码器来进行解码，以预测出 VRP 问题中的路径。序列解码器主要利用强化学习方法进行优化，并将序列解码器的的输出作为分类解码器的标签来对分类解码器进行有监督学习，通过这种联合的学习方法，可以找到 VRP 问题中一个比较优的解。


# 1 导言

## 1.1 VRP 问题

VRP 问题是一个调度任务，描述了这样一个场景：有多个客户对货物有需求，有一个仓库可以提供货物，需要调度一或多辆车来向客户送货。每一辆车都有自己的容量，车辆的行驶具有代价，当车的剩余容量不足以满足客户需求时，需要返回仓库来装载货物。问题目标是找到满足所有客户需求的最小代价解。这个问题是一个 NP-hard 问题，要求解这个问题的最优解是非常困难的，因此希望通过深度学习、强化学习的方法来找到近似最优解。

# 2 实验过程

## 2.1 VRP 问题定义

### 2.1.1 问题描述

在 VRP 问题中，可以将运送货物看出是一个图模型 $G = (\mathcal{V}, \mathcal{E})$，将客户和仓库看成点，设点集为 $\mathcal{V} = \{0, 1, ..., n\}$，其中 0 表示仓库，$i \in \{1, ..., n\}$ 表示客户。在点集的基础上可以定义边集 $\mathcal{E} = \{e_{ij}\}$，其中 $i, j \in \mathcal{V}$。对于顾客点 $i$ 来说，都有坐标 $x_i^c$ 和需求 $x_i^d$ 这两个特征，因此每一个点的特征都可以表示为 $x_i = \{x_i^c, x_i^d\}$；对于仓库点来说，就只有坐标这个特征了。在这个图中，边与距离这个特征相关，点 $i$ 和点 $j$ 之间的距离可以表示为 $m_{ij}$。在这个图中，边与边之间是全连接的，但考虑到实际交通的情况，应该设 $m_{ij} \neq m_{ji}$。

在这个问题中，希望可以找到一组访问序列 $\pi = (\pi_1, \pi_2, ..., \pi_T)$，其中 $\pi_t \in \{0, 1, 2, ..., n\}$，让代价最小。这个序列要满足：（1）序列从 0 开始并以 0 结束；（2）每一个客户都只能访问一次，但是仓库可以访问多次。为了让代价最小，可以定义下面的目标函数：

$$min\, c_v Q_v + c_t \sum_{t=1}^{T-1} m_{\pi_t \pi_{t+1}} \tag{1}$$

其中 $c_v$ 是每一辆车的固定损耗代价，$Q_v$ 是车辆数，$c_t$ 是行驶的单位代价。

## 2.2 模型框架

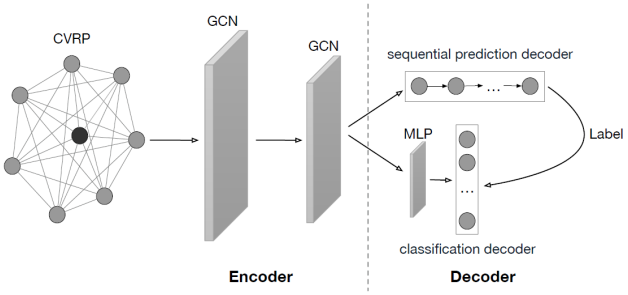



**FIGURE 1:** 模型整体框架

模型整体上利用了**编码器-解码器**模型，使用了 GCN 网络对图上的信息进行编码，使用两个解码器序列解码器和分类解码器进行解码。在序列解码器中，它根据 GCN 中对点信息的编码，利用 GRU 网络和注意力机制来预测出路径序列 $\pi$。在分类解码器中，它根据 GCN 中对边信息的编码，利用 MLP 网络进行解码。两个解码器相互补充，相辅相成：序列解码器的结果作为分类解码器的标签，让分类解码器可以进行有监督的训练；当有序列解码器的效果变好之后，可以提升分类解码器的效果，同时在边特征的帮助下，序列解码器的效果也可以得到提升。

## 2.3 输入层 (Input Layer)

这个部分将点的信息和边的信息进行向量表示。

- **点信息**

  对于点来说，有坐标和距离的特征，可以通过一个全连接层和激活函数，将这些信息用一个向量来表示，方法如下式：（由于仓库没有需求这个特征，因此只需要表示出坐标的信息就可以了）

$$x_i(x) = \begin{cases} Relu(W_1 x_{c_0} + b_1), & if\ i = 0 \\ Relu([W_2 x_i^c; W_3 x_i^d + b2]), & if\ i \geq 0 \end{cases} \tag{2}$$

