计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2024,33(3):146−157 [doi: 10.15888/j.cnki.csa.009435]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于图神经网络的最大化代数连通度算法①

夏春燕[1], 侯新民[1,2,3,4]

[1](中国科学技术大学 大数据学院, 合肥 230026)
[2](中国科学技术大学 数学科学学院, 合肥 230026)
[3](中国科学院 吴文俊数学重点实验室, 合肥 230026)
[4](合肥国家实验室, 合肥 230088)
通信作者: 夏春燕, E-mail: chunyanxia@mail.ustc.edu.cn

摘　要: 随着智能体数量的增加, 多智能体系统中潜在的通信链路数量呈指数级增长. 过多冗余链路的存在给系统带来了大量的能源浪费和维护成本, 而盲目地去除链路又会降低系统的稳定性和安全性. 代数连通度是衡量图连通性的重要指标之一. 然而, 传统的半正定规划 (SDP) 方法和启发式算法在求解大规模场景下的最大化代数连通度问题时非常耗时. 在本文中, 我们提出了一种监督式的图神经网络模型来优化多智能体系统的代数连通度. 我们将传统的 SDP 方法应用于小规模任务场景中, 得到足够丰富的训练样本和标签. 在此基础上, 我们训练了一个图神经网络模型, 该模型可用于更大规模的任务场景中. 实验结果表明, 当需要去除 15 条边时, 我们的模型的平均性能达到了传统 SDP 方法的 98.39%. 此外, 我们的模型计算时间极其有限, 可以推广到实时场景中去.

关键词: 多智能体系统; 代数连通度; 图神经网络; 半正定规划; 舍入技术; 控制研究; 机器学习

引用格式: 夏春燕,侯新民.基于图神经网络的最大化代数连通度算法.计算机系统应用,2024,33(3):146–157. http://www.c-s-a.org.cn/1003-3254/9435.html

Algorithm for Maximizing Algebraic Connectivity Based on Graph Neural Network

XIA Chun-Yan[1], HOU Xin-Min[1,2,3,4]

[1](School of Data Science, University of Science and Technology of China, Hefei 230026, China)
[2](School of Mathematical Sciences, University of Science and Technology of China, Hefei 230026, China)
[3](Wu Wen-tsun Key Laboratory of Mathematics, Chinese Academy of Sciences, Hefei 230026, China)
[4](Hefei National Laboratory, Hefei 230088, China)

Abstract: As the number of agents increases, the number of potential communication links in a multi-agent system grows exponentially. Excessive redundant links lead to a significant waste of energy and maintenance costs for the system, while blindly removing links will reduce the stability and security of the system. Algebraic connectivity is one of the important metrics to measure the connectivity of a graph. However, traditional semidefinite programming (SDP) methods and heuristic algorithms for maximizing algebraic connectivity in large-scale scenarios are time-consuming. This study proposes a supervised graph neural network model to optimize the algebraic connectivity of multi-agent systems. The study applies the traditional SDP method in small-scale task scenarios, obtaining a sufficient amount of diverse training samples and labels. Based on this, it trains a graph neural network model that can be used in larger-scale task scenarios. The experimental results indicate that when removing 15 edges, the proposed model achieves an average performance of 98.39% of the traditional SDP method. In addition, the model has extremely limited computational time and can be extended to real-time scenarios.

Key words: multi-agent system; algebraic connectivity; graph neural network (GNN); semidefinite programming; rounding technique; control study; machine learning

① 基金项目: 国家自然科学基金 (12071453); 量子通信与量子计算机重大项目 (2021ZD0302902)
收稿时间: 2023-08-19; 修改时间: 2023-10-20; 采用时间: 2023-11-03; csa 在线出版时间: 2024-01-18
CNKI 网络首发时间: 2024-01-19

网络 (或图) 的代数连通度是其拉普拉斯矩阵的第二最小特征值[1]. 航空运输网络[2,3], 电子物流网络[4], 电流回路网络[5]以及其他网络都可以从代数连通度中受益. 这对于多智能体系统的一致性研究也是至关重要的. 对于一阶协议, 代数连通度越大, 收敛速度越快[6]. 对于二阶协议, 代数连通度仍然是影响收敛速度的关键因素. 此外, 网络结构中边的数量越多, 表明代数连通度越大, 对应的多智能体系统更加稳健且高效.

然而, 系统中存在过多的通信链路也是浪费的, 因为它会导致能源消耗、多余的链路维护和重复计算成本的问题[7]. 特别是在大规模多智能体系统中, 当每对可通信智能体之间都保持通信时, 通信链路的数量与小规模多智能体系统相比呈指数级增长, 由此产生的多余资源消耗更为显著. 然而, 一些链路的存在对于系统来说是不必要的, 删除它们对代数连通度几乎没有影响[8]. 因此, 如何去除一些相对冗余的链路, 同时保持尽可能大的代数连通度是多智能体系统领域非常感兴趣的问题. 这实际上也是图论中最大化代数连通度的问题. 我们感兴趣的是如何删除指定数量的通信链路以减少消耗, 同时使代数连通度尽可能大.

在学习可适用于最大化代数连通度问题的配置时, 选择适当的模型架构至关重要. 最大化代数连通度问题是优化图的拓扑结构问题. 普通的神经网络模型使用向量或者矩阵表示输入数据, 并且通过线性变换或者非线性激活函数来处理数据, 无法捕捉到图中节点之间的关系以及边的权重, 更无法进行图层面的计算. 另外普通神经网络只能接受固定维度的输入和输出, 这使得模型只能适用于同一种规模的任务场景, 一旦图中的节点数量增加或减少, 原来训练的模型将无法使用, 只能重新训练.

相比普通的神经网络, 图神经网络可以借助图结构的信息并且在图上进行卷积操作, 还能灵活处理不同大小和形状的图结构数据. 这一点是非常重要的, 因为我们希望通过小规模任务场景训练得到的模型可直接使用在大规模任务场景中, 以克服最大化代数连通度问题随着图规模的增加带来的计算时间急剧增加的问题. 另外, 图神经网络的计算图和多智能体系统之间的收敛协议有一定程度的类似. 多智能体系统可以表述为图的形式, 智能体对应图中的节点, 通信链路对应图中的边. 多智能体系统的收敛协议涉及信息的传递和交流, 每个智能体信息的迭代更新是基于能与之通信的智能体进行的, 即相邻的智能体. 图神经网络中每个节点的信息传递与更新也是基于邻居节点的信息. 两者的信息更新均是基于邻居信息, 每个节点或智能体的状态都会影响网络或系统中的其他节点或智能体, 进而影响整个网络或系统的性能.

因此, 我们考虑利用图神经网络模型来解决最大化代数连通度问题. 我们的模型设计旨在衡量系统中每条通信链路的存在对于整个网络的连通能力的贡献大小, 再根据模型的输出删去贡献较小的链路的同时尝试避开关键的通信链路. 基于此, 本文提出了一种基于图神经网络 (GNN) 模型的最大化代数连通度方法. 借助传统的优化算法, 我们生成了一系列的样本. 通过学习这些样本, 我们训练得到的模型能够快速推理出相对冗余的链路. 与现有方法相比, 我们的方法在保证代数连通度尽可能大的同时克服了随着智能体数量增长带来的巨大计算成本, 可以直接应用于大规模的多智能体系统中.

