中国科学：数学　2024 年　第 54 卷　第？期：1～26
SCIENTIA SINICA Mathematica
论　文
《中国科学》杂志社
SCIENCE CHINA PRESS

胶囊聚合注意力机制求解车辆路径规划问题

师瑞阳[1,2,3]，牛凌峰[2,3,4*]，戴彧虹[5]

1. 中国科学院大学数学科学学院, 北京 100049;
2. 中国科学院虚拟经济与数据科学研究中心, 北京 100190;
3. 中国科学院大数据挖掘与知识管理重点实验室, 北京 100190;
4. 中国科学院大学经济与管理学院, 北京 100190;
5. 中国科学院数学与系统科学研究院计算数学与科学工程计算研究所, 北京 100190

E-mail: shiruiyang18@mails.ucas.ac.cn, niulf@ucas.ac.cn, dyh@lsec.cc.ac.cn

收稿日期: 2023-05-19; 接受日期: 2024-01-16;　*通信作者
国家自然科学基金 (批准号: 11991021, 11991020 和 12271503) 资助项目

摘要　近年来, 深度学习在求解车辆路径规划问题 (vehicle routing problem, VRP) 中展示出巨大潜力. 注意力机制在其中发挥着重要作用, 已成为提高求解质量的关键模块, 但其加和聚合范式不足以充分捕捉 VRP 实例中丰富的信息. 针对该问题, 本文提出一种新的胶囊聚合注意力机制. 它使用胶囊存储更多的信息, 应用动态路由机制进行信息聚合, 利用软门控胶囊选择器来区分不同胶囊的重要性, 并修改解码过程中的上下文节点向量以反映状态变化. 基于上述改进, 本文设计了一种强化学习框架下求解 VRP 的新型图注意力网络, 并通过大量数值实验证明了所提出方法的有效性.

关键词　车辆路径规划问题　胶囊网络　注意力机制

MSC (2020) 主题分类　65Z05, 68T20, 68Q32, 90C35

1　引言

车辆路径规划问题 (vehicle routing problem, VRP) 是一类经典的组合优化问题 [59], 在现实世界中应用非常广泛. 此类问题可定义为: 车队从配送中心取货出发, 在满足客户需求和约束的限制下, 确定成本最优的路径集合 (参见文献 [38]). 在不同的约束条件下, VRP 衍生出了许多变体, 如旅行商问题 (traveling salesman problem, TSP)、容量受限 VRP (capacitated VRP, CVRP) 和分割交付 VRP (split delivery VRP, SDVRP) 等. 由于 VRP 的 NP (non-deterministic polynomial) 难特性, 其求解过程非常具有挑战性, 因此如何得到令人满意的解仍然值得探索 (参见文献 [42]).

20 世纪 50 年代, Dantzig 等 [16] 首次将 TSP 建模为整数线性规划进行求解, 取得了历史性的突破. 此后, 研究人员一直致力于寻找不同类型的有效方法来求解 VRP. 目前, 求解 VRP 的方法一般分为精确方法 [3, 6, 13, 17, 39, 40] 和启发式方法 [5, 12, 14, 23, 24, 28, 54] 两大类. 精确方法主要基于动态规

英文引用格式: Shi R Y, Niu L F, Dai Y-H. Capsule aggregated attention for vehicle routing problems (in Chinese). Sci Sin Math, 2024, 54: 1–26, doi: 10.1360/SCM-2023-0307

划 [13]、分支定界 [39,40]、集划分和列生成 [3,6,17]. 此类方法适用于解决小规模的 VRP, 但由于其指数复杂性, 很难扩展到更大规模的 VRP 实例 [40]. 启发式方法主要包括 Clarke-Wright 算法 [14]、Sweep 算法 [24]、插入算法 [28]、禁忌搜索算法 [23] 和遗传算法 [5] 等. 它们能够在可接受的多项式时间内获得可行解或次优解 [28], 然而这些方法高度依赖于专业知识和人类专家经验, 这导致在每个特定领域中都需要花费大量的研究时间 (参见文献 [34]).

在过去的十年里, 深度学习的快速发展 [41] 为求解 VRP 带来新的启发 (参见文献 [63]), 因此许多基于学习的方法 [18,32,46,47,62,65] 被提出. 与传统的启发式方法相比, 基于学习的方法可以在无需推导新的显式算法的情形下, 以通用的方式快速构建输入实例与输出解之间的近似映射关系 [8]. 通常, 根据求解过程的模式, 基于学习的方法可以分为两类, 即自构建方法和学习改进启发式方法 [56]. 自构建方法训练模型以逐节点的方式直接从输入实例得到输出解. 近年来, 许多基于学习的方法都属于自构建的范畴 (参见文献 [7,18,36,37,46,49,67]). 特别地, Kool 等 [36] 设计了基于多头注意力的编码器, 并应用了基于自注意力的解码器, 这被认为是目前最具代表性的自构建方法之一. 此外, Kwon 等 [37] 以 TSP 为例分析了基于强化学习方法中的对称性, 并在训练过程中通过基于文献 [36] 的模型并行多个轨迹; Bi 等 [9] 设计了一种在多个不同示例分布下的自适应多分布知识蒸馏框架; Yuan 等 [67] 在文献 [36] 的模型架构下, 设计了一种结合强化学习和自监督对比学习的两阶段方法.

而对于学习改进启发式方法 [32,47,65], 训练后的模型会辅助一类特定的改进算子, 如为局部搜索算子、破坏修复算子等选择节点对. 整个优化过程与传统的启发式方法类似, 都是迭代地改进解. 而与自构建方法相比, 学习改进启发式方法通常可以获得质量更好的解, 但计算的时间成本更高 (参见文献 [56,65]). 此外, 如何更好地将传统的启发式算子融入到基于学习的模型中也是一项困难的任务 (参见文献 [56]). 具有代表性的学习改进启发式方法包括 Hottung 和 Tierney [32] 使用基于注意力机制的策略网络来求解 CVRP, 以辅助大规模邻域搜索; Wu 等 [65] 提出了一种基于 Transformer 架构 [61] 的策略网络来指导节点对的选择, 并采用了 2-OPT (optimization) 算子来改进解; Ma 等 [47] 改进了文献 [65] 中的模型, 设计了双向协同注意力和循环位置编码, 有效地利用了注意力相关性、解的循环性和对称性. 可以看出, 许多基于学习的方法都采用了注意力机制.

注意力机制对提高求解 VRP 的质量有很大帮助, 它允许模型动态地关注节点序列的某些部分, 使其能够选择与当前状态最密切相关的上下文作为输入 (参见文献 [62]). 这有助于模型在提取出的隐含信息的指导下, 为 VRP 实例正确地选择下一个要访问的节点. 一般地, 注意力机制提取的信息越准确, 模型就越容易求出高质量的解. 然而, 现有的基于学习的 VRP 模型中使用的注意力机制的表达能力十分有限. 以文献 [18,36] 中使用的多头注意力为例, 虽然其在不同的子空间中能够独立地生成多个表示 [51], 从而避免只投影到一个唯一的子空间来避免模型性能降低的风险, 但直接的串联和线性变换操作并不足以充分捕捉 VRP 实例中的丰富信息 (参见文献 [43]). 因此, 为了提高模型的整体表现, 如何增强注意力的信息捕捉和表达能力是值得研究的.

胶囊网络 [55] 是近年来新提出的一种高级网络架构, 已经成功应用于多个领域 [1,20,21,69]. 胶囊网络中的基本胶囊层由胶囊映射和动态路由两部分组成. 每个输入胶囊先被线性投影到更高层次的空间中, 然后相邻层中的胶囊通过动态路由相互交流. 动态路由机制使胶囊网络具有强大的信息聚合能力, 并能更好地处理部分与整体之间的关系 (参见文献 [43]). 因此, 本文为改进信息聚合方式, 将利用胶囊网络中的动态路由机制, 对不同头部的注意力关系进行显式建模, 进而提出胶囊聚合注意力机制. 具体而言, 输入胶囊根据不同注意力头的输出建立, 并为保持其原始特性, 用矩阵代替向量表示. 为度量输入胶囊和输出胶囊之间信息转换的比例, 本文为每个输入胶囊生成一个特有的投票, 然后计算输出胶囊和投票之间的余弦相似度. 为拟合胶囊的矩阵表达形式, 本文基于公式推导修改动态路由机制.

最后, 所有输出胶囊被封装并连接在一起得到最终表达.

与多头注意力相比, 胶囊聚合注意力具有更强的信息捕获和聚合能力. 然而, 平等对待每个胶囊将埋没那些潜在的重要胶囊, 并失去从中受益的机会. 换而言之, 存在一些不重要的胶囊可能会误导模型的最终预测. 因此, 受文献 [50, 68] 中研究的启发, 本文设计了一种软门控胶囊选择器过滤掉不重要的信息, 以减小聚合过程中无用表示的影响. 具体地, 为避免引入过多参数和增加模型的复杂度, 本文使用线性映射及非线性激活来计算每个输出胶囊的软门控值, 来区别不同胶囊的重要性. 此外, 清楚地反映系统状态变化对于指导求解 VRP 的行动选择非常重要. 在基于注意力的模型中, 上下文节点向量在这方面起到关键作用. 例如, 在文献 [36] 中, 上下文节点向量由图嵌入的信息、最后一个节点和第一个节点组成. 尽管这种设计非常直观, 但仅记录这两个节点不足以全面展示整个系统状态的变化. 因此, 通过添加未访问节点的显式信息设计了能够动态感知的上下文节点向量.

除模型设计之外, 训练框架对于学习类方法也很重要. 在监督学习框架下训练模型需要使用精确求解器获得非常多的高质量 VRP 实例, 这需要很高的时间成本. 与监督学习的方式不同, 强化学习命令智能体通过与环境交互来学习求解 VRP 的最优策略, 以此来避免这种困境 (参见文献 [63]). 因此, 我们在这项工作中采用强化学习来训练我们的模型. 从广义上讲, 强化学习算法可以分为基于模型与模型自由两类 [2]. 基于模型的强化学习算法试图理解整个环境并据此作出决策. 而模型自由的强化学习算法不依赖于环境, 仅利用智能体收集的经验进行决策. 本文选择更简单直观的模型自由方式, 并采用策略梯度方法对模型进行训练. 最终, 本文提出一种基于强化学习框架的胶囊聚合图注意力网络 (capsule aggregated graph attention networks, CapsGAT) .

为测试所提出模型的性能, 本文在 TSP 和 CVRP 两个经典的 VRP 上进行了数值实验. 首先将 CapsGAT 与若干 VRP 求解器进行了比较. 结果表明, 在大多数情形下, 本文的模型优于启发式基线和基于学习的基线, 并进一步缩小了与最优解的差距. 其次, 本文比较了 CapsGAT 和其他学习类求解器在不同规模实例、不同分布实例和多个公共数据集上的泛化性能. 结果表明, 本文的模型在大多数情形下都具有更强的泛化能力, 并且这种优势在问题规模增大时变得更加明显. 然后, 从两个角度进行了消融实验, 包括模型不同组件之间的比较, 以及胶囊聚合注意力和全连接聚合注意力之间的比较. 实验结果证实了胶囊聚合注意力的有效性. 最后, 对 TSP 实例上的节点选择过程进行了可视化. 相比于文献 [36] 所提出的方法, CapsGAT 不局限于特定的模式, 更专注于构建较短的路线.

本文的贡献总结如下:

• 本文提出了一种新的胶囊聚合注意力机制, 它通过矩阵胶囊来存储更多信息, 然后利用动态路由进行信息聚合. 这是首个将胶囊网络在动态路由框架下从向量扩展到矩阵的研究.

• 为区分冗余胶囊, 本文设计了一个软门控胶囊选择器. 为反映系统状态变化, 本文改进了上下文节点向量的表示.

• 本文构建了一种新的用于求解 VRP 的学习类模型—强化学习框架下的胶囊聚合图注意力网络 CapsGAT. 据我们所知, 该工作是首次尝试专门设计一种新的注意力机制用于求解 VRP.

• 本文进行了大量的数值实验, 包括与最先进求解器进行比较、泛化能力验证、消融实验及节点选择过程的可视化.

本文的其余部分结构如下. 第 2 节简要概述注意力机制和胶囊网络. 第 3 节介绍 CapsGAT 模型. 其中第 3.1 小节详细说明编码器的结构, 包括胶囊聚合注意力子层、软门控胶囊选择器和全连接前馈子层; 第 3.2 小节描述解码器中上下文向量的构造及解码过程; 第 3.3 小节给出基于策略梯度法的强化学习训练过程. 第 4 节展示数值实验结果和对应的实验分析. 最后, 第 5 节给出全文的总结和未来可能的研究方向.

2 相关工作

2.1 注意力机制

在针对机器翻译任务的研究 [4] 中, 注意力机制被首次提出. 本文所提出的动机是在机器翻译任务中, 对于输入序列中的单词, 只有一部分与后续预测相关, 而如何更好地利用这类信息对于提升模型的表现非常重要. 因此, 注意力机制将相关性的概念通过注意力权重融入建模过程中, 使模型能够动态地关注输入中对于任务输出更重要的部分 [10], 筛选出与任务无关的信息, 并处理长期依赖关系. 注意力机制显著提升了传统序列模型的性能. 如今, 其已被成功应用于多种神经网络结构中 (参见文献 [45, 61, 62]), 并成为解决不同领域任务的热门方法 (参见文献 [11, 22, 27]).

对于求解 VRP 的学习类方法, 基于注意力机制的模型占主流, 接下来介绍几个具有代表性的学习类方法. 指针网络 [62] 使用注意力机制创建指向输入元素的指针求解 TSP, 从而避免输出字典大小固定的限制. Nazari 等 [49] 推广了指针网络来求解 CVRP. 在每个时刻, 静态元素和动态元素会被输入注意力, 可在下一决策点选择的可行目的地上形成概率分布. Ma 等 [46] 提出了图指针网络求解 TSP, 通过设计图嵌入编码器来捕获城市节点之间的关系以及引入基于注意力的解码器来输出指针向量. Kool 等 [36] 采用了基于多头注意力的编码器和基于自注意力的解码器架构来求解多种 VRP. Wu 等 [65] 设计了强化学习框架下基于 Transformer 架构的模型, 辅助 2-OPT 改进算子求解 TSP 和 CVRP. Ma 等 [47] 对模型 [65] 进行了改进, 在充分利用注意力的相关性和解的循环性的前提下, 设计了基于双向协同注意力和循环位置编码的模型求解 TSP 和 CVRP.