  其中 $W_1, W_2, W_3, b_1, b_2, b_3$ 都是可训练参数。

- **边信息**

  对于输入的每一条边 $e_{ij}$，有一个特征 $y_{ij}$，它包含距离和连通 ($a_{ij}$) 的信息, 设特征的维度是 d y 。原来的图是全连接的，但是为了计算方便，只允许 K 临近是连通的。如果节点 $i$ 和 $j$ 是 K 临近的，就设 $a_{ij}$ 为 1，如果 $i == j$ 将 $a_{ij}$ 表示为-1，其他情况就设将 $a_{ij}$ 为 0。这样，就可以表示出边的嵌入了，如下式：

$$y_{ij} = Relu([W_4 m_{ij} + b4; W_5 a_{ij} + b_5]) \tag{3}$$

  其中 $W_4, W_5, b4, b5$ 都是可训练的参数。

## 2.4 图神经网络 Graph Convolutional Network (GCN)

图神经网络的作用是根据输入层的输出，也就是关于点和边的特征向量，进行编码操作。在编码之前，需要先计算点和边的初始嵌入表示 $h_i^0$ 和 $h_{e_{ij}}^0$：

$$\begin{aligned} h_i^0 &= W_{E1}x_i + b_{E1}, \\ h_{e_{ij}}^0 &= W_{E2}y_{ij} + b_{E2} \end{aligned} \tag{4}$$

其中 $W_{E1}, W_{E2}, b_{E1}, b_{E2}$ 都是可训练参数。然后可以经过图神经网络进行编码，这里使用的图神经网络，同时关注了点和边的信息，并且让它们可以同时更新。

在图神经网络中，对于每一层都包括聚合子层和连接子层：

- **聚合子层**

  聚合子层的作用是将点或者边的邻居信息聚合起来，让网络的输出可以融入更多图上的信息，具体操作如下：

  对于点来说，需要对邻居点的特征进行聚合，设当前节点 $i$ 的邻居集合为 $N(i)$，聚合之后第 $l$ 层的嵌入为：

  $$h_{N(i)}^i = \sigma(W_I^l(ATTN(h_i^{l-1}, \{h_{v'}^{l-1}, \forall v' \in N(i)\}))) \tag{5}$$

  其中 $W_I^l$ 是可训练参数。

  在这个实验中，我们使用了缩放点击模型来计算 attention。

  对于边来说，一条边 $e_{ij}$ 只与节点 $i$ 和 $j$ 相关联，因此只需要聚合这两个节点的信息就可以了，因此聚合之后的第 $l$ 层嵌入为：

  $$\begin{aligned} h_{N(e_{ij})}^l &= \sigma(W_E^l AGG_E^l(\{h_{e_{ij}}^{l-1}, h_i^{l-1}, h_j^{l-1}\})) \\ AGG_E^l(\{h_{e_{ij}}^{l-1}, h_i^{l-1}, h_j^{l-1}\}) &= W_{e_1}^l h_{e_{ij}}^l + W_{e2}^l h_i^l + W_{e3}^l h_j^l \end{aligned} \tag{6}$$

  其中 $W_{e_1}^l, W_{e_2}^l, W_{e_3}^l$ 都是可训练参数。

- **连接子层**

  经过聚合子层，可以得到融入了邻居的嵌入表示，在连接子层，通过对 $l-1$ 的嵌入表示和邻居嵌入表示进行联合，来生成当前 $l$ 层的嵌入表示：

  $$\begin{aligned} h_i^l &= [V_I^l h_i^{l-1}; h_{N(i)}^l] \\ h_{e_{ij}}^l &= [V_E^l h_{e_{ij}}^{l-1}; h_{N(e_{ij})}^l], \end{aligned} \tag{7}$$

  其中 $V_I^l$ 和 $V_E^l$ 都是可训练参数。

## 2.5 序列解码器 (Sequential Decoder)

序列解码器是根据图神经网络的输出，进行解码，生成一个可行的访问序列 $\pi = \{\pi_t, t = 1, ..., T\}$。在解码器中，使用一个 GRU 网络结构和基于上下文的注意力机制。由于每次访问节点和之前的访问的节点都紧密相关，使用 GRU 网络，就可以利用这个关系，根据前面的时刻预测出来的值，进而来预测出当前访问的节点。