1 相关工作

1.1 最大化代数连通度

最大化代数连通度问题是 NP 难的问题[9], 目前还没有高效的方法可以精确地求解. Kim 等人考虑了加权图的最大化代数连通度问题[10]. 他们设定每条边的距离大于或等于一个常数, 并且边的权重由这个距离的函数决定. 针对这一约束优化问题, 他们提出了一种基于半正定规划 (SDP) 求解器的迭代算法. Rafiee 等人考虑了多智能体系统中的网络拓扑结构的两个问题[11]. 一是在限定的通信链路数量下使网络结构尽快收敛, 二是在最低成本下找到恰当的网络拓扑结构可以使网络满足所需的基本性能. 他们将这两个关于寻找最佳通信图的问题表述为一个混合整数半正定规划 (MISDP) 问题. Dai 等人将动态网络的最优拓扑设计问题分为 3 类[12]. 一是无权图的网络设计, 二是边的权重为两个端点位置相关的函数的网络设计, 三是需要保持连接的移动网络系统的设计. 他们将上述 3 类问题转化为 MISDP 问题和混合整数二次约束规划 (MIQCP) 问题, 采用求解器进行求解. 上述文献对于代数连通度问题的求解都是基于求解器进行的, 受求解器的限制, 只能用于小规模任务场景的求解.

有很多关于最大化代数连通度的启发式算法被提出. Ghosh 等人考虑的是向给定图中添加一条边使得

代数连通度最大化的问题[13]. 他们提出了一个基于费德勒向量的贪婪扰动启发式算法. Kim 考虑了向给定的图中添加一条边使得代数连通度最大化的问题[14], 针对这个问题给出了一个基于长期方程的高效二分算法. Sydney 等人考虑了如何重新布置网络中的一条边来提高网络的代数连通度问题[15], 即从原图中删除一条边之后再重新添加一条边. Wei 等人向航空运输网络中引入了加权的代数连通度[3], 并且使用了禁忌搜索算法对最大化代数连通问题求解. Wei 等人通过最大化代数连通度的方法优化了航空运输网络结构[16]. 他们首先建立了一个有约束的 SDP 问题, 然后为解决这个问题提出了舍入技术. Li 等人提出了通过添加图中的最小度的顶点和与这个顶点相距最远的顶点之间的边来最大化代数连通度[17]. 上述算法经常应用于各种网络的结构优化中. Trimble 等人使用贪婪扰动算法、二分法、禁忌搜索法和 SDP 方法构建了无人机网络连通性跟踪算法[18]. Cheung 等人采用最大化代数连通度方法解决了受损电子网络的恢复问题[4], 并使用了贪婪扰动算法、禁忌搜索算法和舍入技术.

贪婪扰动算法作为一种基于费德勒向量的求解算法, 求解速度快, 但是性能相对较差. 而最小度最远距离算法作为改进的贪婪扰动算法, 对于图结构的要求更加严格. 除了这两种算法, 上面提及的启发式算法相较于基于求解器的方法计算时间有所提升, 但是随着任务规模的增大, 求解问题花费的时间仍然是十分可观的. 因此, 我们希望找到一种可以快速解决最大化代数连通度问题的方法, 在面对大规模任务场景时有着高效的求解速度以及可靠的性能.

1.2 图神经网络

图神经网络首次由 Scarselli 等人提出[19], 在最近几年发展出了很多不同图神经网络架构. Kipf 等人提出了图卷积神经网络以从图中提取特征结构[20], 完成各种基于图的任务, 包括链接预测、节点分类和图分类等任务. 然而图卷积神经网络无法计算训练过程中没有出现的顶点的嵌入. Hamilton 等人提出了图采用与聚合模型[21], 这是一种能够利用顶点的信息高效产生未知顶点的嵌入的方法. Veličković等人提出了图注意力网络[22], 它利用了掩码自注意力层, 获取每个节点的领域特征的同时给该领域中的每个节点分配了不同的权重, 这种模型的优点在于不需要提前知晓图的结构信息的同时运算成本减少了. You 等人提出了一个通用的图神经网络设计空间[23]. 借助于这个设计空间, 可以快速找到适合具体任务的图神经网络模型的学习架构. 这些图神经网络架构也被应用到了很多领域, 包括社交网络[24,25]、交通工程[26]和生物领域[27,28]等.

图神经网络在这些领域的应用出现了颠覆传统方法的性能和表现. 通过建立多层的神经网络和设计不同的模型架构, 图神经网络能够从多种多样的数据中提取出模型需要的特征, 使得许多难以解决的问题变得可能. 然而目前还没有利用图神经网络模型来解决最大化代数连通度算法的工作. 因此, 本文提出了一种监督式图神经网络模型来求解此问题. 为了保证模型生成解集的质量, 本文选择了基于松弛半正定规划的舍入技术来生成大量小规模任务场景的解作为训练标签. 另外, 随着规模的增长, 图神经网络模型的求解时间几乎不变. 也就是说, 该方法既保证了在大规模任务场景中的求解速度, 也兼顾了求解质量.

2 问题定义

代数连通度随着图中边的数量的增加而增加. 然而对于多智能体系统而言, 有一部分通信链路的存在对于代数连通度的提升贡献不大, 删除这些冗余链路还可以降低系统的损耗. 因此, 在本文中要考虑的问题是, 对于一个多智能体系统而言, 如何在删除指定数量的边的同时以使得代数连通度尽可能地大.

2.1 多智能体系统的图表述

考虑这样一个有 n 个智能体的多智能体系统, 每个智能体 i 的位置表示为 $x_i \in R^2$. 在一般情况下, 每一对智能体之间的通信链路的强度在一定距离内是个常数, 然后在超过一定阈值后迅速消失, 这个消失速度是指数级的[29]. 并且, 在多智能体系统的网络中, 边的权重通常被限制在[0, 1]上. 一个智能体 i 和智能体 j 中的通信强度的一般函数是:

$$w_{ij} = w(x_i, x_j) = f(d_{ij}) \tag{1}$$

其中, $f: R_+ \to R_+$, $d_{ij} = \|x_i - x_j\|_2$. 函数 f 是关于距离 d_{ij} 的递减函数, 这里采取文献[10]中的设定, 即:

$$f(y) = \begin{cases} 1, & y \leqslant \rho_1 \\ \epsilon^{\frac{\rho_1 - y}{\rho_1 - \rho_2}}, & y > \rho_1 \end{cases} \tag{2}$$

其中, $0 < \rho_1 < \rho_2$, $0 < \epsilon < 1$ (图 1). 这个函数 f 表明, 在两个智能体的距离小于 ρ_1 时通信强度是 1, 并且在大于 ρ_1 时通信强度迅速消失直到达到距离 ρ_2. 当距离大于 ρ_2 时信号几乎为 0.