2.2 胶囊网络

随着神经科学的发展, 大量的神经解剖学研究证实了大脑皮层中存在大量的皮质小柱 [57]. 受此启发, Hinton 等 [31] 在 2011 年提出了胶囊的概念, 并由 Sabour 等 [55] 于 2017 年对胶囊网络进行了首次代码实现. 在人工神经网络中, 胶囊是一组神经元, 其输出代表实体的隐含定义. 具体地, 较低级别的胶囊首先被投影到新的胶囊空间中. 然后, 这些胶囊将获得的信息传输给更高级别的胶囊并进行预测, 后者通过基于余弦相似度的动态路由机制来激活. 该机制使每个活动胶囊能够选择一个父胶囊并在实体之间构建部分与整体的关系. 目前, 胶囊网络已经成功地应用于目标检测 [21]、自然语言处理 [69] 和医疗健康 [1] 等多个领域.

动态路由机制是胶囊网络中的一个关键组件, 近年来在信息聚合中得到了广泛的应用. 随着深度学习的成功发展, 层聚合引起了人们的广泛关注, 并被用于跨层融合信息. Gong 等 [25] 提出了一种聚合机制, 以获得具有动态路由策略的固定大小编码, 该策略动态地决定在文本分类任务中需要从每个单词向最终文本序列传递何种以及多少信息. Dou 等 [19] 介绍了神经机器翻译任务中的动态层聚合方法. 本文所提出的算法迭代地学习分配给聚合表示的各个层表示的概率, 并相应地组合各部分. Li 等 [43] 及 Gu 和 Feng [26] 在神经机器翻译任务中, 利用协议路由改进了多头注意力的信息聚合能力.

3 模型构建

本节简要介绍所提出模型的整体框架和对应的数学表达. 以 TSP 为例, 给定实例 $\mathcal{G}$, 总是可以定义在具有 n 个城市节点 $V=\{1,\ldots,n\}$ 的图上. 每个节点 $i\in V$ 都有自己的特征表达 $\boldsymbol{x}_i\in\mathbb{R}^2$, 这是每个节点的二维坐标. 解 $\{\boldsymbol{\pi}=(\pi_1,\ldots,\pi_T),\pi_t\in V\}$ 是序列长度为 n 的节点排列. 本节的 CapsGAT

是基于强化学习框架下的编码器 - 解码器架构设计的. 编码器计算所有输入节点的嵌入及图嵌入, 在整个过程中都是固定的. 解码器将编码器嵌入作为输入, 并用上下文信息扩充图信息, 从城市节点中逐节点生成解 $\boldsymbol{\pi}$. 对于每个时间步 t, 智能体决定新的行动并选择下一个访问节点 π_t. 状态是之前的时间步中选择的行动 $\boldsymbol{\pi}_{1:t-1}$ 的部分解. 回报设置为行程长度的负数 $-L(\boldsymbol{\pi} \mid \mathcal{G})$. 目标是学习一个能够获得最大回报 $-L(\boldsymbol{\pi} \mid \mathcal{G})$ 的自构建策略. 该策略可以通过 $\boldsymbol{\theta}$ 参数化分解表示为

$$p_{\boldsymbol{\theta}}(\boldsymbol{\pi} \mid \mathcal{G}) = \prod_{t=1}^{T} p_{\boldsymbol{\theta}}(\pi_t \mid \mathcal{G}, \boldsymbol{\pi}_{1:t-1}). \tag{3.1}$$

3.1 编码器

首先, 所有节点的特征 $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n]$ 通过一个可学习的线性变换被映射为初始节点嵌入 $\boldsymbol{H}^{(0)} = [\boldsymbol{h}_1^{(0)}, \ldots, \boldsymbol{h}_n^{(0)}]$ [1]. 然后, 初始节点嵌入被送入到编码器, 并在 N 层神经网络中进行更新. 每一层都由一个带有软门控胶囊选择器的胶囊聚合注意力子层和一个前馈子层组成. 编码器最终输出节点嵌入 $\boldsymbol{H}^{(N)} = [\boldsymbol{h}_1^{(N)}, \ldots, \boldsymbol{h}_n^{(N)}]$ 和图嵌入

$$\bar{\boldsymbol{h}}^{(N)} = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{h}_i^{(N)}. \tag{3.2}$$

3.1.1 胶囊聚合注意力子层

多头注意力模型先将节点嵌入序列转换为 M 个不同的表示子空间:

$$(\boldsymbol{Q}_m^{(\ell)}, \boldsymbol{K}_m^{(\ell)}, \boldsymbol{V}_m^{(\ell)}) = (\boldsymbol{W}_{mQ}^{(\ell)}, \boldsymbol{W}_{mK}^{(\ell)}, \boldsymbol{W}_{mV}^{(\ell)}) \boldsymbol{H}^{(\ell-1)}, \tag{3.3}$$

其中 $\boldsymbol{W}_{mQ}^{(\ell)} \in \mathbb{R}^{d_{\boldsymbol{k}} \times d_{\boldsymbol{h}}}$, $\boldsymbol{W}_{mK}^{(\ell)} \in \mathbb{R}^{d_{\boldsymbol{k}} \times d_{\boldsymbol{h}}}$, $\boldsymbol{W}_{mV}^{(\ell)} \in \mathbb{R}^{d_{\boldsymbol{v}} \times d_{\boldsymbol{h}}}$ 为可学习参数. 这里令 $d_{\boldsymbol{k}} = d_{\boldsymbol{v}} = d_{\boldsymbol{h}}/M$. 然后, M 个自注意力函数并行执行来得到输出状态 $\{\boldsymbol{H'}_1^{(\ell)}, \ldots, \boldsymbol{H'}_M^{(\ell)}\}$:

$$\boldsymbol{H'}_m^{(\ell)} = \mathrm{ATT}(\boldsymbol{Q}_m^{(\ell)}, \boldsymbol{K}_m^{(\ell)}, \boldsymbol{V}_m^{(\ell)}), \tag{3.4}$$

其中 $\mathrm{ATT}(\cdot)$ 为自注意力函数, 可以表示为

$$\mathrm{ATT}(\boldsymbol{Q}_m^{(\ell)}, \boldsymbol{K}_m^{(\ell)}, \boldsymbol{V}_m^{(\ell)}) = \boldsymbol{V}_m^{(\ell)} \operatorname{softmax_c}\left(\frac{\boldsymbol{Q}_m^{(\ell)\top} \boldsymbol{K}_m^{(\ell)}}{\sqrt{d_k}}\right). \tag{3.5}$$

之后, M 个输出状态串联为

$$\text{串联操作: } \hat{\boldsymbol{H}}^{(\ell)} = \Big\Vert_{m=1}^{M} \boldsymbol{H'}_m^{(\ell)}, \tag{3.6}$$

$\Big\Vert_{m=1}^{M}$ 为纵向顺序串联操作. 最后, 对 $\hat{\boldsymbol{H}}^{(\ell)}$ 进行线性变换以产生最终状态:

$$\text{线性变换操作: } \boldsymbol{H}^{(\ell)} = \boldsymbol{W}_H^{(\ell)} \hat{\boldsymbol{H}}^{(\ell)}, \tag{3.7}$$

其中 $\boldsymbol{W}_H^{(\ell)} \in \mathbb{R}^{d_{\boldsymbol{h}} \times d_{\boldsymbol{h}}}$ 为可训练参数.

1) 所有向量都以列表示, 即 $d \times 1$.

多头注意力机制在不同的子空间中独立地生成多个表示 [51]. 它可以通过只投影到一个唯一的子空间来避免模型性能降低的风险. 然而, 多头注意力的聚合 - 求和范式不足以充分捕捉丰富的信息 [43,44], 这将直接阻碍使用注意力机制生成 VRP 解的质量的提高. 因此, 设计一种更高级的聚合策略是一个非常值得研究的问题. 胶囊网络 [55] 提供了一种通过动态路由机制对输入胶囊的信息进行聚类的有效方法, 并将每个簇的代表性信息封装在输出胶囊中. 受此启发, 本文将原来的多头注意力子层修改为胶囊聚合注意力子层, 从而可以对来自所有头部的信息进行分类和重新整合.

针对数学形式而言, 输入胶囊是由多头注意力的输出状态 $\hat{\boldsymbol{H}}^{(\ell)}$ 进行非线性变换构建的, 即

$$\boldsymbol{O}_m^{(\ell)} = \mathrm{LeakyReLU}(\boldsymbol{W}_{mO}^{(\ell)}\hat{\boldsymbol{H}}^{(\ell)}), \tag{3.8}$$

其中, $\mathrm{LeakyReLU}(\cdot)$ 为非线性激活函数, $\{\boldsymbol{O}_1^{(\ell)},\ldots,\boldsymbol{O}_M^{(\ell)}\}$ 为输入胶囊. 需要注意的一点是, 在文献 [55] 中, 胶囊表示为向量. 然而, 本文的输入胶囊是基于多头注意力的输出构建的, 并表示为矩阵. 为了保持与注意力头部的属性对齐, 本文将输入胶囊的第一个维数设置为 $d_{\boldsymbol{v}}$, 因此 $\boldsymbol{W}_{mO}^{(\ell)} \in \mathbb{R}^{d_{\boldsymbol{v}}\times d_{\boldsymbol{h}}}$, 且输入胶囊 $\boldsymbol{O}_m^{(\ell)}$ 的形状为 $d_{\boldsymbol{v}} \times d_n$. 然后, 给定 Z 个输出胶囊, 每个输入胶囊 $\boldsymbol{O}_m^{(\ell)}$ 与这 Z 个输出胶囊进行交流, 并使用可学习的变换矩阵产生投票 $\boldsymbol{W}_{z\,|\,m}^{(\ell)} \in \mathbb{R}^{d_{\boldsymbol{v}}\times d_{\boldsymbol{v}}}$:

$$\boldsymbol{U}_{z\,|\,m}^{(\ell)} = \boldsymbol{W}_{z\,|\,m}^{(\ell)}\boldsymbol{O}_m^{(\ell)}. \tag{3.9}$$

根据耦合系数计算, 每个输出胶囊 $\boldsymbol{\Omega}_z^{(\ell)}$ 为所有投票的加权和:

$$\boldsymbol{\Omega}_z^{(\ell)} = \mathrm{Squash}\Big(\sum_{m=1}^{M} a_{z\,|\,m}^{(\ell)}\boldsymbol{U}_{z\,|\,m}^{(\ell)}\Big), \tag{3.10}$$

并且利用非线性 “挤压” 函数来表示输出胶囊被激活的概率为

$$\mathrm{Squash}(\boldsymbol{X}) = \Big\|_{k=1}^{n} \frac{\|\boldsymbol{X}_{:,k}\|_2^2}{1+\|\boldsymbol{X}_{:,k}\|_2^2}\frac{\boldsymbol{X}_{:,k}}{\|\boldsymbol{X}_{:,k}\|_2}, \tag{3.11}$$

其中 $\|\cdot\|_2$ 是向量的 2- 范数. 该过程将以自适应的方式给出耦合系数将多少信息从每个输入胶囊传递到每个输出胶囊 (部分到整体的关系) .

耦合系数 $\{a_{z\,|\,m}^{(\ell)}\}$ 全部初始化为 $1/Z$, 然后按照如下方式通过计算分对数的 softmax 值进行动态更新:

$$a_{z\,|\,m}^{(\ell)} = \frac{\exp(b_{z\,|\,m}^{(\ell)})}{\sum_{z=1}^{Z}\exp(b_{z\,|\,m}^{(\ell)})}. \tag{3.12}$$

对于每个分对数 $b_{z\,|\,m}^{(\ell)}$, 它度量了第 m 个输入胶囊 $\boldsymbol{O}_m^{(\ell)}$ 应该耦合到第 z 个输出胶囊 $\boldsymbol{\Omega}_z^{(\ell)}$ 的程度, 并通过当前输出胶囊 $\boldsymbol{\Omega}_z^{(\ell)}$ 与投票 $\boldsymbol{U}_{z\,|\,m}^{(\ell)}$ 之间的余弦相似度细化如下:

$$b_{z\,|\,m}^{(\ell)} \leftarrow b_{z\,|\,m}^{(\ell)} + \langle \mathrm{flatten}(\boldsymbol{U}_{z\,|\,m}^{(\ell)}), \mathrm{flatten}(\boldsymbol{\Omega}_z^{(\ell)})\rangle, \tag{3.13}$$

其中, $\mathrm{flatten}(\cdot)$ 表示折叠为一维的矩阵副本, $\langle\cdot,\cdot\rangle$ 表示两个向量的内积. 整体的动态路由算法总结在算法 1 中. 按顺序经过 (3.12)、(3.10) 和 (3.13) τ 次的更新, 所有的 Z 个输出胶囊串联在一起产生胶囊注意力子层的最终输出:

$$\hat{\boldsymbol{H}}^{(\ell)} = [\hat{\boldsymbol{h}}_1^{(\ell)},\ldots,\hat{\boldsymbol{h}}_n^{(\ell)}] = \Big\|_{z=1}^{Z}\boldsymbol{\Omega}_z^{(\ell)}. \tag{3.14}$$

胶囊注意力子层的整体框架如图 1 所示.