在这个解码器中，对于输入 S，希望极大化后验概率 $P(\pi|S;\theta)$，这个概率可以通过链式法则来进行计算：

$$P(\pi|S;\theta) = \prod_{t=0}^{T} p(\pi_{t+1}|z_t, f(S,\theta_e), \pi_t; \theta_d), \tag{8}$$

其中 $f(S,\theta_e)$ 是编码器的输出，$\theta_d$ 是一个可训练参数。上式中引入了一个参数 $z_t$，它是 GRU 网络的隐状态，加上这个部分，可以当前时刻的输出考虑到前面时刻的输出值。

在解码过程中，预测当前应该访问哪个节点，需要考虑到当前小车的剩余容量和客户的需求，因此每次选择当前节点时，都应该对每个客户点进行考虑，如果不满足约束条件，就应该不考虑访问该点。约束条件为：

1. 对每一个客户节点，只能访问一次
2. 每次访问客户节点时，必须保证当前小车的剩余容量大于客户需求
3. 不能连续访问仓库

根据上面的约束规则，可以在每一个时刻 $t$，计算出不能访问节点的集合 $N_{mt}$。根据 $N_{mt}$ 可以为每一个点算出一个权重 $u_{ti}$，这个权重经过 softmax 之后，就可以表示为当前时间，每一个节点会被选择的概率。权重 $u_{ti}$ 的计算公式如下：

$$x_i(x) = \begin{cases} -\infty, & \forall j \in N_{mt} \\ h_a^T tanh(W^G[v_i; z_t]), & otherwise \end{cases} \tag{9}$$

## 2.6 分类解码器 (Classification Decoder)

分类解码器是希望根据边的信息，使用有监督的方法来进行学习，解码出一个矩阵 $p_{e_{ij}}^{VRP}$，矩阵第 $i$ 行，第 $j$ 列表示，当前点是 $i$ 时，下一个点预测为 $j$ 的概率。分类解码器使用一个 MLP 网络，由于不知道问题的最优解是什么，因此可以将序列解码器的输出来作为分类解码器的标签来进行训练。

设 GCN 网络对边信息的嵌入为 $h_{e_{ij}}^L$，利用 MLP 网络和 softmax 函数可以计算出每一条边 $e_{ij}$ 被选择的概率：

$$p_{e_{ij}}^{VRP} = softmax(MLP(h_{e_{ij}}^L)) \in [0,1]^2 \tag{10}$$

## 2.7 整体模型 loss

由于这个模型中有用强化学习对序列解码器进行训练，用有监督学习对分类解码器进行训练，需要将这两个方法的 loss 结合起来。

对于强化学习来说，loss 函数定义如下：

$$\begin{aligned} L_r(\theta_e, \theta_d) &= \sum_S E_{\pi \sim P(\pi|S;\theta_e,\theta_d)}(L(\pi) - b(S)) \\ &= \sum_S ((L(\pi) - b(S))) \sum_{i=1}^{T} logp(\pi_i|\pi_{i-1}, S; \theta_e, \theta_d) \end{aligned} \tag{11}$$

其中，$L(\pi)$ 是解 $\pi$ 的代价，$b(S)$ 是利用贪婪算法求出来的解对应的代价，这里利用贪婪算法作为基准，认为用贪婪算法求出来的解是比较优的。根据模型的运行，需要对使用贪婪算法的模型参数进行适时更新，使用 t-test 可以对根据当前序列解码器的结果来确定什么时候对贪婪算法模型进行更新。

对于有监督学习来说，使用交叉熵作为 loss：

$$L_s(\theta_e, \theta_c) = -\sum_{S,\pi} crossEntropy(P_E^{VRP}, P_E^{VRP*}) \tag{12}$$