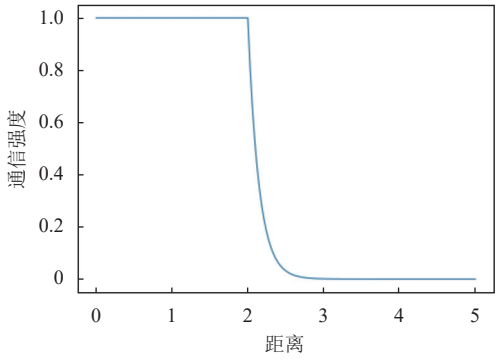


图 1 函数f, 其中 $\rho_1=2$, $\rho_2=3$, $\epsilon=0.001$

这样的一个多智能体系统可以用一个加权的无向图来表示. 系统中的每个智能体对应于图中的一个节点. 对于任意一对智能体, 如果它们之间可以通信, 那么这对智能体对应的顶点之间存在边, 否则边不存在. 对应的图可以表示为$G=(V, E, w)$, 这是一个无向、加权的连通图, 其中$V=\{v_1, v_2, \cdots, v_n\}$是具有 n 个顶点的集合, E 是边的集合, $w\in R^{|E|}$是权重集合. 则该图的邻接矩阵 A 是一个 $n\times n$ 的对称矩阵. 如果顶点v_i和顶点v_j是连通的, 则$(A)_{ij}=w_{ij}$, 否则为$(A)_{ij}=0$. 度矩阵 D 是一个 $n\times n$ 对角矩阵, 其中的每个对角元素$(D)_{ii}$ 表示的是节点 i 的度, 即$(D)_{ii}=\sum_{j=1}^{n}(A)_{ij}$. 那么图 G 的拉普拉斯矩阵定义为$L=D-A$, 是一个实对称的半正定矩阵.

此外, 拉普拉斯矩阵还可以表示为边向量的外积和[13]. 对于边集 E 中的每条边$e=(i, j)$, 边向量h_e的第 i 项值为 1, 第 j 项值为−1, 其余项均为 0. 那么拉普拉斯矩阵也可以写成$L=\sum_{e\in E}w_e h_e h_e^{\mathrm{T}}$, 其中$w_e$是$w$中对应于边 e 的值.

2.2 最大化代数连通度问题

图 G 的代数连通度是拉普拉斯矩阵的第 2 小特征值, 用$\lambda_2(L)$表示[1]. 如果代数连通度为 0, 这意味着图是非连通的, 即在对应的多智能体系统中, 至少存在一对智能体无法直接或者间接通信. 图的代数连通度越高, 意味着要删除更多的边使图变成非连通图, 这样多智能体系统更加稳定. 在本文的设定中, 距离小于 ρ_2时, 这对智能体之间可以相互通信. 如果任意一对智能体之间都可以直接通信, 代数连通度虽然是最大的状态, 但是会造成能源的浪费. 因此可以选择关闭一些通信链路以减轻系统负荷. 另外, 如果多智能体系统对应的图是非连通图, 那么无论怎样修改每条通信链路的状态, 代数连通度始终为 0. 在本文中, 默认多智能体系统对应的图一定是连通图.

在此框架下, 我们需要考虑的问题是:

$$\begin{cases}\max \quad \lambda_2\left(L_0-\sum\limits_{e\in E}x_e w_e h_e h_e^{\mathrm{T}}\right)\\ \text{s.t.}\begin{cases}\sum\limits_{e\in E}x_e=b\\ x\in\{0,1\}^{|E|}\end{cases}\end{cases} \tag{3}$$

其中, b 是给定的正整数, L_0是初始图 G 的拉普拉斯矩阵. 在对初始图 G 删除 b 条边之后, 可以减少系统的负荷的同时仍然使得代数连通度尽可能地大.

3 问题求解与样本生成

我们所使用的 GNN 模型是通过监督学习获得的. 因此一个完整的 GNN 模型需要明确定义的任务场景和相应的解集作为输入. 每一组这样的输入构成一个样本, 而我们需要大量的样本来进行模型训练, 以达到较好的效果. 然而问题 (3) 是一个有$C_{|E|}^{b}$种可能性的组合问题, 如果直接采取暴力求解的方式, 即便只生成一个样本都需要大量的时间, 大量样本的生成需要的时间无法估量. 因此, 在本节中, 我们首先将问题 (3) 转换成一个松弛的半正定规划问题, 这样可以通过求解器求解得到该问题的松弛解. 然后采用舍入技术, 得到该问题的最终解. 根据这个解, 我们可以获得求解需要的训练标签.

3.1 问题求解

对于问题 (3), 第 1 步是将该问题转化为松弛问题, 即将每条边是否存在的二元变量扩展为在[0, 1]上的连续变量. 问题 (3) 的松弛问题为:

$$\begin{cases}\max \quad \lambda_2\left(L_0-\sum\limits_{e\in E}x_e w_e h_e h_e^{\mathrm{T}}\right)\\ \text{s.t.}\begin{cases}\sum\limits_{e\in E}x_e=b\\ x\in[0,1]^{|E|}\end{cases}\end{cases} \tag{4}$$

显然问题 (4) 比问题 (3) 拥有更大的可行集, 因此问题 (4) 的解是问题 (3) 的上界. 这个松弛问题可以重新写成以下的松弛半正定规划问题:

$$\begin{cases}\max \quad s\\ \text{s.t.}\begin{cases}L_0-\sum\limits_{e\in E}x_e w_e h_e h_e^{\mathrm{T}}\succcurlyeq s\left(I_n-\dfrac{11^{\mathrm{T}}}{n}\right)\\ \sum\limits_{e\in E}x_e=b\\ x\in[0,1]^{|E|}\end{cases}\end{cases} \tag{5}$$

这里的松弛半正定规划问题 (5) 和松弛问题 (4) 是等价关系, 其中 s 为移除边之后的代数连通度[30]. 使用 MOSEK 求解器可以求解松弛半正定规划问题 (5). 在这个松弛的 SDP 问题中, 解是在[0, 1]之间的连续值, 因此需要进一步选择边.

3.2 样本生成

我们采取文献[16]中提到的舍入技术中的逐步法来选择边. 即每次选择一条解向量中值最大的边, 从边集 E 中删除, 存入有序集合 S 中, 并更新拉普拉斯矩阵和约束 b, 然后再对问题 (6) 进行求解, 重复以上步骤直到约束 b 为 0. 最后, 我们获得了一个包含了 b 条边的有序集合 S. 在这里, 对于给定的约束 b, 问题 (5) 求解得到的边集合只能针对一个约束. 一旦约束 b 发生改变, 标签不再正确, 训练的模型失效. 为了模型的可扩展性, 我们在此处设定 $b = |E| - (|V| - 1)$, 其中 $|V| - 1$ 是保证一个无向图连通的最少的边, b 的值恰好是边集 E 中保证图连通的情况下可以删除的最多的边的数量. 完整的流程如算法 1 所示.