算法 1 动态路由机制

Require: 层数 ℓ, 迭代次数 τ, 输入胶囊个数 M, 输出胶囊个数 Z

procedure ROUTING($\boldsymbol{U}_{z|m}^{(\ell)}, \tau$)
  $\forall m \in \{1,\dots,M\}, z \in \{1,\dots,Z\}$: $b_{z|m}^{(\ell)} \leftarrow 0$
  for 第 τ 次迭代 **do**:
    $\forall m \in \{1,\dots,M\}, z \in \{1,\dots,Z\}$: 公式 (3.12)
    $\forall z \in \{1,\dots,Z\}$: 公式 (3.10)
    $\forall m \in \{1,\dots,M\}, z \in \{1,\dots,Z\}$: 公式 (3.13)
  end for
return $\boldsymbol{\Omega}_z^{(\ell)}$
end procedure

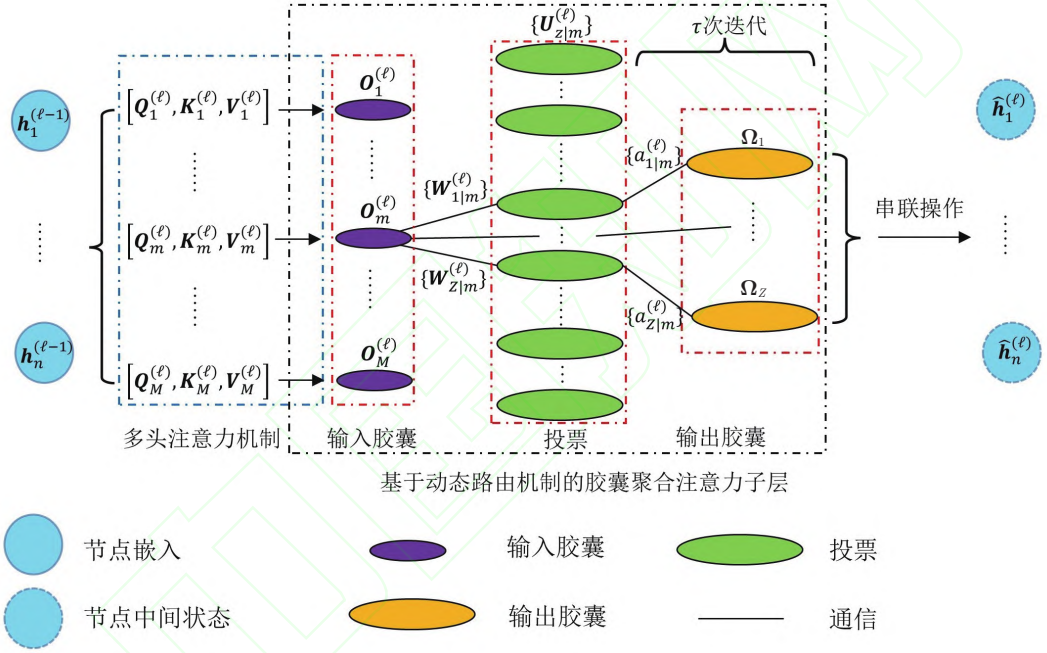

图 1 胶囊聚合注意力子层的架构

3.1.2 软门控胶囊选择器

通过引入具有动态路由机制的胶囊聚合注意力, 多头注意力中的信息聚合过程得到了增强. 然而, 当最终聚合多个输出胶囊时, 所有这些胶囊仍然只是简单地连接在一起. 注意到并非所有这些胶囊都同样有助于生成给定城市节点的嵌入, 有些胶囊甚至可能是冗余的 [57]. 不加区别地处理每个胶囊将导致捕获无用的表达, 并且有可能会误导最终预测, 失去从某些胶囊中获益的机会 (参见文献 [68]). 因此, 用软门控胶囊选择器修改胶囊聚合注意力子层, 通过计算每个输出胶囊的软门控值, 对其重要性加以区分, 门控值的范围在 0 到 1 之间. 软门控胶囊选择器更新的城市节点可以表示为

$$\hat{\boldsymbol{h}}_i^{(\ell)} = \Big\Vert_{z=1}^{Z} g_{iz}^{(\ell)} \boldsymbol{\Omega}_{z_{(:,i)}}^{(\ell)}, \tag{3.15}$$

其中 $g_{iz}^{(\ell)}$ 为门控值, 表示胶囊 z 对节点 i 的重要性. 门控值向量 $\boldsymbol{g}_i^{(\ell)} = [g_{i1}^{(\ell)}, \dots, g_{iZ}^{(\ell)}]$ 可以被计算为

$$\boldsymbol{g}_i^{(\ell)} = \mathrm{FC}_{\boldsymbol{\theta}_g}^{\sigma}(\boldsymbol{h}_i^{(\ell-1)}), \tag{3.16}$$

其中 $\mathrm{FC}^{\sigma}_{\boldsymbol{\theta}_g}$ 使用 sigmoid 激活函数将特征映射到 Z 个门. 与文献 [68] 不同, 这里不考虑邻居节点的最大池化和平均池化. 这是因为 VRP 的图结构是全连通的, 并且每个节点与其邻居具有相同的关系. 因此, 只关注给定的节点本身, 并构建一个简单的全连接子网络. 软门控选择器的结构如图 2 所示.

3.1.3 全连接前馈子层

在该子层中, 遵循文献 [36] 中的图注意力网络架构, 将软门控向量发送到逐节点的全连接前馈网络. 此外, 将跳跃连接 [29] 和批规范化 [33] 添加到胶囊聚合子层, 具体公式如下:

$$\tilde{\boldsymbol{h}}_i^{(\ell)} = \mathrm{BatchNorm}(\boldsymbol{h}_i^{(\ell-1)} + \hat{\boldsymbol{h}}_i^{(\ell)}), \tag{3.17}$$

$$\mathrm{FF}(\tilde{\boldsymbol{h}}_i^{(\ell)}) = \mathrm{FC}_{\boldsymbol{\theta}_{f_2}}(\mathrm{FC}^{\mathrm{ReLU}}_{\boldsymbol{\theta}_{f_1}}(\tilde{\boldsymbol{h}}_i^{(\ell)})), \tag{3.18}$$

其中 $\mathrm{FF}(\cdot)$ 表示前馈网络, 该前馈网络包含一个具有 $4 \cdot d_{\boldsymbol{h}}$ 维数和 ReLu 激活的隐藏子层. 然后再次使用跳跃连接和批规范化来生成最终的节点嵌入. 对于节点 i, 由层 ℓ 计算的最终节点嵌入可以公式化为

$$\boldsymbol{h}_i^{(\ell)} = \mathrm{BatchNorm}(\tilde{\boldsymbol{h}}_i^{(\ell)} + \mathrm{FF}^{(\ell)}(\tilde{\boldsymbol{h}}_i^{(\ell)})). \tag{3.19}$$

综上, 编码器的整体架构如图 3 所示.

3.2 解码器

通过编码过程获得节点嵌入和图嵌入后, 我们开始逐步构建输出解. 在解码过程中, 需要记录状态随时间的变化, 因此使用上下文节点向量来增强图信息, 以表示解码的上下文信息. 本文考虑二维 Euclid 距离下两个经典的 VRP, 即 TSP 和 CVRP. TSP 的目标是所有城市节点均被访问且只被访问一次, 并以最短的行程返回出发城市, 因此序列长度为 $T = n$. CVRP 是 TSP 的一个推广, 其目的也是寻找最短路线. 一系列客户节点有自己的需求, 并构建多条以仓库为起点和终点的子路线. 约束条件必须满足: (1) 路线中的每个客户需要且仅被访问一次, 因此 $T > n + 1$; (2) 每条路线上所有客户的需求都不会超过车辆的容量. 对于 CVRP, 原始的 n 个节点代表客户, 每个节点的需求为 $\delta_i \in (0, D]$, 其

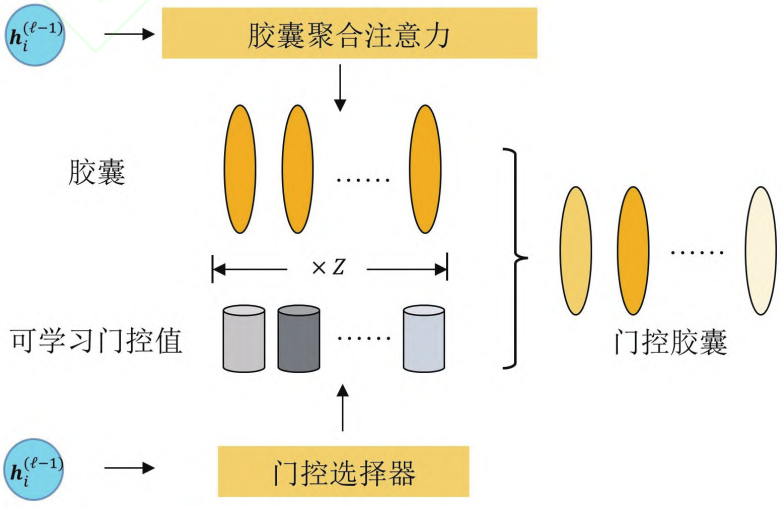

图 2 软门控胶囊选择器的整体结构. 具有不同灰度的柱体表示不同的门控值

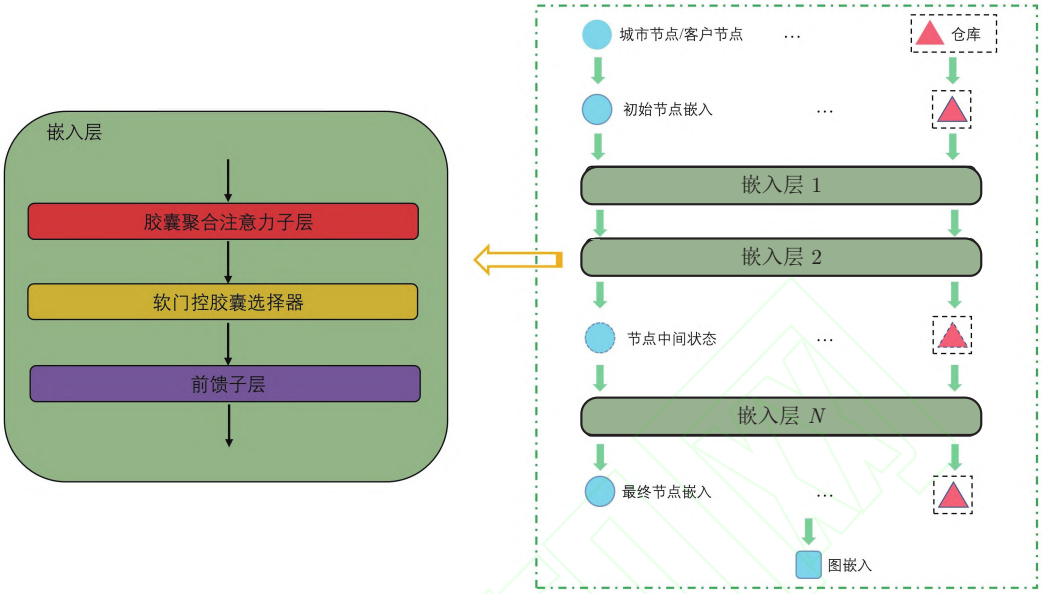

图 3 编码器的整体架构. 首先对输入节点进行投影以获得初始嵌入. 然后, 将所有节点的初始嵌入发送到嵌入层. 它由带有软门控胶囊选择器的胶囊聚合注意力子层和顺序连接的全连接前馈子层组成. 在经过 N 次处理后, 通过所有节点的嵌入的平均值来获得图嵌入

中 $D>0$ 表示车辆容量. 每个客户节点的特征在二维坐标的基础上还增加了相应的需求. 此外, 节点 $i=0$ 被添加到 V 中代表仓库, $\delta_0=0$.

对于解码过程中的每一步 $t\in\{1,\ldots,T\}$, 当前步骤中的节点根据节点嵌入和已生成的部分路径 $\pi_{t'<t}$ 选择下一个访问节点. 此外, 还需要上下文节点向量 $\boldsymbol{c}$ 来表示解码的上下文信息. 在文献 [36] 中, 上下文节点向量构造如下:

$$\boldsymbol{c}_{(\mathrm{TSP})}=\begin{cases}[\bar{\boldsymbol{h}}^{(N)^\top},\boldsymbol{h}_{\pi_{t-1}}^{(N)^\top},\boldsymbol{h}_{\pi_1}^{(N)^\top}]^\top, & t>1,\\ [\bar{\boldsymbol{h}}^{(N)^\top},\boldsymbol{v}_1^\top,\boldsymbol{v}_f^\top]^\top, & t=1,\end{cases}\tag{3.20}$$

$$\boldsymbol{c}_{(\mathrm{CVRP})}=\begin{cases}[\bar{\boldsymbol{h}}^{(N)^\top},\boldsymbol{h}_{\pi_{t-1}}^{(N)^\top},D_t]^\top, & t>1,\\ [\bar{\boldsymbol{h}}^{(N)^\top},\boldsymbol{h}_0^{(N)^\top},D_t]^\top, & t=1,\end{cases}\tag{3.21}$$

其中 $\boldsymbol{v}_1$ 和 $\boldsymbol{v}_f$ 是 $d_{\boldsymbol{h}}$ 维的输入占位符. 对于 CVRP, 当 $t=1$ 时不需要占位符, 因为每条路线的起点和终点都是仓库.