其中 $P_E^{VRP}$ 是使用分类解码器的输出，$P_E^{VRP*}$ 是序列解码器的输出。

最后可以得到整体模型的 loss 为：

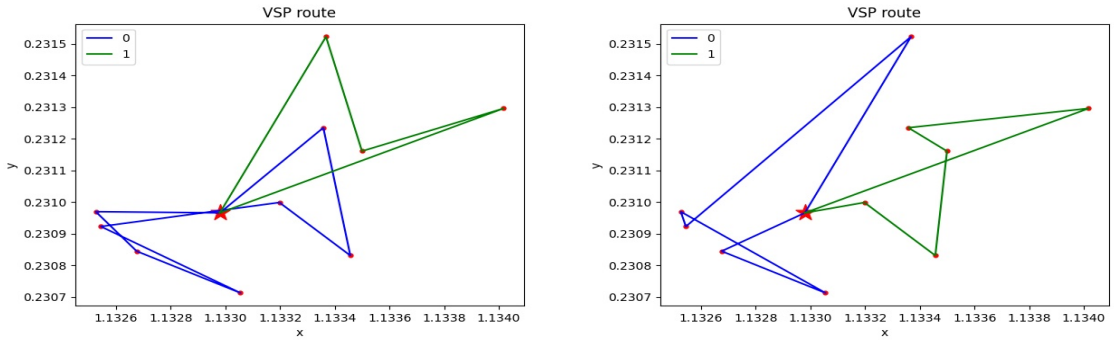
$$L_\theta = \alpha \times L_s(\theta) + \beta \times L_r(\theta) L_\theta = \alpha \times L_s(\theta) + \beta \times L_r(\theta) \tag{13}$$

# 3 结果分析

## 3.1 车辆运输路径变化图

以下呈现在迭代次数为 60 的情况下，在 **G-20-training** 数据集和 **G-20-testing** 数据集下路径变化情况。

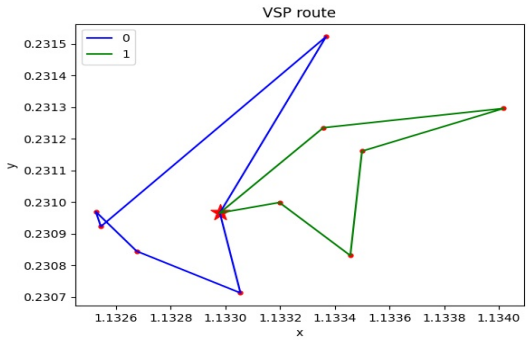
**G-20-training** 在该数据集下，提取前 10 个点（即满足 10 个客户的需求）进行模型的训练。以两个例子为例。