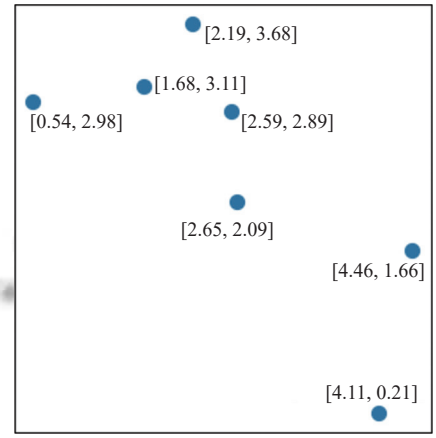
算法 1. SDP-Step[16]

输入: $G=(V, E, w)$, b
输出: 有序集合 S

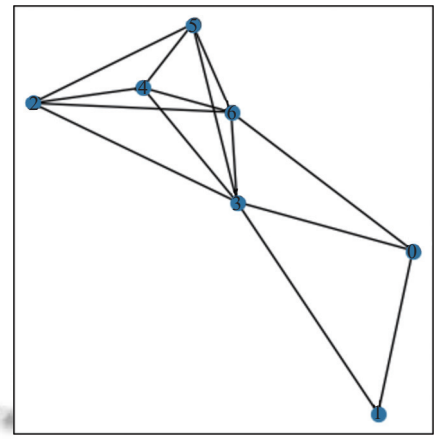
1) $S=\varnothing$
2) while $b>0$ 且 $E\neq\varnothing$ do
3) 求解问题 (5) 得到解向量 x
4) $e^*=\arg\max_{e\in E} x_e$
5) $b=b-1$
6) 移除 e^* 并更新问题 (5)
7) $E=E\backslash\{e^*\}$
8) $S=S\cup\{e^*\}$
9) end while

我们使用算法 1 中的优化专家来计算边的删除次序. 在这个算法的基础上, 我们可以生成大量的随机任务场景, 并且根据算法 1 对问题进行求解, 从而生成训练模型需要的标签.

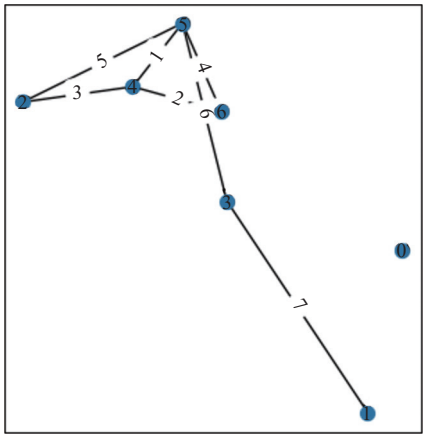
图 2(a) 显示了在大小为[0, 5]×[0, 5]的二维平面上随机生成的多智能体系统, 图 2(a) 上标示了系统中每个智能体的二维坐标. 这里我们假设 $\rho_1=2$, $\rho_2=3$, $\epsilon=0.001$, 根据每个智能体的二维坐标以及式 (1), 可以计算出每一对智能体之间的通信强度, 并得到对应的无向加权图 $G=(V, E, w)$. 在图 2(b) 显示了该多智能体系统对应的图 G. 使用算法 1 进行求解, 可以获取一个包含 b 条边的有序集合 S. 如图 2(c) 所示, 算法 1 给出的有序集合 S 是{(4, 5), (4, 6), (2, 4), (5, 6), (2, 5), (3, 5), (1, 3)}. 至此, 一个随机的任务场景对应的图 G 和有序集合 S 构成了一个样本. 根据本节的方法, 我们可以生成大量的样本用以训练模型.



(a) 任务场景生成



(b) 转换成无向加权图



(c) 生成有序集合 S

图 2 单个样本的生成示例

4 模型框架

本文提出的模型是基于文献[23]中提出的图神经网络设计空间构建的. 整个模型框架如图 3 所示, 首先通过第 3 节的方法生成大量的任务场景以及对应的有序集合. 据此训练好我们的模型, 然后根据模型的边选择策略来移除边. 在本节中, 首先, 我们对该设计空间进行简要的阐述. 然后, 我们给出了图的初始节点特征设定. 在此设计空间的基础上, 我们给出了模型的具体模型架构. 之后, 根据每个样本中的无向图$G=(V, E, w)$和有序集合 S, 给出了模型的损失函数.

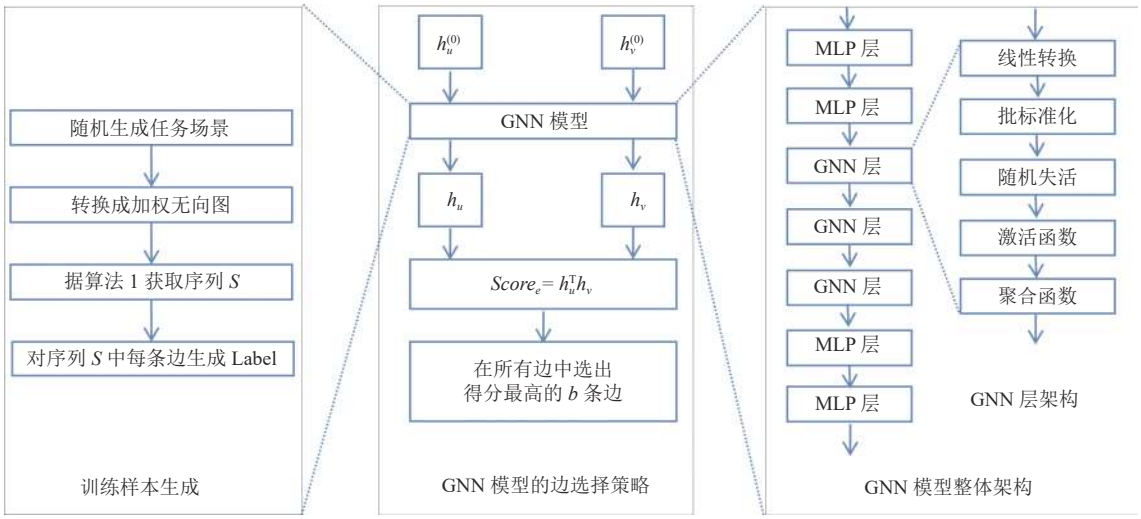


图 3 GNN 模型框架

4.1 图神经网络设计空间

图神经网络设计空间有 3 个维度, 即层内设计、层间设计以及训练配置[23]. 层内设计是图神经网络架构的核心, 是整个模型中的消息传递层. 对于层内设计, 第 k 个图神经网络层可以被定义为:

$$\begin{cases} h_{u,\text{mid}}^{(k)} = Dropout\left(BN\left(W^{(k)}\left(w_{uv}h_u^{(k)}+b^{(k)}\right)\right)\right) \\ h_v^{(k+1)} = AGG\left(\left\{ACT\left(h_{u,\text{mid}}^{(k)}\right), u\in N(v)\right\}\right) \end{cases} \tag{6}$$

其中, $h_v^{(k)}$ 是节点 v 的第 k 层嵌入, w_{uv} 是节点 u 和节点 v 之间的权重, 在本文中含义是智能体 u 和智能体 v 之间的通信强度. $W^{(k)}$、$b^{(k)}$ 是可训练的权值, $N(v)$ 是 v 的局部邻域. 式 (6) 表示的是在第 k 次消息传递过程中的信息加工. 在第 k 次消息传递中, 每个节点 v 接收来自邻居节点 u 的信息$h_u^{(k)}$与(u,v)这条边的权重. 首先, 对每个邻域的信息进行线性转换. 然后可以选择是否进行批标准化 (*BN*) 和随机失活 (*Dropout*). 对当前信息采用激活函数 *ACT*, 激活函数一般有 ReLU、PReLU、Swish 等函数. 将所有来自邻域的信息通过聚合函数 *AGG* 进行聚合, 得到当前节点的新信息. 聚合函数是对输入的信息排列可置换的函数, 比如对所有的信息的平均、求和或者最大化运算等.