但是, 在 $\boldsymbol{c}$ 中能反映系统状态变化的元素仅为上一步访问的节点, 即 $\boldsymbol{h}_{\pi_{t-1}}^{(N)}$, 这不足以反映出整个系统的状态随时间的变化. 对于 VRP, 由于我们关心那些尚未被访问的城市节点, 因此受文献 [66] 的启发, 本文将未访问的节点信息添加到上下文节点向量中, 并设计了一个新版本的上下文节点向量 $\boldsymbol{c}$ 如下:

$$\boldsymbol{c}_{(\mathrm{TSP})}=[\mathbf{unv}_t^\top,\bar{\boldsymbol{h}}^{(N)^\top},\boldsymbol{h}_{\pi_{t-1}}^{(N)^\top},\boldsymbol{h}_{\pi_1}^{(N)^\top}]^\top,\tag{3.22}$$

$$\boldsymbol{c}_{(\mathrm{CVRP})}=[\mathbf{unv}_t^\top,\bar{\boldsymbol{h}}^{(N)^\top},\boldsymbol{h}_{\pi_{t-1}}^{(N)^\top},D_t]^\top,\tag{3.23}$$

其中 $\mathbf{unv}_t$ 表示未访问的子图嵌入, 计算公式如下:

$$\mathbf{MASK}_t=\mathrm{Repeat}_{d_{\boldsymbol{h}}}(\mathrm{FC}_{\boldsymbol{\theta}_{unv}}^{\mathrm{Softmax}}(\boldsymbol{H}^{(N)})^\top\odot\mathbf{mask}_t^\top),\tag{3.24}$$

$$\mathbf{unv}_t = \mathrm{Ave}(\mathbf{MASK}_t \odot \boldsymbol{H}^{(N)}), \tag{3.25}$$

这里, $\mathrm{FC}_{\boldsymbol{\theta}_{unv}}^{\mathrm{Softmax}}$ 将 $\boldsymbol{H}^{(N)}$ 映射为 n 个权重, 掩码 $\mathbf{mask}_t$ 是一个二值化的向量, 其中未访问的节点设置为 1, 否则为 0, $\odot$ 代表 Hadamard 积, $\mathrm{Repeat}_{d_{\boldsymbol{h}}}$ 表示为了生成矩阵而在列的维数上重复 $d_{\boldsymbol{h}}$ 次, $\mathrm{Ave}(\cdot)$ 表示计算节点数量维数的平均值. 值得注意的是, 对于 CVRP, 由于仓库被反复访问, 因此仓库一直设置为未访问状态. 对于节点 $i \in \{1, \ldots, n\}$, 掩码中的相应元素被更新为

$$\mathbf{mask}_{i,t+1} = \begin{cases} 1, & \text{若节点 } i \text{ 在 } t \text{ 时刻没有被访问}, \\ 0, & \text{其他}. \end{cases} \tag{3.26}$$

然后使用一个含有 M 个头的多头注意力子层计算上下文节点嵌入 $\boldsymbol{h}_{\boldsymbol{c}}$. 最终用于输出层的查询 $\boldsymbol{q}_{\boldsymbol{c}}$ 和键值 $\boldsymbol{k}$ 分别由上下文节点嵌入 $\boldsymbol{h}_{\boldsymbol{c}}$ 和城市 (客户) 节点嵌入进行线性变换, 得到

$$\boldsymbol{q}_{\boldsymbol{c}} = \boldsymbol{W}_q \boldsymbol{h}_{\boldsymbol{c}}, \quad \boldsymbol{k}_j = \boldsymbol{W}_k \boldsymbol{h}_j^{(N)}, \tag{3.27}$$

其中 $\boldsymbol{W}_q \in \mathbb{R}^{d_{\boldsymbol{h}} \times d_{\boldsymbol{c}}}$ 和 $\boldsymbol{W}_k \in \mathbb{R}^{d_{\boldsymbol{h}} \times d_{\boldsymbol{h}}}$ 是可训练的.

最后, 使用一个单头注意力计算输出概率 $p_\theta(\pi_t \mid \mathcal{G}, \pi_{1:t-1})$, 即

$$p_j = p_\theta(\pi_t = j \mid \mathcal{G}, \pi_{1:t-1}) = \frac{\mathrm{e}^{u_j}}{\sum_{j'} \mathrm{e}^{u_j}}, \tag{3.28}$$

$$u_j = \begin{cases} C \cdot \tanh\left(\dfrac{\boldsymbol{q}_{\boldsymbol{c}}^{\mathrm{T}} \boldsymbol{k}_j}{\sqrt{d_k}}\right), & \text{若 } \mathbf{mask}_{j,t} = 1, \\ -\infty, & \text{其他}, \end{cases} \tag{3.29}$$

其中 C 用于裁剪值使其介于 $[-C, C]$. 图 4 展示了解码过程.

3.3 模型训练

在解码过程的最后, 求得的解 $\boldsymbol{\pi}$ 由每个时间步选择的行动而生成 $(\pi_1, \ldots, \pi_T)$. 我们的目标是优化行动策略, 以获得最优的路径长度. 与基于值的方法旨在找到最佳状态 - 动作值函数并对其贪婪地采取行动以获得最佳策略不同, 基于策略的方法旨在直接对策略进行建模和优化. 此外, 对随机策略进行建模是很自然的事情, 这在求解 VRP 时更为合理. 因此, 本文根据策略梯度方法对模型进行优化.

给定一个实例 $\mathcal{G}$, 解码器对轨迹 $\boldsymbol{\pi}$ 进行采样而得到训练损失, 其定义为对路径长度 $L(\boldsymbol{\pi} \mid S)$ 的期望:

$$\mathcal{L}(\boldsymbol{\theta} \mid \mathcal{G}) = \mathrm{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{\pi} \mid \mathcal{G})} L(\boldsymbol{\pi} \mid \mathcal{G}). \tag{3.30}$$

根据策略梯度定理 [58], 策略梯度公式为

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta} \mid \mathcal{G}) = \mathrm{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{\pi} \mid \mathcal{G})}[(L(\boldsymbol{\pi} \mid \mathcal{G}) - b(\mathcal{G})) \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{\pi} \mid \mathcal{G})], \tag{3.31}$$

其中 $b(\mathcal{G})$ 是用于减少梯度方差和加速收敛的轨迹基线估计器. 由于策略 (模型) 随时间变化, 所以将轨迹基线策略更新为每个时期的最新策略. 在训练过程中, 每个实例 $\mathcal{G}_i$ 都来自同一分布, 因此通过 Monte Carlo 采样, 策略梯度可以近似表示为

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta} \mid \mathcal{G}) = \frac{1}{B} \sum_{i=1}^{B} [(L(\boldsymbol{\pi}_i^s \mid \mathcal{G}_i) - L(\boldsymbol{\pi}_i^g \mid \mathcal{G}_i)) \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{\pi}_i^s \mid \mathcal{G}_i)], \tag{3.32}$$

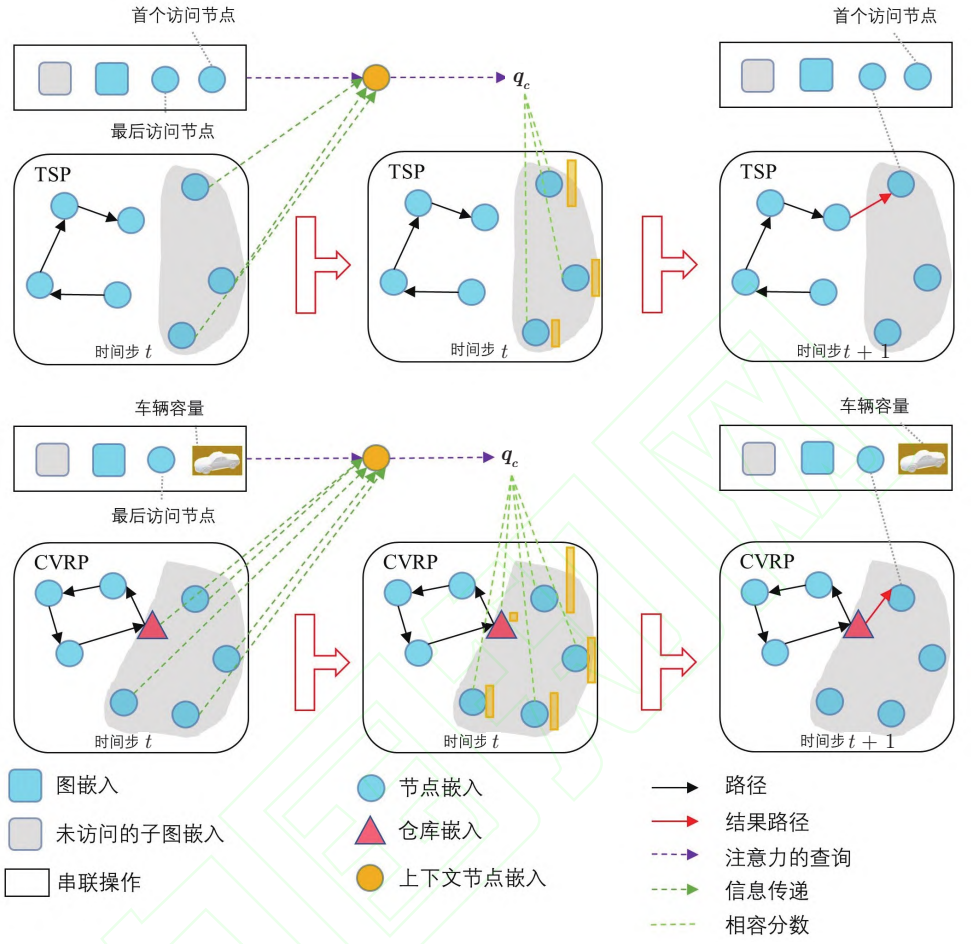

图 4 **TSP 和 CVRP 在第 t 步的解码过程**

其中, B 是批的大小, $\boldsymbol{\pi}_i^s$ 和 $\boldsymbol{\pi}_i^g$ 表示采样策略和贪婪策略推导出的轨迹. Adam 优化器 [35] 被用来更新 $\boldsymbol{\theta}$. 模型训练过程展示在算法 2 中.

算法 2 基于 REINFORCE 的策略优化算法

Require: 训练时期的数量 E, 每个时期的步数 F, 批的大小 B
Ensure: 最终策略 $\boldsymbol{\theta}$

初始化 $\boldsymbol{\theta}$
$\boldsymbol{\theta}^{\mathrm{BL}} = \boldsymbol{\theta}$
for epoch $= 1, \ldots, E$ **do**:
  for step $= 1, \ldots, F$ **do**:
    $S_i \leftarrow \mathrm{RandomInstance}(), \forall\, i \in \{1, \ldots, B\}$
    $\boldsymbol{\pi}_i^s \leftarrow \mathrm{SampleRollout}(\mathcal{G}_i, p_{\boldsymbol{\theta}}), \forall\, i \in \{1, \ldots, B\}$
    $\boldsymbol{\pi}_i^g \leftarrow \mathrm{GreedyRollout}(\mathcal{G}_i, p_{\boldsymbol{\theta}^{\mathrm{BL}}}), \forall\, i \in \{1, \ldots, B\}$
    $\nabla_{\boldsymbol{\theta}} \mathcal{L} \leftarrow \frac{1}{B} \sum_{i=1}^{B} [(L(\boldsymbol{\pi}_i^s|\mathcal{G}_i) - L(\boldsymbol{\pi}_i^g|\mathcal{G}_i)) \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{\pi}_i^s|\mathcal{G}_i)]$
    $\boldsymbol{\theta} \leftarrow \mathrm{Adam}(\boldsymbol{\theta}, \nabla_{\boldsymbol{\theta}} J)$
  end for
  $\boldsymbol{\theta}^{\mathrm{BL}} = \boldsymbol{\theta}$
end for
return $\boldsymbol{\theta}$

4 实验设计及分析

选择在两类经典的 VRP—二维 Euclid 距离下的 TSP 和 CVRP 上进行数值实验. 所有实验均在单个 GPU (graphics processing unit) (Nvidia 3090) 上进行.

4.1 实验设置

在本文的小规模 TSP 实验和 CVRP 实验中使用的实例包含 20 个、50 个和 100 个节点. 为了方便起见, 它们分别表示为 TSP20、TSP50、TSP100、CVRP20、CVRP50 和 CVRP100. 每个城市 (或仓库、客户) 的坐标是随机采样的, 在单位正方形区域 $[0,1]\times[0,1]$ 上均匀分布. 对于 CVRP, 车辆容量 $D_{(20)}=30$, $D_{(50)}=40$, $D_{(100)}=50$. 每个客户的需求在 $\{1,\ldots,9\}$ 中均匀随机选取, 仓库的需求被设置为 0.

对于 CapsGAT 的超参数, 为权衡性能和计算复杂度, 设置编码器的层数为 $N=3$, 维数 $d_{\boldsymbol{h}}$、$d_{\boldsymbol{k}}$ 和 $d_{\boldsymbol{v}}$ 分别为 128、16 和 16, 注意力头和胶囊的数量都为 8. 此外, 动态路由算法的迭代次数 $\tau=2$, 剪切参数 $C=10$. 在训练过程中, 将训练时期的数量设置为 100, 学习率设置为 10^{-4}. 对于每个时期, 为适应 GPU 的显存, 在 2,500 个批次, 每个批次含有 512 个实例上实现训练过程; 而对于 CVRP100, 在 2,500 个批次, 每个批次含有 256 个实例上实现训练过程. 在测试过程中, 本文报告了模型在 10,000 个测试实例上的性能.

4.2 实验基线

将 CapsGAT 与一些传统的求解器、启发式基线和学习类基线进行比较. 各基线具体介绍如下:

• Concorde [2]: 一种专门用于求解 TSP 的精确求解器, 它使用割平面法迭代求解 TSP 整数规划模型的线性松弛;

• LKH3 [30]: 一种启发式求解器, 使用 3-OPT 找到最优解, 在许多 VRP 上实现了最先进的结果;

• OR-Tools [3]: 一种广泛使用的基于元启发式的 VRP 求解器, 由谷歌生产, 具有优异的性能;

• Nearest neighbor [54]: 一种通过选择最近访问节点的最近可行邻域的启发式方法;

• Random/Nearest Insertion [54]: 一种通过选择插入成本最小的节点的启发式方法;

• Clarke-wright saving [14]: 一种根据可产生的最大节约成本合并路径的启发式方法;

• Sweep saving [24]: 一种在极坐标系下对顶点进行排序和交换以节省成本的启发式方法;

• PtrNet-RL [7]: 强化学习框架下的指针网络;

• Nazari-RL [49]: PtrNet-RL 的一个广义版本;

• EAN [18]: 强化学习框架下的图注意力网络模型, 具有多头注意力编码过程和指针机制解码过程;

• AM [36]: 强化学习框架下的图注意力网络模型, 具有多头注意力编码过程和自注意力解码过程;

• GPN [46]: 强化学习框架下的图指针网络模型;

• MRAM [66]: 强化学习框架下的多关系 AM 模型;

• POMO [37]: 最先进的基于注意力的自构建模型;

2) Http://www.math.uwaterloo.ca/tsp/concorde.
3) Https://developers.google.com/optimization/.

• LIH-DACT [47]: 一种最先进的基于注意力的改进启发式模型, 使用双向协同 Transformer 和循环位置编码来辅助选择 2-OPT 算子;

• AMDKD-AM [9]: 最先进的基于注意力的模型之一, 具有多分布知识蒸馏框架, 其中教师网络基于 AM;

• RL-CSL [67]: 基于强化学习和自监督对比学习的 AM 模型.