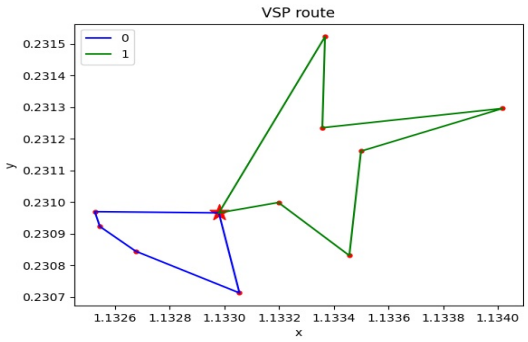

(a)G20:01:01




(b)G20:01:02

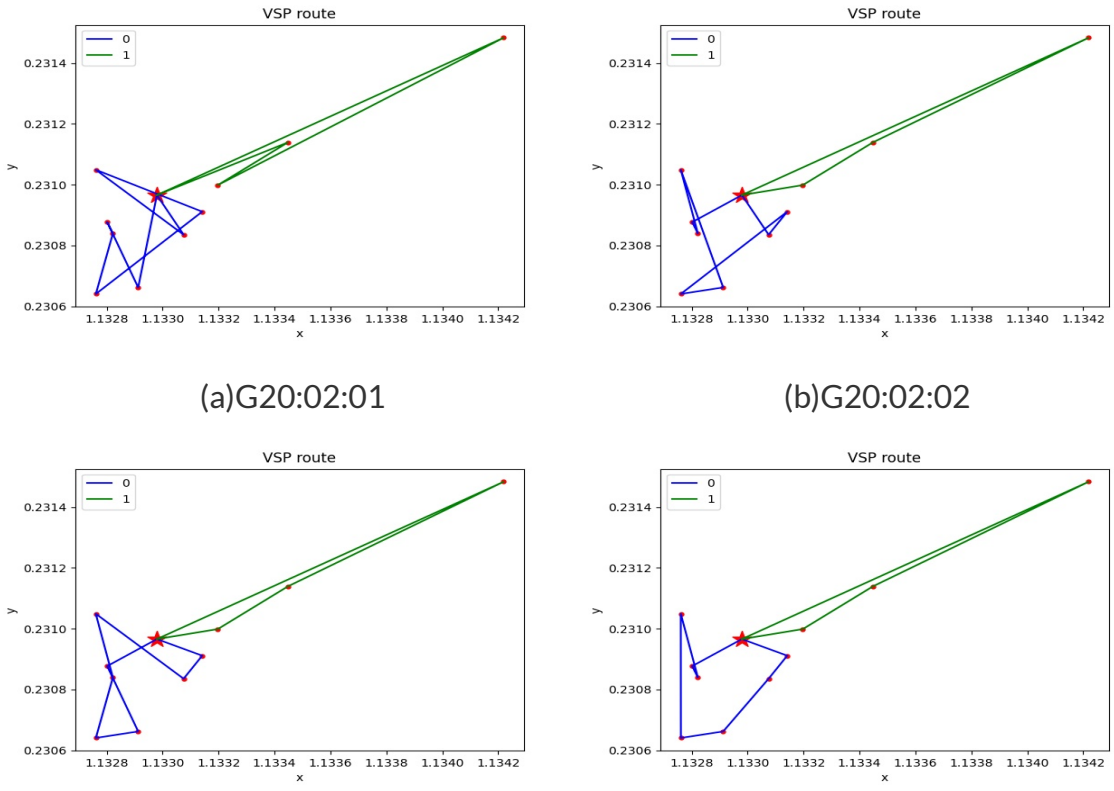



(c)G20:01:03




(d)G20:01:04

**FIGURE 2:** G20-training:01



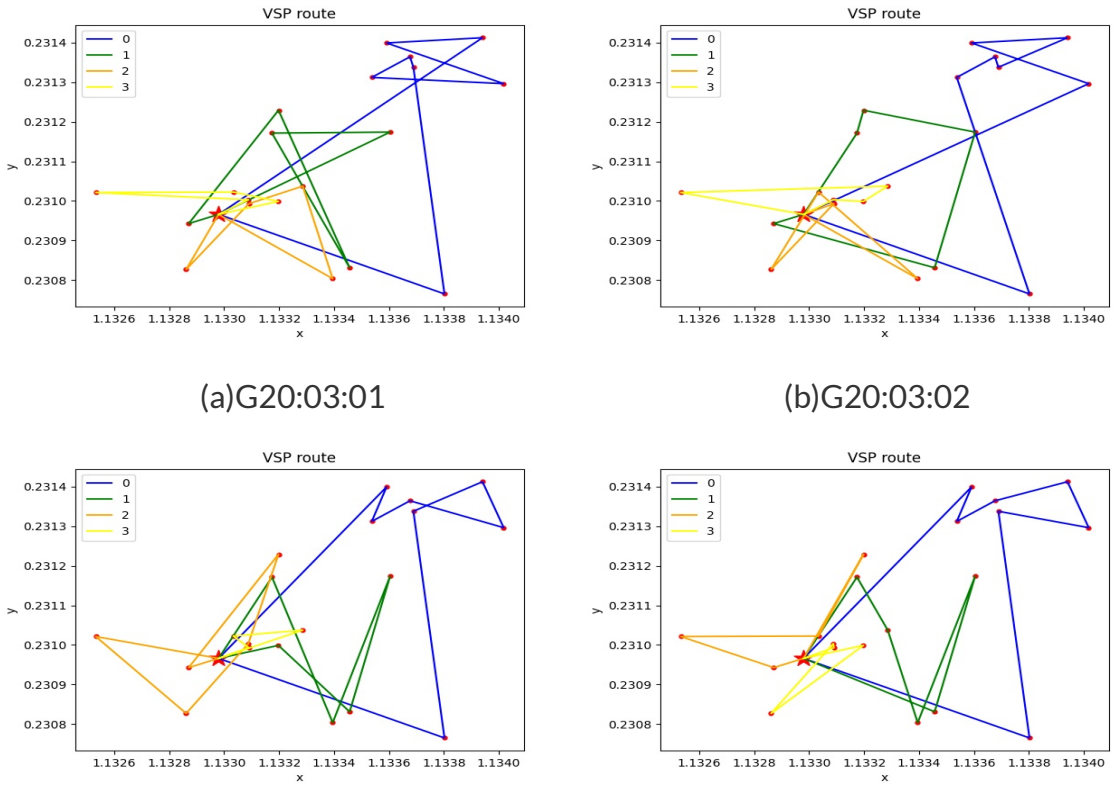

(a)G20:02:01 (b)G20:02:02

(c)G20:02:03 (d)G20:02:04

**FIGURE 3:** G20-training:02

从上面两个例子可以看出，在训练集上，随着迭代次数的增加，效果明显提升。具体表现为车辆的总路程在变小，路径交叉程度也在变小，说明了此联合方法确实可以较好地解决 VRP 问题。

**G-20-testing** 在该数据集下，提取 20 个点（即满足所有客户的需求）进行模型的预测验证。以下面的例子为例分析。




(a)G20:03:01 (b)G20:03:02