层间设计包括预处理层、后处理层和信息传递层的数量的选择. 训练配置方面包括批大小、学习率、优化器和训练周期. 更详细的说明可以参考文献[23].

4.2 节点的初始特征设定

在设计 GNN 模型的过程中, 需要输入的主要变量是节点特征和边的权重. 边的权重由两个相邻智能体之间的通信强度决定. 根据任务场景, 多智能体系统通常规模不定, 因此必须确保当智能体数量在一定范围内时模型是有效的. 为了使 GNN 模型可扩展, 节点的初始特征维度必须与图中的节点数量无关. 在本文中, 我们将每个节点 u 的初始特征向量的第 v 个元素定义为:

$$h_{u,v}^{(0)} = \begin{cases} 1, & (u,v)\in E \\ 0, & (u,v)\notin E \text{ or } v>n \end{cases} \tag{7}$$

其中, E 是图的边集, n 是图的顶点数量. 当节点 u 与节点 v 之间有边连接时, 特征向量的第 v 个元素为 1, 如果没有连接, 则为 0. 特征向量中第 n 个元素之后都用 0 进行填充, 以确保模型对一定规模内的图都可以计算. 那么向 GNN 模型输入的整体的特征矩阵为$H^{(0)}=[A;0]$, 这是一个 $n\times N$ 的矩阵, 其中 A 也是图 G 对应的邻接矩阵. 在本文中, 我们设定 N=50.

4.3 学习架构

在使用图神经网络对节点特征进行处理时, 一种有效的方法是在 GNN 模型之前和之后添加多层感知机 (MLP) 层进行处理. 通过这种方式, 在保持输入特征的基本信息的同时, 可以降低 GNN 模型中需要学习的参数数量, 减少模型的复杂度, 并且减轻模型训练的压力. 因此, 预处理层和后处理层的引入是必要的. 分别设定为两个 MLP 层, 每个 MLP 层的参数不是共享的. 通过实验, 我们发现 GNN 模型层数设定为 3 层是比较合适的, 每一层 GNN 模型中的线性层参数不一致, $W^{(k)}$、$b^{(k)}$的具体值由训练得出. 之后的批标准化 (BatchNorm)、随机失活 (Dropout)、激活函数 (Activation)、聚合函数 (Aggregation) 参数设定见表 1. 关于训练配置的参数设定见表 2. 该 GNN 模型的整体架构如图 3 右侧所示.

表 1 层内设计的参数设定

批标准化	随机失活	激活函数	聚合函数
是	0.3	PReLU	MEAN

表 2 训练配置的参数设定

批大小	学习率	优化器	训练周期
32	0.01	Adam	200

4.4 预测层

经过上述模型我们只能计算出每个节点的最终节点特征. 然后, 我们的目标是在避开比较关键的通信链路的同时选择出一些不那么重要的通信链路. 因此, 我们还需要通过预测层来选择出这些边.

针对图中已经存在的每一条边$e=(u,v)$, 我们可以计算出节点 u 和节点 v 的节点特征的内积, 并将其作为边 e 对于整个图的代数连通度的贡献度的得分. 边 e 的得分被定义为:

$$score_e = h_u^{\mathrm{T}} h_v \tag{8}$$

其中, h_u和h_v是 GNN 模型对节点 u 和节点 v 的输出. 在获得每条边的得分后, 可以通过选择得分最高的 b 条边来确定 GNN 模型的最终结果.

4.5 损失函数

损失函数主要包括两个部分, 一部分是针对有序集合 S 中的每条边. 另一部分是针对不在有序集合 S 中但在边集 E 中的边. 对于有序集合 S 中的边, 从第 2 节中我们知道越是排在前面的边, 相对于整个图来说相对没那么重要, 是可以先舍弃的. 相反, 对于那些相对靠后的边, 我们越是需要谨慎删除. 对于那些不在有序集合 S 中但在边集 E 中的边, 哪怕只是删除一条都对代数连通度的影响都很大.

首先, 我们将有序集合 S 中的每条边的次序都对应到一个标签, 对应关系定义为:

$$label_e = 1 - \frac{rank_e}{b} \tag{9}$$

其中, b 是问题中的约束, $rank_e$是边 e 在有序集合 S 中的次序. 当算法 1 将边 e 视为较不重要的时候, 比如 $rank_e = 0$时, $label_e = 1$, 它是所有的边对应的标签中值最大的.

我们将均方误差损失函数作为其优化目标函数, 定义为:

$$Loss_S = \sum_{e\in S}\left(\left(1-\frac{rank_e}{b}\right)-score_e\right)^2 \tag{10}$$

然后, 对于那些不在有序集合 S 中但在边集 E 中的边, 我们希望模型预测出来的贡献度得分的绝对值越小越好, 因此, 我们将这部分损失函数定义为:

$$Loss_{E\setminus S} = \sum_{e\in E\setminus S} score_e^2 \tag{11}$$

综合损失函数如下:

$$Loss = Loss_S + Loss_{E\setminus S} \tag{12}$$

5 仿真实验与结果分析

在本节中, 我们将对训练得到的 GNN 模型进行仿真实验, 并将实验结果与传统算法进行对比. 任务场景限制在$[0, 5]\times[0, 5]$的平面上. 首先在平面上随机生成一个智能体, 之后每一个智能体都生成在当前所有智能体的可通讯范围内, 从而保证多智能体系统对应的图一定是连通的. 根据式 (1) 的通信模型计算边的权重. 设定 $\rho_1 = 2$, $\rho_2 = 3$, $\epsilon = 0.001$. 我们的 GNN 模型是在第 4.1 节生成的 500k 样本集上训练得到的, 其任务空间中有 20 个智能体. 为了更全面地评估 GNN 模型的性能, 我们使用传统算法构建解集进行性能比较. 传统算法包括贪婪扰动算法[13]、二分法算法[14]、禁忌搜索算法[2,31], 以及算法 1 (SDP-Step 算法). 对于禁忌搜索算法, 禁忌表的长度和迭代次数对搜索结果有显著影响, 参考文献[4], 设定迭代次数设置为 3 000, 禁忌表的长度为 $n/2$, 其中 n 为智能体的数量.

5.1 风车图测试

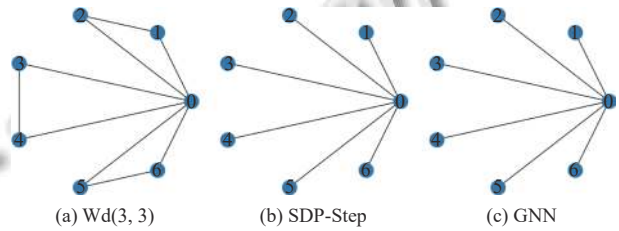
首先, 我们利用风车图对 GNN 模型的性能进行了

简单的测试, 并对 GNN 模型的预测结果进行了解释. 风车图是一个包含 n 个团的图, 每个团共享一个节点, 每个团中有 k 个相互连接的节点. 我们用符号 Wd(n, k) 来表示这种图的结构特征. 风车图有许多冗余边. 在删除与中心节点无关的边时, 风车图的代数连通度保持不变[32,33]. 这为我们直接评估 GNN 模型的性能提供了便利.