4.3 实验结果

本小节将本文的模型与各个基线进行比较. 对比结果展示在表 1 中. 对于学习类模型, 在测试时有两种常见的解码策略: 贪婪解码和采样解码. 其中贪婪解码在每一步中选择概率最高的最佳动作作为下一个目的地, 而采样解码对许多解进行采样并返回最优解. 本文分别使用这两种解码策略来评估本文的模型, 并将采样解的数量设置为 1,280. 为了强调模型的特性和解码策略, 将所有基线分组为求解器 (Solver)、启发式 (H)、贪婪解码 (G) 和采样解码 (S). 对于与本文的模型不同类别的基线, 使用单轨迹 (ST) 来表征 POMO [4], 使用时间步长限制 (T) 来表征 LIH-DACT. AMDKD-AM 和 RL-CSL 的分类方式与我们的贪婪和采样模型相同. 为全面评估模型, 本文主要关注路径长度 (Tour Len.)、最优差距 (Opt. Gap) 和测试时间 (Time). 不能得到的结果标记为 "–". 最优差距可以公式化为

$$\text{Opt. Gap.} = \frac{L - L_{\text{best}}}{L_{\text{best}}} \times 100\%. \tag{4.1}$$

本文使用精确求解器 Concorde 和 LKH3 [30] 来分别得到 TSP 和 CVRP 的最优路径长度.

从表 1 中首先观察到, 在推理时间上, 学习类模型可以比精确求解器更快地获得解. 而就解的质量而言, 学习类模型优于所有启发式基线. 其次, 通过比较贪婪解码组和采样解码组, 可以发现贪婪解码策略所需要的推理时间更短, 而采样解码可以获得更高质量的解. 在实际应用中, 当需要快速获得具有良好质量的初始解时, 贪婪解码策略是一个不错的选择. 当追求解的质量时, 特别是对于大规模问题, 采样解码更符合需求.

接下来, 对所提出的模型和其他学习类基线进行更详细的比较. 我们发现 CapsGAT 在所有针对模块修改系列的模型中获得了 Tour Len 和 Opt. Gap 最好的改进. 具体而言, 对于规模较小的实例, 如 TSP20 和 CVRP20, 尽管 AM 的 Opt. Gap 已足够小, 但 CapsGAT 仍然能够作出一些改进, 例如在 TSP20 上能够获得 0.03% 的提升, 在 CVRP20 上获得 0.76% 的提升. 随着问题规模的增加, 本文的模型获得的解的质量得到了更明显的提高. 此外, 由于编码器 (胶囊聚合注意力和软门控选择器) 和解码器 (未访问的子图嵌入) 的额外计算成本, 本文的模型的推理时间比 AM 和 MRAM 稍长. 但考虑到本文的模型在求解质量方面的优势, 我们认为稍微增加推理的时间成本是值得的. 此外, 随着问题规模的增加, 本文的模型所花费的额外时间与总推理时间的比率逐渐降低. 最后, 必须承认的是, 与最先进的基线相比, AM 模型已经逐渐被超越. 然而, 基于 AM 的专门设计的注意力模型显著提高了其性能, 不仅缩小了与这些最先进模型的差距, 而且在所有情形下都优于 RL-CSL, 甚至在一些情形下优于 POMO、LIH-DACT 和 AMDKD-AM. 这表明了本文方法的有效性.

4) 如文献 [37] 所述, 由于 POMO 从多个轨迹中选择最佳轨迹, 这使得 POMO 与 AM 系列模型相比具有不公平的优势. 因此, 在 POMO 训练的网络上以单轨迹模式进行推理.

表 1　与非学习类基线和学习类基线相比, CapsGAT 在小规模 TSP 和 CVRP 实例上的性能

方法	类型	TSP20			TSP50			TSP100		
		Tour Len.	Opt. Gap.	Time	Tour Len.	Opt. Gap.	Time	Tour Len.	Opt. Gap.	Time
Concorde	Solver	**3.83**	**0.00**%	5 min	**5.69**	**0.00**%	13 min	**7.76**	**0.00**%	1 h
LKH3	Solver	**3.83**	**0.00**%	42 s	**5.69**	**0.00**%	6 min	**7.76**	**0.00**%	25 min
OR-Tools	Solver	3.86	0.94%	1 min	5.85	2.87%	5 min	8.06	3.86%	23 min
Nearest Neighbour	H	4.50	17.49%	5 s	7.00	23.02%	5 s	9.68	24.74%	5 s
Nearest Insertion	H	4.33	12.91%	1 s	6.78	19.03%	2 s	9.46	21.82%	6 s
Random Insertion	H	**4.00**	**4.36**%	1 s	**6.13**	**7.65**%	1 s	**8.52**	**9.70**%	3 s
PtrNet-RL	G	3.89	1.42%	–	5.95	4.46%	–	8.30	6.90%	–
GPN	G	3.87	0.96%	1 s	5.96	4.56%	4 s	8.51	9.69%	10 s
EAN	G	3.86	0.66%	1 min	5.92	3.98%	3 min	8.42	8.41%	6 min
AM	G	3.85	0.34%	1 s	5.79	1.68%	2 s	8.12	4.53%	5 s
MRAM	G	3.84	0.27%	2 s	5.78	1.55%	3 s	8.08	4.13%	6 s
CapsGAT [Ours]	G	**3.84**	**0.22**%	2 s	**5.77**	**1.44**%	2 s	**8.05**	**3.72**%	7 s
EAN	S (1280)	3.84	0.11%	3 min	5.79	1.93%	7 min	8.77	13.21%	20 min
AM	S (1280)	3.84	0.08%	3 min	5.73	0.52%	10 min	7.94	2.26%	23 min
MRAM	S (1280)	3.84	0.06%	5 min	5.72	0.46%	14 min	7.92	1.93%	27 min
CapsGAT [Ours]	S (1280)	**3.83**	**0.05**%	5 min	**5.71**	**0.38**%	15 min	**7.90**	**1.83**%	29 min
POMO	ST	3.83	0.12%	1 s	5.73	0.64%	2 s	**7.84**	**1.07**%	7 s
LIH-DACT	T (1000)	**3.83**	**0.04**%	2 min	**5.70**	**0.14**%	5 min	7.89	1.62%	7 min
AMDKD-AM	G	3.86	0.83%	1 s	5.80	2.00%	3 s	8.04	3.65%	6 s
AMDKD-AM	S (1280)	3.85	0.39%	3 min	5.70	0.25%	8 min	7.86	1.21%	20 min
RL-CSL	G	3.84	0.22%	1 s	5.78	1.62%	2 s	8.10	4.41%	6 s
RL-CSL	S (1280)	3.84	0.06%	5 min	5.72	0.51%	24 min	7.94	2.35%	1 h

方法	类型	CVRP20			CVRP50			CVRP100		
		Tour Len.	Opt. Gap.	Time	Tour Len.	Opt. Gap.	Time	Tour Len.	Opt. Gap.	Time
LKH3	Solver	**6.11**	0.00%	1 h	**10.38**	**0.00**%	5 h	**15.68**	**0.00**%	9 h
OR Tools	Solver	6.46	5.68%	2 min	11.27	8.61%	13 min	17.12	9.18%	46 min
Clarke-Wright saving	H	**6.81**	**11.64**%	–	**12.25**	**18.07**%	–	**18.96**	**20.96**%	–
Sweep saving	H	7.08	16.07%	–	12.96	24.91%	–	20.33	29.70%	–
Nearest Neighbour	H	7.51	22.91%	5 s	13.01	25.34%	5 s	19.45	24.04%	5 s
Nazari-RL	G	6.59	8.03%	–	11.39	9.78%	–	17.23	9.12%	–
AM	G	6.40	4.75%	1 s	10.98	5.78%	2 s	16.76	6.89%	6 s
MRAM	G	6.39	4.77%	2 s	10.95	5.49%	3 s	16.69	6.43%	7 s
CapsGAT [Ours]	G	**6.38**	**4.66**%	2 s	**10.92**	**5.21**%	4 s	**16.62**	**5.98**%	7 s
AM	S (1280)	6.27	2.89%	5 min	10.62	2.38%	20 min	16.24	3.89%	51 min
MRAM	S (1280)	6.25	2.43%	7 min	10.61	2.34%	22 min	16.20	3.32%	53 min
CapsGAT [Ours]	S (1280)	**6.23**	**2.13**%	7 min	**10.59**	**2.02**%	25 min	**16.11**	**2.75**%	54 min
POMO	ST	6.35	3.72%	1 s	10.74	3.52%	2 s	16.15	3.01%	8 s
LIH-DACT	T (1000)	**6.15**	**0.28**%	4 min	10.61	2.13%	9 min	16.17	3.18%	15 min
AMDKD-AM	G	6.35	3.85%	1 s	10.89	4.91%	4 s	16.50	5.23%	8 s
AMDKD-AM	S (1280)	6.19	1.31%	5 min	**10.58**	**1.93**%	10 min	**16.07**	**2.47**%	25 min
RL-CSL	G	6.40	5.16%	1 s	10.96	5.64%	3 s	16.74	6.99%	8 s
RL-CSL	S (1280)	6.25	2.47%	6 min	10.61	2.18%	30 min	16.18	3.40%	2 h

注: 差距是以 LKH3 作为最优解计算的. 加粗数字代表每组中最好的结果.

为了研究所提出的新方法能否增强学习能力和提高收敛效率, 将 CapsGAT 与 AM 和 MRAM 基线进行了进一步比较, 后者具有与本文相同的建模和训练框架模式. 这 3 个模型在 TSP 和 CVRP 实例上的学习曲线如图 5 所示. 一般来说, 如果一个模型具有更强大的学习能力, 则它可以在相同的训练时期内取得比对照组更好的结果. 如果一个模型更快地收敛, 则它可以在更少的训练周期内实现与对照组相似的性能. 如图 5 所示, 一方面, CapsGAT 的曲线总是低于 AM 和 MRAM 的曲线. 这表明提出的模型确实提高了学习能力, 并进一步缩小了与最优解的差距; 另一方面, 以 TSP50 为例, CapsGAT 的第 20 个时期的路径长度与 AM 的第 60 个时期相近, 这表明本文的模型比 AM 和 MRAM 具有更快的收敛速度和更高的收敛效率.

4.4 泛化性能评价

泛化能力是衡量学习类模型优劣的一个重要方面. 因此, 本小节从不同的角度进行了一系列实验, 以测试本文的 CapsGAT 的泛化性能.

4.4.1 小规模 TSP 和 CVRP 实例上的泛化性能

首先在小规模 TSP 和 CVRP 实例上进行了实验, 并在表 2 中报告了路径长度的泛化结果. 具体而言, 本文在 20、50 和 100 个节点的 TSP 上训练的模型分别表示为 CapsGAT-TSP20、CapsGAT-TSP50 和 CapsGAT-TSP100, 在 CVRP 上训练的模型分别表示为 CapsGAT-CVRP20、CapsGAT-CVRP50 和 CapsGAT-CVRP100. 本文对更容易的 TSP 使用贪婪解码策略, 对更难的 CVRP 使用采样解码策略. 在测试过程中, 使用不同节点大小的数据集对模型进行测试, 即 CapsGAT-TSP20 在具有 50 和 100 个节点的实例上进行测试, CapsGAT-CVRP50 在具有 20 和 100 个节点的实例上测试. 结果表明, 本文的

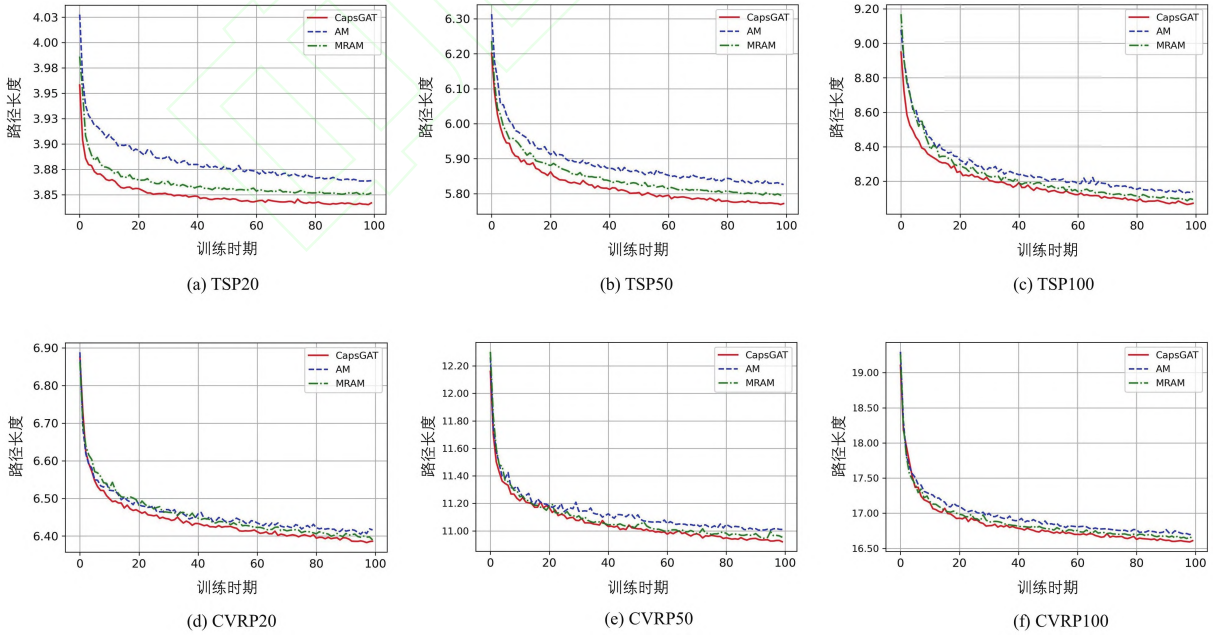

图 5 **AM、MRAM 和 CapsGAT 的测试行程长度与 (a) TSP20, (b) TSP50, (c) TSP100, (d) CVRP20, (e) CVRP50, (f) CVRP100 实例上的训练时期的关系图. 蓝色虚线表示 AM 模型, 绿色点线表示 MRAM 模型, 红色实线表示本文的 CapsGAT 模型. 随着问题规模的增加, 本文的模型具有更明显的优势**

表 2　小规模 TSP 和 CVRP 实例的泛化结果

方法	类型	TSP20	TSP50	TSP100	方法	类型	CVRP20	CVRP50	CVRP100
Concorde	Solver	3.83	5.69	7.76	LKH3	Solver	6.11	10.38	15.68
AM-TSP20	G	–	5.95	9.02	AM-CVRP20	S	–	10.97	17.98
MRAM-TSP20	G	–	5.93	8.87	MRAM-CVRP20	S	–	10.94	17.97
POMO-TSP20	ST	–	5.99	9.81	POMO-CVRP20	ST	–	11.04	18.11
LIH-DACT-TSP20	T	–	6.27	9.66	LIH-DACT-CVRP20	T	–	11.69	19.80
AMDKD-AM-TSP20	G	–	5.99	9.02	AMDKD-AM-CVRP20	S	–	11.06	22.25
CapsGAT-TSP20	G	–	**5.90**	**8.84**	CapsGAT-CVRP20	S	–	**10.92**	**17.95**
AM-TSP50	G	3.89	–	8.18	AM-CVRP50	S	6.39	–	16.39
MRAM-TSP50	G	3.89	–	8.14	MRAM-CVRP50	S	6.37	–	16.38
POMO-TSP50	ST	3.88	–	8.15	POMO-CVRP50	ST	8.02	–	16.49
LIH-DACT-TSP50	T	4.13	–	8.30	LIH-DACT-CVRP50	T	6.46	–	17.31
AMDKD-AM-TSP50	G	3.89	–	8.20	AMDKD-AM-CVRP50	S	**6.27**	–	**16.28**
CapsGAT-TSP50	G	**3.87**	–	**8.12**	CapsGAT-CVRP50	S	6.35	–	16.35
AM-TSP100	G	4.22	5.99	–	AM-CVRP100	S	6.49	10.82	–
MRAM-TSP100	G	4.18	5.96	–	MRAM-CVRP100	S	6.48	10.79	–
POMO-TSP100	ST	4.14	5.92	–	POMO-CVRP100	ST	10.43	11.60	–
LIH-DACT-TSP100	T	4.27	6.30	–	LIH-DACT-CVRP100	T	7.16	10.78	–
AMDKD-AM-TSP100	G	4.28	5.88	–	AMDKD-AM-CVRP100	S	**6.39**	**10.72**	–
CapsGAT-TSP100	G	**4.13**	**5.91**	–	CapsGAT-CVRP100	S	6.46	10.77	–

注: 加粗数字代表每组中最好的结果.