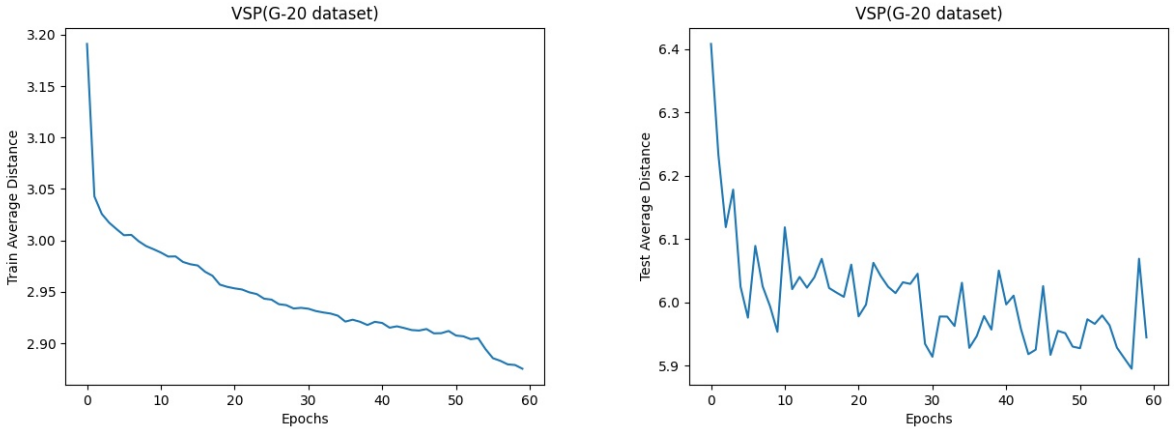
(c)G20:03:03 (d)G20:03:04

**FIGURE 4:** G20-testing:01

从此例子可以看出，在测试集上预测效果一般，在中间部分仍有路径交叉情况，但随着迭代次数的增加，路径交叉情况和路径长度有变好的趋势。

## 3.2 模型收敛曲线

以下呈现在迭代次数为 60 的情况下，在 **G-20-training** 数据集和 **G-20-testing** 数据集下平均路径长度的变化情况。




(a)Train (b)Test

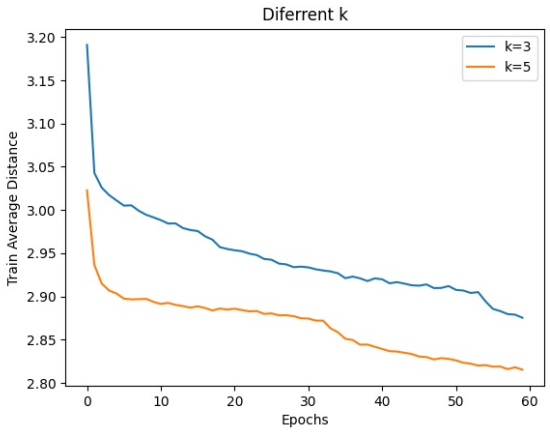
从上图可看出，在训练集中，平均路径长度随着迭代次数的增加稳步下降，且还有下降的空间。而在测试集中，平均路径长度随着迭代次数的增加波动下降，与训练集趋势相近，表明联合模型的效果较好。