对于图 4(a) 中的风车图 Wd(3, 3), 当需要删除 3 条边时, SDP-Step 算法得到的结果为{(1, 2), (3, 4), (5, 6)} (图 4 (b)), 与 GNN 模型的输出相同 (图 4(c)). 然而, 在对图 5(a) 中的 Wd(3, 4) 进行测试时, 可以看出, 当需要删除 4 条边时, SDP-Step 算法选择删除{(4, 5), (7, 8), (1, 3), (2, 3)} (图 5(b)), GNN 模型选择删除{(1, 2), (2, 3), (1, 3), (4, 6)} (图 5(c)). 虽然我们的 GNN 模型给出的边删除方案与 SDP-Step 算法不太一致, 但两者的代数连通度都是 1.0. 因此, 从代数连通度的角度来看, GNN 模型的表现并不逊色于 SDP-Step 算法. GNN 模型的输出与 SDP-Step 算法的结果存在差异, 这是因为在给定的任务空间中, 可能存在多种不同的优化策略. 相较于对比解集的差异性, 采用代数连通度作为指标能更客观地判断 GNN 模型的性能.

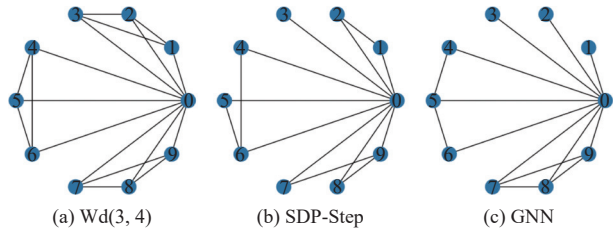
5.2 泛化性

在本节中, 我们采用相同的方式随机生成了 100 个拥有 20 个智能体的任务场景, 并将 GNN 模型的结果与这些传统算法的结果进行了比较. 为了评估删除边的数量对 GNN 模型性能的影响, 我们将可删除边的数量 b 设置为 1, 2, …, 15. 表 3 描述了 GNN 模型和其他算法在预算限制下的平均代数连通度和平均计算时间. 对于每个算法, 第 1 行是代数连通度, 第 2 行是计算时间. 对于代数连通度表现, 我们将最佳结果以加粗和下划线标明, 次好的结果以加粗标明.



(a) Wd(3, 3) (b) SDP-Step (c) GNN

图 4 风车图 Wd(3, 3) 对比



(a) Wd(3, 4) (b) SDP-Step (c) GNN

图 5 风车图 Wd(3, 4) 对比

表 3 GNN 模型与其他算法的代数连通度和平均计算时间对比 (n=20)

算法	指标	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SDP-Step	代数连通度	**1.05**	**1.05**	**1.05**	**1.05**	**1.05**	**1.05**	**1.05**	**1.05**	**1.05**	**1.05**	**1.04**	**1.04**	**1.04**	**1.04**	**1.04**
	平均计算时间 (s)	0.03	0.06	0.08	0.11	0.15	0.17	0.19	0.22	0.25	0.29	0.31	0.33	0.36	0.42	0.49
禁忌搜索	代数连通度	**1.05**	**1.05**	**1.05**	**1.05**	**1.05**	**1.05**	**1.05**	**1.05**	**1.05**	**1.05**	**1.04**	**1.04**	**1.04**	**1.04**	**1.03**
	平均计算时间 (s)	1.08	1.14	1.23	1.42	1.60	1.47	1.51	1.56	1.62	1.70	1.73	1.75	1.83	2.03	2.25
GNN	代数连通度	**1.05**	**1.05**	**1.05**	**1.04**	**1.04**	**1.03**	**1.03**	**1.01**	**1.01**	**1.00**	**0.98**	**0.96**	**0.94**	**0.92**	0.90
	平均计算时间 (s)	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02
贪婪扰动	代数连通度	**1.04**	**1.03**	**1.02**	0.99	0.98	0.96	0.95	0.93	0.92	0.90	0.87	0.86	0.84	0.83	0.82
	平均计算时间 (s)	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
二分法	代数连通度	1.03	1.02	0.99	0.98	0.95	0.94	0.91	0.88	0.89	0.85	0.82	0.77	0.76	0.74	0.62
	平均计算时间 (s)	0.07	0.13	0.18	0.26	0.34	0.38	0.44	0.49	0.54	0.59	0.65	0.77	0.78	0.87	0.96

显然, SDP-Step 算法的求解效果是最佳的. 在这种较小规模的任务场景中, 进行了 3 000 次迭代搜索的禁忌搜索算法表现和 SDP-Step 算法表现几乎一致. 可以发现 GNN 模型的表现不俗, 和性能极佳的 SDP-Step 算法和禁忌搜索算法的表现相差不大, 甚至是比贪婪扰动启发式算法和二分法算法表现更好. GNN 模型的表现没有超越 SDP-Step 算法而是始终略差于它, 这是很自然的. 因为在小规模任务场景中, 随着更多的边被移除, 剩余的边中关键边的比例稳步增长, 这无意中给模型的性能带来更大的挑战. 在这个过程中, 不经意间可能会破坏网络的更多关键边, 从而对模型的性能造成更大的压力.

另外, 我们可以观察到 GNN 模型和其他算法在计算时间上随着要移除的边的数量增加时的变化情况. 与 SDP-Step 算法相比, 禁忌搜索算法需要更多的计算时间, 而二分法算法的计算时间也不可忽视. 只有贪婪

扰动算法的计算时间与 GNN 模型相当; 然而, 该算法的性能表现明显不及 GNN 模型.

因此, 我们的模型是具有一定的泛化能力的, 即便在没有见过的任务场景中, 表现也相对出色. 另外, 在小规模的任务场景中, GNN 模型表现出两个优势: 相对较强的性能和几乎可以忽略的计算时间.

5.3 可扩展性

当任务场景中的智能体的数量增加时, 为 GNN 模型生成新的样本数据以重新训练模型参数成本很高. 此外, SDP-Step 算法是一种基于求解器的方法, 然而随着智能体的数量增加, 求解器的执行时间会显著增加, 从而导致生成新标签的代价变得异常昂贵且不切实际. 因此, 我们旨在通过在小规模任务场景中训练 GNN 模型, 实现其在大规模场景中的泛化应用. 禁忌搜索算法和贪婪扰动算法随着智能体的数量增多需要花费大量时间, 而传统算法 (包括二分法) 的性能都不及 SDP-Step 算法. 因此, 在这种情况下, 我们仅使用 SDP-Step 算法评估 GNN 模型的性能. 将 GNN 模型的代数连通度记为 $\lambda_{2,\mathrm{GNN}}$, 将 SDP-Step 的代数连通度记为 $\lambda_{2,\mathrm{SDP\text{-}Step}}$. 为了进行更直观的性能对比, 我们引入了一个评估指标, 即使用 $\lambda_{2,\mathrm{GNN}}/\lambda_{2,\mathrm{SDP\text{-}Step}}$ 来表示 SDP-Step 与 GNN 之间的比值.