CapsGAT 在大多数情形下对 TSP 和 CVRP 具有更好的泛化能力, 例如在所有情形下都超过了 POMO 和 LIH-DACT 这两个具有代表性的最先进的基线. 在 CVRP50 和 CVRP100 上训练的 AMDKD-AM 模型在小规模实例上可以获得更好的泛化结果, 但在评估其他实例时, 本文的模型与 AMDKD-AM 之间的差距并不大.

4.4.2　大规模 TSP 和 CVRP 实例上的泛化性能

本文比较了 CapsGAT 与 AM、MRAM、POMO、LIH-DACT 和 AMDKD-AM 在最多含有 1,000 个节点的大规模 TSP 和 CVRP 实例上的泛化性能. 对于 CVRP, 将所有规模的大型 CVRP 实例的车辆容量设置为 70. 本文使用在 TSP100 和 CVRP100 上预训练好的模型, 并分别在 TSP200、TSP500、TSP1000 和 CVRP200、CVRP500、CVRP1000 上评估泛化能力, 每个测试集都有 10,000 个实例. 为了节省推理时间, 在测试过程中使用贪婪解码策略. 泛化结果见表 3. 这里还比较了 Opt. Gap 和推理时长. 在小规模的情形下, 不考虑这两个指标, 因为它们之间的差异并不明显. 从表 3 中可以观察到, 在所有大规模 TSP 实例中, 与给定的基线相比, CapsGAT 获得了最短的路径长度. 在 CVRP 实例上进行测试时, 本文的模型在较大规模实例上的性能改进更为显著. 具体而言, CapsGAT 在 CVRP200 上排名第三, 在 CVRP500 上排名第二, 在 CVRP1000 上排名第一. 特别值得注意的是, 与 CVRP1000 上排名第二的 LIH-DACT 相比, 本文的模型只需要 54 秒的推理时间, 而 LIH-DACT 需要一天以上的时间. 此外, 与 AM 和 MRAM 相比, CapsGAT 的推理时间仅略有增加, 数量级上并没有变化. 结果表

表 3 大规模 TSP 和 CVRP 实例的泛化结果

方法	类型	TSP200			TSP500			TSP1000		
		Tour Len.	Opt. Gap.	Time	Tour Len.	Opt. Gap.	Time	Tour Len.	Opt. Gap.	Time
LKH3	Solver	**10.71**	**0.00**%	3 min	**16.54**	**0.00**%	11 min	**23.13**	**0.00**%	24 min
AM-TSP100	G	11.61	8.40%	1 s	20.09	21.39%	5 s	31.16	34.83%	37 s
MRAM-TSP100	G	11.54	7.75%	1 s	19.78	19.52%	7 s	30.59	32.37%	40 s
POMO-TSP100	ST	11.64	8.68%	1 s	20.31	22.79%	2 s	30.54	32.04%	10 s
LIH-DACT-TSP100	T	12.93	20.73%	1 h	20.74	25.39%	4 h	30.48	31.78%	11 h
AMDKD-AM-TSP100	G	11.53	7.62%	6 s	19.87	20.13%	19 s	30.81	33.20%	1 min
CapsGAT-TSP100	G	**11.52**	**7.56**%	2 s	**19.67**	**18.85**%	10 s	**30.33**	**31.24**%	45 s
方法	类型	CVRP200			CVRP500			CVRP1000		
		Tour Len.	Opt. Gap.	Time	Tour Len.	Opt. Gap.	Time	Tour Len.	Opt. Gap.	Time
LKH3	Solver	**22.21**	**0.00**%	28 min	**48.91**	**0.00**%	1.5 h	**91.47**	**0.00**%	3.5 h
AM-CVRP100	G	24.48	10.22%	1 s	56.29	15.09%	4 s	114.50	25.18%	14 s
MRAM-CVRP100	G	24.40	9.86%	2 s	55.96	14.41%	10 s	109.82	20.06%	49 s
POMO-CVRP100	ST	**23.56**	**6.09**%	2 s	**54.43**	**11.29**%	5 s	111.54	21.94%	12 s
LIH-DACT-CVRP100	T	25.00	12.56%	3.9 h	59.15	20.94%	11 h	106.76	16.72%	30 h
AMDKD-AM-CVRP100	G	24.01	8.10%	9 s	56.97	16.48%	30 s	122.99	34.46%	1.5 min
CapsGAT-CVRP100	G	24.29	9.37%	2 s	55.75	13.98%	11 s	**106.62**	**16.56**%	54 s

注: 加粗数字代表每组中最好的结果.

明, 本文的模型在处理大规模问题时确实具有更好的性能. 因此, 改进注意力机制是提高大规模实例泛化性能的潜在方向.

4.4.3 不同分布 TSP 和 CVRP 实例的泛化性能

如前所述, 训练和测试中的 TSP 和 CVRP 实例都是由均匀分布 $U(0,1)$ 生成的. 在学习类方法中, 与训练数据集相比, 如果测试数据集的数据分布不同, 模型的性能也会受到很大影响. 因此, 研究数据分布变化时的泛化性能是有价值的. 本小节令模型在均匀分布 $U(0,1)$ 的实例上进行训练, 在不同均值的正态分布实例上进行测试. 具体地, 使用服从分布 $N(0.5,0.01)$ (与 $U(0,1)$ 具有相同均值、不同方差)、$N(0.25,0.01)$ 和 $N(0.75,0.01)$ (与 $U(0,1)$ 具有不同均值、不同方差) 的实例进行测试. 为了更清晰地观察城市节点之间的分布差异, 图 6 给出了在 4 种不同分布下含有 200,000 个采样节点 (10,000 个 TSP20 实例) 的散点图. 从图 6 中可以观察到, 正态分布产生的城市节点大多集中在 $(\mu-3\sigma,\mu+3\sigma)$, 其中 μ 为均值, σ 为标准差, 而由均匀分布生成的节点在整个约束空间中均匀分布.

表 4 报告了具有不同分布的 TSP50 和 CVRP50 的路径长度的泛化结果. 由于 20 和 100 个节点与 50 个节点具有相似的泛化性能, 因此在此省略它们的结果. 在这个表中, 左边列出的模型都是在均匀分布的数据上训练的. 从表 4 中容易得出如下结论: (1) 当测试集在所有 TSP 和 CVRP 实例上遵循不同的分布时, CapsGAT 的性能优于 AM 和 MRAM; (2) 由于正态分布的数据点更集中, 更接近均值, 因此正态分布实例的路径长度远短于均匀分布实例. 通过更仔细的观察, 发现 CapsGAT 在大部分情形下都优于大多数基线, 这表明设计专门的注意力机制可以在一定程度上增强分布鲁棒性. 然而, 本文的 CapsGAT 还没有超过 AMDKD-AM. 这是因为后者基于精心设计的分布鲁棒结构, 并且其训练数据集已包含多个示例分布, 如均匀分布、正态分布和它们的混合分布等. 在未来的工作中, 为了进一

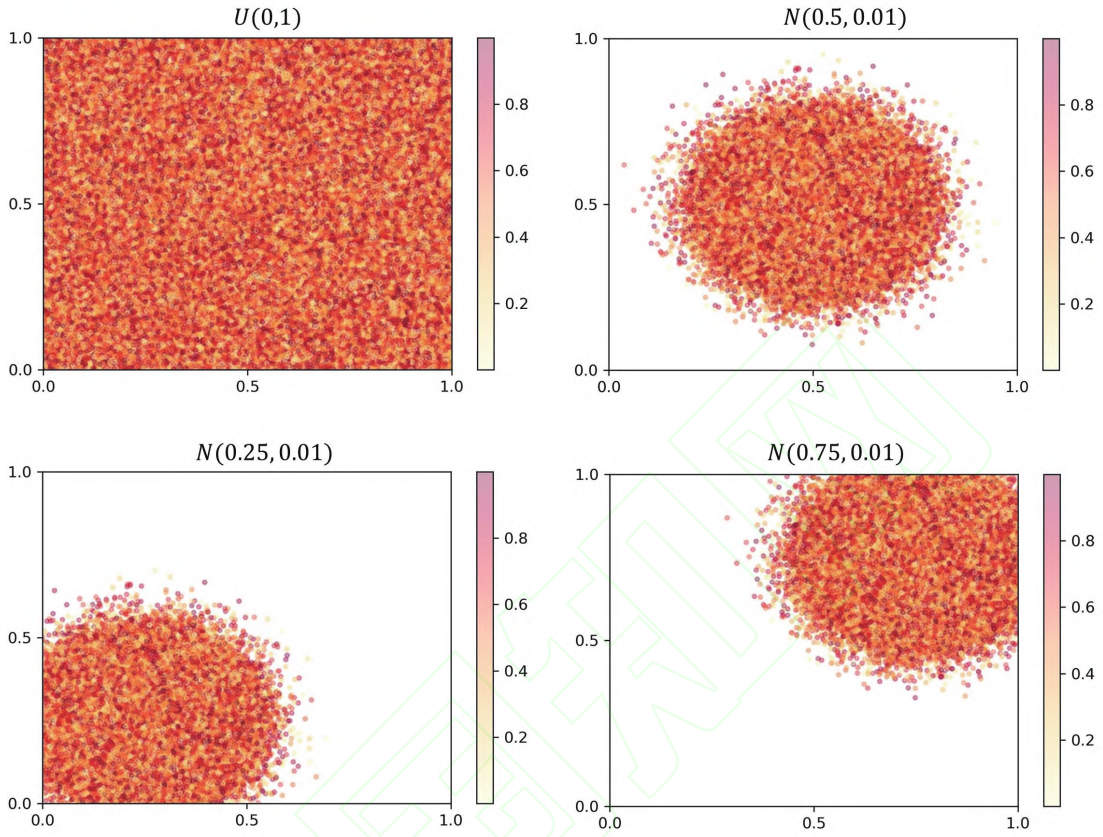

图 6　不同分布城市节点的散点图

表 4　不同分布 **TSP** 和 **CVRP** 实例的泛化结果

方法	类型	$N(0.5, 0.01)$	$N(0.25, 0.01)$	$N(0.75, 0.01)$
AM-TSP50	G	2.79	2.82	2.85
MRAM-TSP50	G	2.72	2.78	2.76
POMO-TSP50	ST	2.71	2.74	2.75
LIH-DACT-TSP50	T	2.68	2.50	2.92
AMDKD-AM-TSP50	G	**2.48**	**2.49**	**2.48**
CapsGAT-TSP50	G	2.71	2.75	2.74
AM-CVRP50	S	4.15	4.20	4.19
MRAM-CVRP50	S	4.13	4.17	4.15
POMO-CVRP50	ST	4.21	4.28	4.28
LIH-DACT-CVRP50	T	4.11	4.12	4.12
AMDKD-AM-CVRP50	S	**4.00**	**4.05**	**4.04**
CapsGAT-CVRP50	S	4.09	4.11	4.09

注: 加粗数字代表每组中最好的结果.

步提高模型的分布泛化能力, 我们也计划研究如何将注意力机制与分布鲁棒技术相结合, 在本文中暂不进行更深一步的探讨.

表 5 在 TSPLIB 和 CVRPLIB 上的泛化结果

实例	Opt.	OR-Tools	AMDKD-AM	AM	MRAM	POMO	LIH-DACT	CapsGAT
a280	2579	2713⋆	3872	3976	3684	3882	4295	**3553**
berlin52	7542	7945⋆	9478	8694	8211	9629	9931	**7998**
d198	15780	15963⋆	31622	79302	36645	30776	**24935**	29280
eil51	426	436	427⋆	436	434	438	493	**432**
eil76	538	561	542	544	552	550	541	**539**⋆
kroA100	21282	21448⋆	21650	32614	27388	31359	31449	**24645**
kroA150	26524	27592	27354⋆	38397	34022	37906	40847	**30689**
kroA200	29368	29741⋆	31112	44717	41600	44095	48415	**35580**
pr76	108159	111104⋆	131202	112665	112099	136692	132182	**111261**
pr124	59030	62519	60042⋆	63504	62937	63207	62826	**62272**
pr152	73682	75834⋆	78120	88741	84221	**83888**	87424	83904
tsp225	3916	4046⋆	4888	5132	5092	5185	5471	**5057**
u159	42080	45778	42944⋆	46254	45803	**43827**	47552	44557
X-n101-k25	27591	29405⋆	30782	37582	32069	31523	38906	**30232**
X-n120-k6	13332	14242	14162	14270	14221	15137	14099	**14096**⋆
X-n143-k7	15700	17470	16509	17398	16994	17047	**16373**⋆	16935
X-n167-k10	20557	22477	21468⋆	22374	**21598**	21987	22483	21976
X-n186-k15	24145	26017	25526⋆	25885	26077	25721	27450	**25617**
X-n200-k36	58578	61009⋆	62254	78839	65013	64165	63989	**63948**

注: 加粗数字代表每组中最好的结果, ⋆ 表示除最优解之外的最好结果.