## 3.3 参数分析

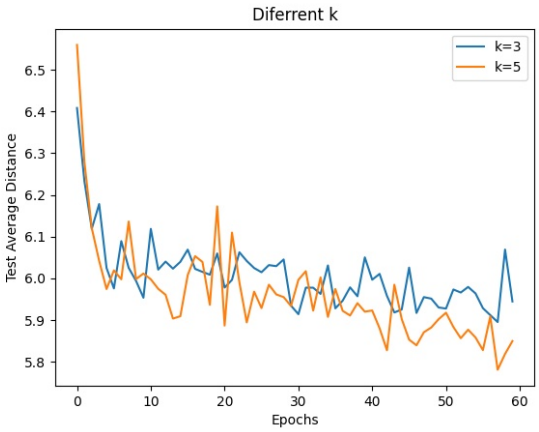
本小节介绍本实验中以下几个参数的意义和对模型的影响。

### 3.3.1 k 近邻中的 k

$k$ 参数用于求一个节点的邻居节点。一个点周围最近的 $k$ 个点作为其邻居。




(a)Train

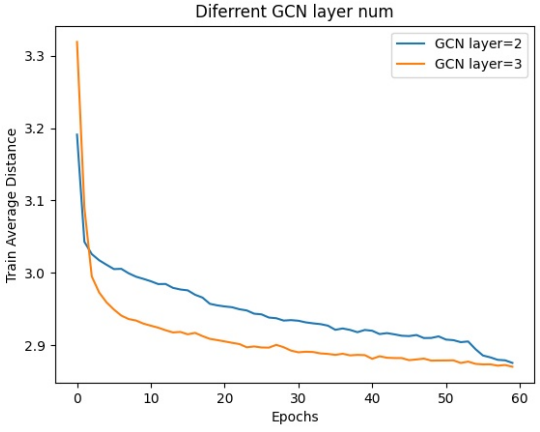



(b)Test

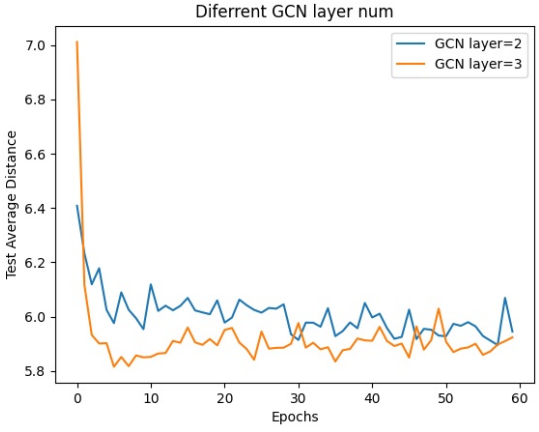
由上面图像可以看出，随着 k 的增加，模型在训练集和测试集上的效果都明显变好。这是因为 k 变大时，可以聚合更多邻居节点的信息，有利于数据特点的提取和模型参数的训练。