图 6 给出了在拥有 30 个智能体的 100 个随机任务场景下分别删除 1、5、10、15 条边的实验结果分布. 横坐标为 $\lambda_{2,\mathrm{GNN}}/\lambda_{2,\mathrm{SDP\text{-}Step}}$ 的比值区间, 纵坐标为满足该比值区间的任务场景数量. 在移除 1、5、10 和 15 条边后, GNN 模型的平均性能分别达到了 SDP-Step 的 99.98%、99.80%、99.01% 和 98.16%. 图 6 显示, 在这 100 个具有 30 个智能体的随机任务场景上, 当去除最多 5 条边时, GNN 模型在几乎所有任务场景上都达到了 99% 以上, 当去除最多 10 条边时, GNN 模型达到了 95% 以上. 在去掉 15 条边后, 对于其中的 7 个任务场景, GNN 的性能低于 95%. 然而, GNN 模型在剩余任务场景中的 4 个上的表现在 100% 和 110% 之间, 比 SDP-Step 算法表现更优.

同样, 图 7 给出了在拥有 50 个智能体的 100 个随机任务场景下分别删除 1、5、10、15 条边的实验结果分布. 在去除 1、5、10 和 15 条边后, GNN 模型的平均性能分别达到了 SDP-Step 的 99.95%、99.66%、99.23% 和 98.63%. 从图 7 中, 我们可以看到, GNN 模型没有再超越 SDP-Step, 但在移除 10 条边的范围内, GNN 模型几乎在所有情况下都能达到 97% 以上的性能. 当移除 15 条边时, 它的性能几乎达到 95% 以上. 此外, 我们观察到在其中一个任务场景中, 当移除 10 或 15 条边时, GNN 模型的性能略低于 90%. 但是需要注意的是, 与总计 100 个任务场景相比, 这个特定任务场景所占的比例极小, 因此对整体结果的影响非常有限.

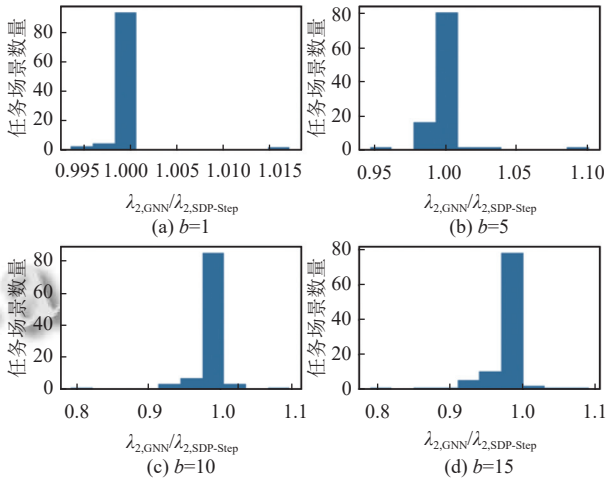


图 6 b=1, 5, 10, 15 时 $\lambda_{2,\mathrm{GNN}}/\lambda_{2,\mathrm{SDP\text{-}Step}}$ 的分布 (n=30)

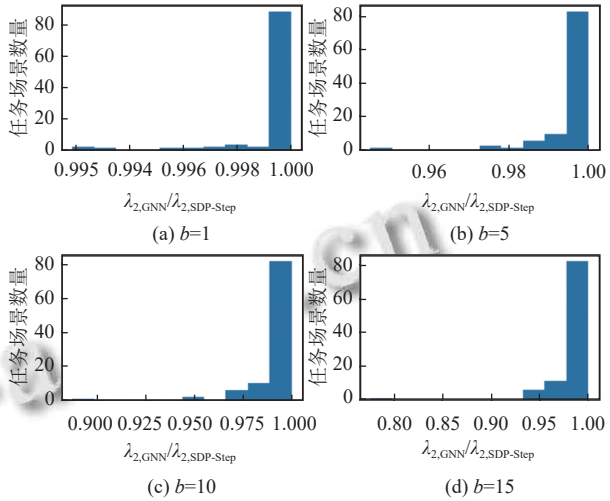


图 7 b=1, 5, 10, 15 时 $\lambda_{2,\mathrm{GNN}}/\lambda_{2,\mathrm{SDP\text{-}Step}}$ 的分布 (n=50)

从这两个实验可以看出, 当任务场景中智能体的数量较大时, GNN 模型的性能同样非常稳定. 在经过小规模任务场景中最大化代数连通度方案的学习之后, GNN 模型能够有效地推理出更大规模任务场景中的最大化策略. 这表明 GNN 模型具备了处理大规模任务场景的能力, 并且对于不同规模的任务场景具有一定的可扩展性.

5.4 大规模任务场景计算时间对比

我们建立 GNN 模型不是为了小规模的任务场景, 而是为了更大规模的任务场景. 尽管 SDP-Step 算法总是能够给出优秀的解决方案, 但随着任务场景中智能

体数量的增加, SDP-Step 算法产生解决方案所需的时间变得漫长, 不适合实时的应用. 相比之下, GNN 模型的计算时间与智能体的数量无关, 可以推广到更大规模的场景. 图 8 比较了 SDP-Step 和 GNN 模型在智能体的数量逐渐增加的情况下, 去除特定数量的边所需的平均计算时间. 与 SDP-Step 的计算时间相比, GNN 模型的计算时间基本上可以忽略不计. 这是因为在 SDP-Step 中, 我们必须重复求解 SDP 问题, 而重复的次数是由要去除的边的数量决定的. 此外, 当工作场景的规模增长时, 解决 SDP 问题需要更长的时间. 然而, 随着智能体数量的增加, GNN 花费的时间几乎没变, 可以推广到实时应用中去.

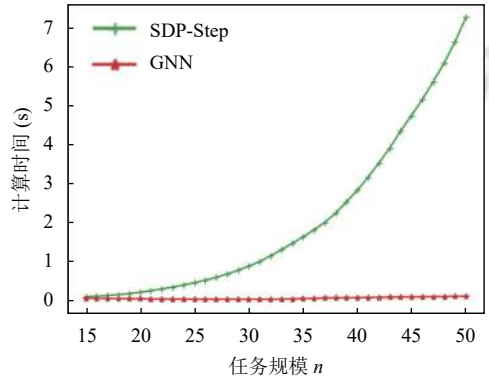


图 8 n=15–50 时 GNN 与 SDP-Step 的计算时间

5.5 参数实验

本文提出的图神经网络的模型框架主要是参考设计空间的架构建立的. 根据实验尝试, 模型的部分参数设定对模型本身的性能没有太大的影响. 但是有些参数对模型的性能影响较大, 包括图神经网络层数、聚合函数、学习率、随机失活和批标准化. 在本节中, 我们通过实验来分析批标准化和随机失活这两个参数的设定与否对模型性能的影响, 以及图神经网络层数、聚合函数和学习率这些参数取不同的值时模型结果的变化. 实验中默认的参数设置为: GNN 层数为 3 层, 聚合函数为 MEAN, 学习率为 0.01, 随机失活为 0.3 以及批标准化为 True. 实验仍然以 SDP-Step 的解作为基准, 即每个模型的代数连通度与 SDP-Step 的代数连通度的比值作为指标衡量模型性能好坏, 测试的任务场景规模为 n=20.