4.4.4 公共数据集上的性能

本文在公共基准数据集 TSPLIB (Traveling Salesman Problem LIBrary)[5] [52] 和 CVRPLIB (Capacitated Vehicle Routing Problem LIBrary)[6] [60] 的若干问题实例上进一步评估了模型的泛化性能. 因为这些实例遵循与本文在训练中使用的实例完全不同的分布, 如节点位置模式、客户需求和车辆容量, 该测试任务极具挑战性. 表 5 报告了 TSPLIB 和 CVRPLIB 上的路径长度的泛化结果. 将本文所提出的模型与最优解、OR-Tools 和 5 个学习类基线进行了比较, 其中包括在 TSP100 和 CVRP100 上训练的具有最佳分布鲁棒性的 AMDKD-AM. AM、MRAM 和 AMDKD-AM 在测试过程中使用了 1,280 个样本的采样解码策略.

对于 TSPLIB, 随机选择 13 个节点少于 300 的对称 TSP 实例. 从表 5 中首先观察到, 尽管本文的 CapsGAT 是在均匀分布的合成数据上训练的, 但它仍然与 OR-Tools 求解器表现相当甚至更好. 其次, 我们发现本文所提出的模型在所有情形下都比 AM 和 MRAM 更好, 在大多数情形下比 POMO 和 LIH-DACT 更好, 这表明本文的模型具有更强的泛化能力. 通过更仔细的观察, 发现在 eil76 上, 本文的模型获得了除最优解之外的最好的结果. 尽管如此, 不得不承认, 在大多数情形下, 本文的 CapsGAT 与 OR-Tools 和 AMDKD-AM 相比仍然存在差距. 这是合理的, 因为对于基于深度学习的模型来说, 除非利用一些分布鲁棒性技术并在一些有代表性的分布实例上训练模型 [9], 否则实现良好的分布外泛

5) Http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html.
6) Http://vrp.galgos.inf.puc-rio.br/index.php/en/.

化几乎是不可能的 (参见文献 [15, 65]). 对于 CVRPLIB, 随机选择大小在 100 到 200 之间的 6 个实例. 每个实例都根据不同的仓库定位、客户定位和需求分布生成. 可以得到一个类似的结论, 即除了 AMDKD-AM 之外, 本文的模型比 AM、MRAM、POMO 和 LIH-DACT 都要好, 并发现 CapsGAT 在 X-n120-k6 上获得了除最优解之外的最佳结果.

4.5 消融实验

4.5.1 CapsGAT 不同组件的比较

本小节进行了模型中每个组件的消融实验. 具体而言, 首先分别考虑编码和解码阶段的组件. 在编码阶段, 本文使用了胶囊聚合注意力. 在解码阶段, 本文在上下文向量中添加了未访问的节点信息. 为了方便起见, 将它们分别表示为 c-a-a 和 r-dc. 本文进行了以下消融实验, 包括不同模块相互组合的 4 种情形:

- 无 c-a-a, 无 r-dc (原始 AM): AM 模型是本文的 CapsGAT 所基于的基本架构, 不采用胶囊聚合注意力和修正的解码上下文;
- 有 c-a-a, 无 r-dc (AM + c-a-a): AM + c-a-a 是采用胶囊聚合注意力的模型, 而不考虑修改后的解码上下文;
- 无 c-a-a, 有 r-dc (AM + r-dc): AM + r-dc 是采用了修正的解码上下文的模型, 而不考虑胶囊聚合注意力;
- 有 c-a-a, 有 r-dc (CapsGAT): CapsGAT 是本文提出的模型, 其中采用了胶囊聚合注意力和修正的解码上下文.

表 6 报告了不同配置的模型在 TSP50、TSP100、CVRP50 和 CVRP100 上的路径长度和最优差距. 从实验结果中可以观察到: (1) 胶囊聚合注意力和修正解码上下文策略都有助于提高原始 AM 模型的性能; (2) 当这两个组件分别使用时, 胶囊聚合注意力比使用未访问节点修改解码上下文更有效.

表 6 CapsGAT 不同组件的对比结果

方法	组件			TSP50		TSP100	
	c-s	s-s	r-dc	Tour Len.	Opt. Gap.	Tour Len.	Opt. Gap.
AM				5.73	0.52%	7.94	2.26%
AM + c-a-a	✓	✓		5.71	0.40%	7.91	1.88%
AM + r-dc			✓	5.72	0.50%	7.93	2.19%
CapsGAT	✓	✓	✓	5.71	0.38%	7.90	1.83%
AM + c-s	✓			5.71	0.41%	7.91	1.92%
AM + c-s + s-s	✓	✓		5.71	0.40%	7.91	1.88%
方法	组件			CVRP50		CVRP100	
	c-s	s-s	r-dc	Tour Len.	Opt. Gap.	Tour Len.	Opt. Gap.
AM				10.62	2.38%	16.24	3.89%
AM + c-a-a	✓	✓		10.60	2.14%	16.12	2.81%
AM + r-dc			✓	10.62	2.35%	16.22	3.70%
CapsGAT	✓	✓	✓	10.59	2.02%	16.11	2.75%
AM + c-s	✓			10.61	2.29%	16.14	2.93%
AM + c-s + s-s	✓	✓		10.60	2.14%	16.12	2.81%

然后, 将胶囊聚合注意力额外划分为两个组件, 即胶囊聚合注意力子层 (c-s) 和软门控胶囊选择器 (s-s), 并考虑以下两种情形:

• 有 c-s, 无 s-s, 无 r-dc (AM + c-s): AM + c-s 是具有胶囊聚合注意力子层的改进模型, 但没有采用软门控胶囊选择器.

• 有 c-s, 有 s-s, 无 r-dc (AM + c-s + s-s): AM + c-s + s-s 是同时具有胶囊聚合注意力子层和软门控胶囊选择器的改进模型. 此设置与 AM+c-a-a 相同, 为更好地区分每个组件, 本文在此处将其重命名.

相应的结果也列于表 6 中. 从中可以观察到, 与原始 AM 相比, 引入胶囊聚合注意力子层具有显著的效果. 此外, 添加软门控胶囊选择器可以进一步提高 AM 的性能. 就模型性能改进的幅度而言, 胶囊聚合注意力子层比软门控胶囊选择层发挥着更重要的作用.

4.5.2 与常见聚合的改进方法比较

一般地, 增加网络的复杂性能够提高模型性能. 因此, 我们将本文所提出的胶囊聚合注意力与一种简单的改进方法进行对比, 即非线性全连接聚合注意力. 我们在 TSP50、TSP100、CVRP50 和 CVRP100 上相继测试了在多头注意力子层的最终注意力输出中添加 1、3 和 5 个全连接层的实验. 设置的非线性激活函数是 ReLU 函数. 对于一个全连接层的情形, 将其尺寸设置为 128, 以使其与原始输出保持一致. 对于 3 个全连接层的情形, 将每个全连接层的尺寸分别设置为 128、256 和 128. 对于 5 个全连接层的情形, 将每个全连接层的尺寸分别设置为 128、256、512、256 和 128. 将这些结果与 CapsGAT 和 AM+c-s 进行比较, 后者是本文的 CapsGAT 不考虑软门控胶囊选择器和修改的解码上下文的情形. 实验结果展示在表 7 中.

从表 7 中可以首先观察到, 与添加非线性全连接层相比, 本文提出的胶囊聚合注意力显著提高了原始 AM 模型的性能, 表明其具有更强的特征提取能力. 其次, 当使用非线性全连接层来聚合多头注意力时, 模型的性能并没有随着全连接层数量的增加而持续提高. 以 TSP 为例, 当只添加一个全连接层时, 模型的性能只得到了轻微的提高. 当全连接层数增加到 3 层时, 模型的性能得到了进一步的提

表 7 AM 模型的胶囊聚合注意力和非线性全连接聚合注意力的比较结果

方法	类型	TSP50			TSP100		
		Tour Len.	Opt. Gap.	Time	Tour Len.	Opt. Gap.	Time
AM	G	5.79	1.68%	2 s	8.12	4.53%	5 s
AM + 1FC-layer	G	5.79	1.68%	2 s	8.12	4.49%	5 s
AM + 3FC-layer	G	5.78	1.64%	2 s	8.10	4.34%	5 s
AM + 5FC-layer	G	5.80	1.90%	2 s	8.12	4.63%	6 s
AM + Caps-layer	G	5.78	1.51%	2 s	8.07	4.05%	7 s
CapsGAT	G	5.77	1.44%	2 s	8.05	3.72%	7 s
AM	S	5.73	0.52%	10 min	7.94	2.26%	23 min
AM + 1FC-layer	S	5.73	0.52%	10 min	7.94	2.09%	23 min
AM + 3FC-layer	S	5.72	0.51%	13 min	7.92	1.96%	24 min
AM + 5FC-layer	S	5.73	0.62%	14 min	7.96	2.58%	24 min
AM + Caps-layer	S	5.71	0.41%	15 min	7.91	1.92%	29 min
CapsGAT	S	5.71	0.38%	15 min	7.90	1.83%	29 min

(续表)

方法	类型	CVRP50			CVRP100		
		Tour Len.	Opt. Gap.	Time	Tour Len.	Opt. Gap.	Time
AM	G	10.98	5.78%	2 s	16.76	6.89%	6 s
AM + 1FC-layer	G	10.98	5.75%	2 s	16.75	6.80%	6 s
AM + 3FC-layer	G	10.99	5.85%	2 s	16.69	6.44%	6 s
AM + 5FC-layer	G	11.02	6.14%	2 s	16.70	6.51%	6 s
AM + Caps-layer	G	10.95	5.48%	4 s	16.64	6.09%	7 s
CapsGAT	G	10.92	5.21%	4 s	16.62	5.98%	7 s
AM	S	10.62	2.38%	20 min	16.24	3.89%	51 min
AM + 1FC-layer	S	10.62	2.31%	20 min	16.22	3.44%	51 min
AM + 3FC-layer	S	10.66	2.70%	20 min	16.18	3.19%	52 min
AM + 5FC-layer	S	10.66	2.73%	21 min	16.19	3.25%	52 min
AM + Caps-layer	S	10.61	2.29%	25 min	16.14	2.93%	54 min
CapsGAT	S	10.59	2.02%	25 min	16.11	2.75%	54 min

高. 然而, 当使用 5 个全连接层时, 性能非但没有提高, 反而急剧下降, 甚至比原来的 AM 模型更差. 这种性能下降源于训练数据不足和模型过于复杂导致的过拟合现象. 对于推理时间, 一些简单非线性全连接层的堆叠的确比 CapsGAT 稍快, 但不同模型的推理时间基本在同一水平上. 考虑到本文的模型产生的解的质量更高, 可以断言 CapsGAT 优于简单的非线性全连接层.

4.6 可视化实验

为了直观地观察模型决策过程, 并比较本文的 CapsGAT 模型与 AM 模型之间的差异, 本文在 TSP100 上进行了可视化实验. 图 7 展示了 CapsGAT 和 AM 在贪婪解码下逐节点生成路径的过程. 步数和相应的子路程长度标记在每张图的顶部. 从图 7 中可以看到, AM 模型通常从下到上构建路径, 这与在文献 [36] 中观察到的实验现象一致. 与之相比, 本文的 CapsGAT 生成的路径更具灵活性, 它专

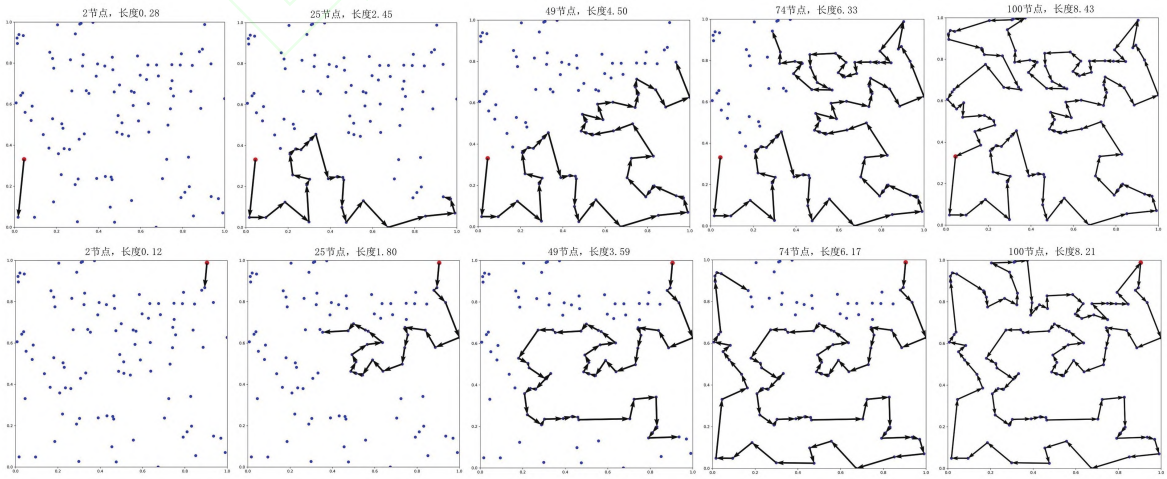

图 7 **AM (上) 和 CapsGAT (下) 节点选择过程在同一 TSP100 实例上的可视化. 步数和相应的子路径长度标记在每张图的顶部**

注于如何找到更短的路径, 而不是遵循特定的模式. 本文的模型的最终的路径长度比 AM 模型更短, 表明了 CapsGAT 的有效性.