### 3.3.2 GCN 层数

GCN 网络结构参数网络结构的参数控制了点边信息编码的次数。



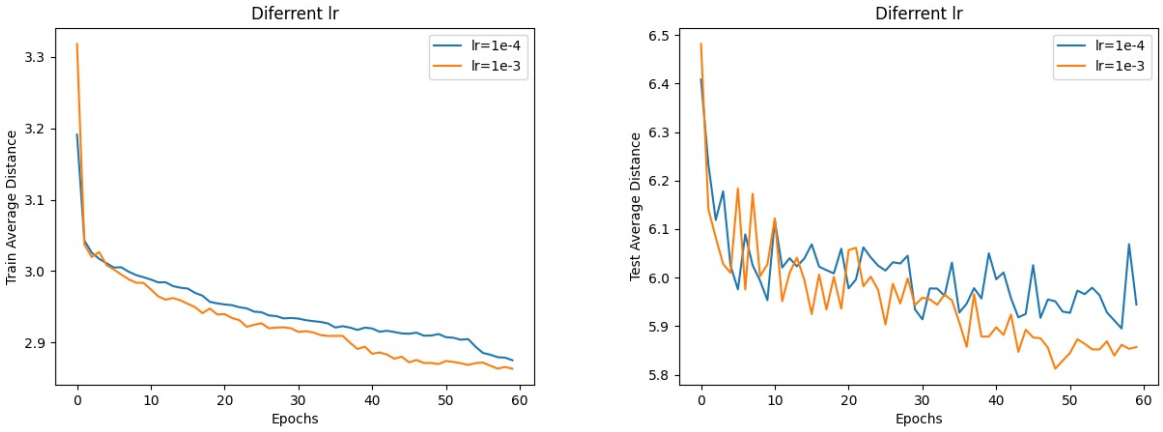

(a)Train




(b)Test

由上面图像可以看出，随着 GCN 层数的增加，模型在训练集和测试集上的效果都明显变好。这是因为 GCN 层数变大时，可以编码更多的点边信息，提取更多的问题特征。

### 3.3.3 学习率




(a)Train (b)Test

由上面的图像可以看出，随着学习率的增大，模型收敛速度提高。一开始我们听取了大家的经验，使用了较小的学习率进行实验。后来发现学习率的设置对于本模型影响不大，提高学习率可以提高模型收敛速度。

### 3.3.4 各参数效果汇总

**TABLE 1:** 各参数效果汇总

| 参数 | 测试集最短路径 |
|---|---|
| k=3, lr=1e-4, GCN=2 | 5.895 |
| k=5, lr=1e-4, GCN=2 | 5.781 |
| k=3, lr=1e-4, GCN=3 | 5.817 |
| k=3, lr=1e-3, GCN=2 | 5.769 |
| Or-tools | 3.783 |

总体来说，我们的模型对于 VSP 求解问题进行了很多的优化，但是和 Or-tools 的结果还是有一定差距，可能是因为训练时间不够造成的，可以继续训练来找到模型更优的结果。

# 4 结论与思考

1. 为什么要设置两个解码器？

   在这个模型中，使用两个解码器来对分别利用点的信息和边的信息进行解码，序列解码器的输出作为分类解码器的标签，让分类解码器可以进行有监督的学习。为什么这这样设置呢？从模型结构上说，序列解码器的输出成为分类解码器的标签，为什么不能只用一个序列解码器，用强化学习进行训练呢？但实际上，分类解码器也是有益于序列解码器得到更好的结果的。因为在分类解码器中，进行梯度回传时，会让 GCN 网络提取出边的信息，形成的嵌入可以更好的表征边。而在对点信息进行编码时，聚合了边的信息，这样就能生成更好的点嵌入表示，进而有效提高序列解码器的效果。

2. 为什么序列解码器使用了 GRU 网络结构，而分类解码器使用了 MLP 网络？

   在序列解码器中，GRU 网络是一个循环神经网络结构，可以结合路径中序列的顺序，之前预测出来的点来预测新的点。GRU 网络的结构，非常符合这个问题的情景。那为什么分类解码器要用 MLP 网络呢？我觉得一方面是因为需要让点信息和边信息都可以作用到模型中，使用点信息的预测结果来指导边信息来预测，使用这样一种有监督的学习方法可以让边信息和点信息有机交融。另一方面是 MLP 网络结构简单，也可以提取出很多有效信息，使用 MLP 网络可以提高模型训练速度。

3. 为什么要用深度学习的方法来解决 VRP 问题？

   VRP 问题是一个 NP-hard 问题，想找到真实的最优路径是不现实的。总的来说，目前解决这个问题有主流两个方向，一个是使用数学模型求解，另一个是使用深度学习方法。使用数学模型是基于启发式算法，来找到模型的近似最优解，这个方法找到的解质量比较高，但是这些算法本身的复杂度很大，难以在实际中使用。因此，用深度学习的方法越来越受到关注，它模型自己学习算法，通过在计算时加上约束，就可以找到合理的解。虽然用这种方法找到的解不一定很好，甚至会出现效果很一般的解，但模型一旦训练好，对于未知的数据来说，可以很快找到解，在实际应用中有很大的价值。

4. 在序列解码器中，如何实现分 batch 训练？

   由于一个 batch 中，每一个样例的访问序列总长度都不一样，为了能进行分 batch 训练，需要对样例进行对齐，也就是只有最长的序列被预测出来之后，才可以停止序列解码器的解码操作。在最长序列没有被预测之前，对于已经终止的样例来说，我们设置这个样例不断访问仓库，让整个 batch 的结果都是合法解。

# 5 参考文献

[1] Duan,L.,Zhan,Y.,Hu,H.,Gong,Y.,Wei,J.,Zhang,X.,Xu,Y.:Efficiently solving the practical vehicle routing problem: A novel joint learning approach. In:KDD. pp.3054–3063 (2020)