5 结论

本文提出了一种新的图注意力网络 CapsGAT 来求解 VRP. 据我们所知, 这是首个将胶囊网络应用于组合优化领域的研究. 新模型基于编码器 - 解码器结构, 在编码过程中, 为更好地聚合信息并提高性能, 本文设计了一种新的胶囊聚合注意力机制, 并给出了一个软门控胶囊选择器来区分多余的胶囊以避免误导训练. 在解码过程中, 本文额外修改了上下文节点向量的表示, 以进一步稳定学习过程. 为避免对高质量标签的需求, 本文在强化学习框架下训练模型. 为测试 CapsGAT 的性能, 本文在两类典型的 VRP, 包括 TSP 和 CVRP 上进行了大量的数值实验. 结果表明, 在大多数情形下, 本文的模型优于启发式基线和基于学习的基线, 并进一步缩小了与最优解的差距.

在未来的研究中, 我们主要关注于如何进一步增强学习类模型的分布鲁棒性. 到目前为止, 大多数工作都是在均匀分布的实例上训练模型, 而在包含各种不同分布的真实世界数据集上测试时, 训练的模型仍然与最优解存在显著差距. 因此, 如何正确地将分布鲁棒技术与学习类模型相结合是值得研究的. 通常, 一个直观的想法是, 从不同的分布中采样实例, 然后将它们提供给模型. 然而, 经过实验发现, 训练实例数量需要成倍增长, 导致训练的时间成本显著增加. 此外, 模型需要不断更新可训练参数, 以最大限度地减小在不同分布实例上的损失, 这会导致训练过程不稳定. 因此, 我们计划设计一个基于 Wasserstein 距离正则化的分布鲁棒模型, 该模型可以学习对分布不敏感的节点嵌入, 以提高泛化性能.

致谢 作者感谢审稿人提出的修改意见.

参考文献

1 Afshar P, Heidarian S, Naderkhani F, et al. COVID-CAPS: A capsule network-based framework for identification of COVID-19 cases from X-ray images. Pattern Recogn Lett, 2020, 138: 638–643

2 Arulkumaran K, Deisenroth M P, Brundage M, et al. Deep reinforcement learning: A brief survey. IEEE Signal Process Mag, 2017, 34: 26–38

3 Azi N, Gendreau M, Potvin J Y. An exact algorithm for a vehicle routing problem with time windows and multiple use of vehicles. European J Oper Res, 2010, 202: 756–763

4 Bahdanau D, Cho K, Bengio Y. Neural machine translation by jointly learning to align and translate. arXiv:1409.0473, 2014

5 Baker B M, Ayechew M A. A genetic algorithm for the vehicle routing problem. Comput Oper Res, 2003, 30: 787–800

6 Baldacci R, Christofides N, Mingozzi A. An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. Math Program, 2008, 115: 351–385

7 Bello I, Pham H, Le Q V, et al. Neural combinatorial optimization with reinforcement learning. arXiv:1611.09940, 2016

8 Bengio Y, Lodi A, Prouvost A. Machine learning for combinatorial optimization: A methodological tour d'horizon. European J Oper Res, 2021, 290: 405–421

9 Bi J, Ma Y, Wang J, et al. Learning generalizable models for vehicle routing problems via knowledge distillation. In: Proceedings of the 35th International Conference on Neural Information Processing Systems. Montreal: Curran Associates, 2022, 31226–31238

10 Chaudhari S, Mithal V, Polatkan G, et al. An attentive survey of attention models. ACM Trans Intell Syst Tech, 2021, 12: 1–32

11 Cho K, Courville A, Bengio Y. Describing multimedia content using attention-based encoder-decoder networks. IEEE Trans Multimedia, 2015, 17: 1875–1886

12 Christofides N. Worst-case analysis of a new heuristic for the travelling salesman problem. Oper Res Forum, 2022, 3: 20

13 Christofides N, Mingozzi A, Toth P. State-space relaxation procedures for the computation of bounds to routing problems. Networks, 1981, 11: 145–164

14 Clarke G, Wright J W. Scheduling of vehicles from a central depot to a number of delivery points. Oper Res, 1964, 12: 568–581

15 Cobbe K, Klimov O, Hesse C, et al. Quantifying generalization in reinforcement learning. In: Proceedings of the 36th International Conference on Machine Learning. New York: PMLR, 2019, 1282–1289

16 Dantzig G, Fulkerson R, Johnson S. Solution of a large-scale traveling-salesman problem. Oper Res, 1954, 2: 393–410

17 Desrochers M, Desrosiers J, Solomon M. A new optimization algorithm for the vehicle routing problem with time windows. Oper Res, 1992, 40: 342–354

18 Deudon M, Cournut P, Lacoste A, et al. Learning heuristics for the TSP by policy gradient. In: Proceedings of Integration of Constraint Programming, Artificial Intelligence, and Operations Research. Cham: Springer, 2018, 170–181

19 Dou Z Y, Tu Z, Wang X, et al. Dynamic layer aggregation for neural machine translation with routing-by-agreement. AAAI, 2019, 33: 86–93

20 Du Y, Zhao X, He M, et al. A novel capsule based hybrid neural network for sentiment classification. IEEE Access, 2019, 7: 39321–39328

21 Duarte K, Rawat Y, Shah M. Videocapsulenet: A simplified network for action detection. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. Montreal: Curran Associates, 2018, 7610–7619

22 Galassi A, Lippi M, Torroni P. Attention in natural language processing. IEEE Trans Neural Netw Learn Syst, 2021, 32: 4291–4308

23 Gendreau M, Laporte G, Musaraganyi C, et al. A tabu search heuristic for the heterogeneous fleet vehicle routing problem. Comput Oper Res, 1999, 26: 1153–1173

24 Gillett B E, Miller L R. A heuristic algorithm for the vehicle-dispatch problem. Oper Res, 1974, 22: 340–349

25 Gong J, Qiu X, Wang S, et al. Information aggregation via dynamic routing for sequence encoding. In: Proceedings of the 27th International Conference on Computational Linguistics. Stroudsburg: Association for Computational Linguistics, 2018, 2742–2752

26 Gu S, Feng Y. Improving multi-head attention with capsule networks. In: Proceedings of Natural Language Processing and Chinese Computing. Cham: Springer, 2019, 314–326

27 Guo M H, Xu T X, Liu J J, et al. Attention mechanisms in computer vision: A survey. Comp Vis Media, 2022, 8: 331–368

28 Hassin R, Keinan A. Greedy heuristics with regret, with application to the cheapest insertion algorithm for the TSP. Oper Res Lett, 2008, 36: 243–246

29 He K, Zhang X, Ren S, et al. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Piscataway: IEEE, 2016, 770–778

30 Helsgaun K. An extension of the Lin-Kernighan-Helsgaun TSP solver for constrained traveling salesman and vehicle routing problems. Technical Report. Roskilde: Roskilde University, 2017

31 Hinton G E, Krizhevsky A, Wang S D. Transforming auto-encoders. In: Proceedings of the 21st International Conference on Artificial Neural Networks and Machine Learning. Berlin: Springer, 2011, 44–51

32 Hottung A, Tierney K. Neural large neighborhood search for the capacitated vehicle routing problem. arXiv:1911.09539, 2019

33 Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Proceedings of the 32nd International Conference on Machine Learning. New York: PMLR, 2015, 448–456

34 Khalil E, Dai H, Zhang Y, et al. Learning combinatorial optimization algorithms over graphs. In: Proceedings of the 30th International Conference on Neural Information Processing Systems. Montreal: Curran Associates, 2017, 6348–6358

35 Kingma D, Ba J. Adam: A method for stochastic optimization. arXiv:1412.6980, 2014

36 Kool W, Van Hoof H, Welling M. Attention, learn to solve routing problems. arXiv:1803.08475, 2018

37 Kwon Y D, Choo J, Kim B, et al. Pomo: Policy optimization with multiple optima for reinforcement learning. In: Proceedings of the 33rd International Conference on Neural Information Processing Systems. Montreal: Curran Associates, 2020, 21188–21198

38 Laporte G. Fifty years of vehicle routing. Transp Sci, 2009, 43: 408–416

39 Laporte G, Mercure H, Nobert Y. An exact algorithm for the asymmetrical capacitated vehicle routing problem. Networks, 1986, 16: 33–46

40 Laporte G, Nobert Y. A branch and bound algorithm for the capacitated vehicle routing problem. Oper Res Spektrum, 1983, 5: 77–85
41 LeCun Y, Bengio Y, Hinton G. Deep learning. Nature, 2015, 521: 436–444
42 Lenstra J K, Kan A H G R. Complexity of vehicle routing and scheduling problems. Networks, 1981, 11: 221–227
43 Li J, Yang B, Dou Z Y, et al. Information aggregation for multi-head attention with routing-by-agreement. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). Stroudsburg: Association for Computational Linguistics, 2019, 3566–3575
44 Lin T, Wang Y, Liu X, et al. A survey of transformers. AI Open, 2022, 3: 111–132
45 Liu Z, Lin Y, Cao Y, et al. Swin transformer: Hierarchical vision transformer using shifted windows. In: Proceedings of the IEEE/CVF international conference on computer vision. Piscataway: IEEE Computer Society, 2021, 10012–10022
46 Ma Q, Ge S, He D, et al. Combinatorial optimization by graph pointer networks and hierarchical reinforcement learning. arXiv:1911.04936, 2019
47 Ma Y, Li J, Cao Z, et al. Learning to iteratively solve routing problems with dual-aspect collaborative transformer. In: Proceedings of the 34th International Conference on Neural Information Processing Systems. Montreal: Curran Associates, 2021, 11096–11107
48 Mazyavkina N, Sviridov S, Ivanov S, et al. Reinforcement learning for combinatorial optimization: A survey. Comput Oper Res, 2021, 134: 105400
49 Nazari M, Oroojlooy A, Snyder L, et al. Reinforcement learning for solving the vehicle routing problem. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. Montreal: Curran Associates, 2018, 9861–9871
50 Parisotto E, Song F, Rae J, et al. Stabilizing transformers for reinforcement learning. In: Proceedings of the 37th International Conference on Machine Learning. New York: PMLR, 2020, 7487–7498
51 Raganato A, Tiedemann J. An analysis of encoder representations in transformer-based machine translation. In: Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP. Stroudsburg: Association for Computational Linguistics, 2018, 287–297
52 Reinelt G. TSPLIB—A traveling salesman problem library. ORSA J Comput, 1991, 3: 376–384
53 Rennie S J, Marcheret E, Mroueh Y, et al. Self-critical sequence training for image captionling. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Piscataway: IEEE Computer Society, 2017, 7008–7024
54 Rosenkrantz D J, Stearns R E, Lewis II P M. An analysis of several heuristics for the traveling salesman problem. SIAM J Comput, 1977, 6: 563–581
55 Sabour S, Frosst N, Hinton G E. Dynamic routing between capsules. In: Proceedings of the 30th International Conference on Neural Information Processing Systems. Montreal: Curran Associates, 2017, 3856–3866
56 Shi R, Niu L. A brief survey on learning based methods for vehicle routing problems. Procedia Comput Sci, 2023, 221: 773–780
57 Shi R, Niu L, Zhou R. Sparse CapsNet with explicit regularizer. Pattern Recogn, 2022, 124: 108486
58 Sutton R, McAllester D, Singh S, et al. Policy gradient methods for reinforcement learning with function approximation. In: Proceedings of the 12th International Conference on Neural Information Processing Systems. Cambridge: MIT Press, 1999, 1–7
59 Toth P, Vigo D. The Vehicle Routing Problem. Philadelphia: SIAM, 2002
60 Uchoa E, Pecin D, Pessoa A, et al. New benchmark instances for the capacitated vehicle routing problem. European J Oper Res, 2017, 257: 845–858
61 Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need. In: Proceedings of the 30th International Conference on Neural Information Processing Systems. Montreal: Curran Associates, 2017, 6000–6010
62 Vinyals O, Fortunato M, Jaitly N. Pointer networks. In: Proceedings of the 28th International Conference on Neural Information Processing Systems. Montreal: Curran Associates, 2015, 2692–2700
63 Wang Q, Tang C. Deep reinforcement learning for transportation network combinatorial optimization: A survey. Knowledge-Based Syst, 2021, 233: 107526
64 Williams R J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Mach Learn, 1992, 8: 229–256
65 Wu Y, Song W, Cao Z, et al. Learning improvement heuristics for solving routing problems. IEEE Trans Neural Netw Learn Syst, 2022, 33: 5057–5069
66 Xu Y, Fang M, Chen L, et al. Reinforcement learning with multiple relational attention for solving vehicle routing problems. IEEE Trans Cybern, 2021, 52: 11107–11120

67 Yuan Z, Li G, Wang Z, et al. RL-CSL: A combinatorial optimization method using reinforcement learning and contrastive self-supervised learning. IEEE Trans Emerg Top Comput Intell, 2023, 7: 1010–1024
68 Zhang J, Shi X, Xie J, et al. GaAN: Gated attention networks for learning on large and spatiotemporal graphs. In: Proceedings of 34th Conference on Uncertainty in Artificial Intelligence. Corvallis: AUAI Press, 2018, 339–349
69 Zhao W, Ye J, Yang M, et al. Investigating capsule networks with dynamic routing for text classification. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. Stroudsburg: Association for Computational Linguistics, 2018, 3110–3119

Capsule aggregated attention for vehicle routing problems

Ruiyang Shi, Lingfeng Niu & Yuhong Dai

Abstract Deep learning-based methods have shown great potential for solving the vehicle routing problem (VRP) in recent years. In the current learning-based models, attention mechanism plays an important role and becomes one of the key modules for improving the performance. However, the aggregate-by-summation paradigm of attention is not expressive enough to fully capture the rich information in the VRP. To solve this problem, we propose a novel capsule aggregated attention mechanism, which utilizes capsule to store more information, applies dynamic routing for information aggregation, exploits soft gated capsule selector to differentiate the importance of different capsules, and modifies the context node vector in the decoding process to reflect the state changes. Based on the above modifications, we design a new graph attention network for solving the VRP under the reinforcement learning framework in this paper, and validates the effectiveness of our proposed method through sufficient numerical experiments.

Keywords **vehicle routing problem, capsule network, attention mechanism**

MSC(2020) **65Z05, 68T20, 68Q32, 90C35**

doi: 10.1360/SCM-2023-0307