5.5.1 批标准化与随机失活

批标准化和随机失活都是可以有效减少模型过拟合的一种方式. 批标准化是模型对每层的输入都进行了归一化处理, 随机失活则是随机舍弃一部分邻点传递过来的信息. 这里批标准化参数取值范围为{True, False}, 即采用批标准化和不采用批标准化, 随机失活参数取值范围为{False, 0.3, 0.6}, 即不采用随机失活、采用并设为 0.3 和设为 0.6. 为了综合测试这两个参数的设定对模型效果的影响, 对这两组参数取值两两组合, 共进行了 6 次实验. 实验结果如图 9 所示. 横轴标签中“-”左侧是批标准化的取值, T 对应 True, F 对应 False. 右侧对应随机失活的取值, F 对应 False, 0.3 和 0.6 对应本身的值. 纵坐标为模型的代数连通度与 SDP-Step 的代数连通度的比值. 可以看出, 虽然批标准化对模型性能的提升非常有限, 但是采用批标准化比不采用表现确实更好一些. 另外, 一定的随机失活可以使模型的泛化能力更强表现更稳定可靠, 但是取值较大时无法充分获取图上的信息, 模型性能表现较差.

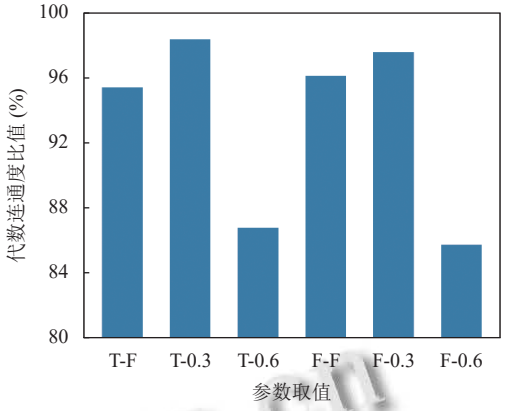


图 9 批标准化与随机失活参数分析

5.5.2 图神经网络层数

当图神经网络层数为一层时, 意味着每个节点只获取了其一阶邻点的信息. 随着图神经网络层数的增加, 每个节点能由近及远获取到图中更多节点的信息. 此处层数取值范围为{1, 2, 3, 4, 5, 6, 7}, 实验结果如图 10 所示. 横坐标为图神经网络的层数, 纵坐标为模型的代数连通度与 SDP-Step 的代数连通度的比值. 从图 10 中可以看出, 图神经网络层数为 1 层时模型效果差很多, 2 层时相对较差. 当层数为 3 层以上时, 模型表现逐渐变差, 但比 1–2 层要好一些. 因此图神经网络层数选择 3 层是合适的.

5.5.3 聚合函数

本文考虑的聚合函数的取值范围为{MAX, SUM, MEAN}. 实验结果如图 11(a) 所示. 横坐标为聚合函数的取值, 纵坐标为模型的代数连通度与 SDP-Step 的代数连通度的比值. 可以看出, 选择 MEAN 作为聚合函

数模型的性能会优于 MAX 和 SUM, 分别高了 5.54% 和 2.55%. 可以推测 MEAN 优于 MAX 是因为本文的问题虽然是链接预测任务, 但是连通度的衡量是基于整个图的, 相比之下 MEAN 获取的邻点的信息更加全面一些. 根据实验结果来看, 聚合函数推荐使用 MEAN.

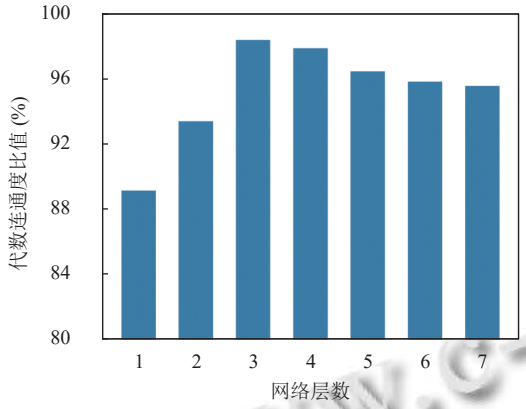


图 10 图神经网络层数参数分析

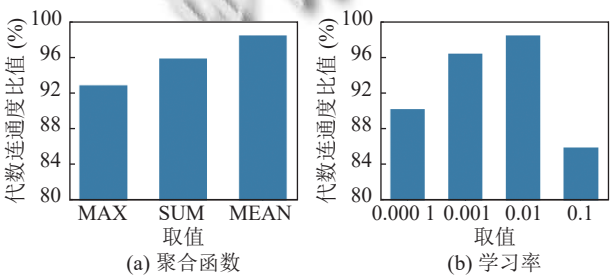


图 11 聚合函数和学习率参数分析

5.5.4 学习率

本文考虑的学习率的取值范围为{0.000 1, 0.001, 0.01, 0.1}. 实验结果如图 11(b) 所示. 横坐标为学习率参数的取值, 纵坐标为模型的代数连通度与 SDP-Step 的代数连通度的比值. 可以发现当学习率较小和较大时, 模型的性能均表现不佳. 当学习率过小时, 模型每次更新的步长更小, 收敛速度慢, 而当学习率过大时, 模型参数在训练过程中波动较大, 难以收敛. 从实验结果来看, 学习率设置为 0.01 是更合适的.

6 结论

本文提出了一种基于图神经网络的最大化代数连通度算法. 该算法学习如何在大量潜在通信链路中去除一部分链路以降低系统损耗的同时保持尽可能大的代数连通度. 传统的优化和启发式算法随着智能体的数量的增加消耗的时间骤增, 而 GNN 模型消耗的时间和计算成本几乎不变. 该 GNN 模型是一种监督模型, 需要使用传统算法生成参考解集. 然而, 对于组合优化问题, 好的解集不是唯一的, 甚至相同性能的两种解集完全不同. 对于一个有监督的模型, 解集中发生一两个边的偏差也可能会导致更糟糕的结果. 一个自然的问题是, 我们是否能找到一个更好的模型, 可以自发地探索最大化代数连通度的问题, 更智能地适应更大规模的任务场景, 同时表现更好. 这是我们未来尝试要解决的问题.

参考文献

1 Fiedler M. Algebraic connectivity of graphs. Czechoslovak Mathematical Journal, 1973, 23(2): 298–305. [doi: 10.21136/CMJ.1973.101168]

2 Lumbanraja EMU, Sugeng KA, Hariadi N. Algebraic connectivity optimization in flight routes addition problem using tabu search method. AIP Conference Proceedings, 2017, 1862(1): 030137. [doi: 10.1063/1.4991241]

3 Wei P, Sun DF. Weighted algebraic connectivity: An application to airport transportation network. IFAC Proceedings Volumes, 2011, 44(1): 13864–13869. [doi: 10.3182/20110828-6-IT-1002.00486]

4 Cheung KF, Bell MGH. Improving connectivity of compromised digital networks via algebraic connectivity maximisation. European Journal of Operational Research, 2021, 294(1): 353–364. [doi: 10.1016/j.ejor.2021.01.015]

5 Goddet E, Retière N, Stojanović V, *et al*. Maximizing the algebraic connectivity of meshed electrical pathways used as current return network. Mathematics and Computers in Simulation, 2019, 158: 18–31. [doi: 10.1016/j.matcom.2018.05.002]

6 Zhu JD. On consensus speed of multi-agent systems with double-integrator dynamics. Linear Algebra and Its Applications, 2011, 434(1): 294–306. [doi: 10.1016/j.laa.2010.08.024]

7 Fang C, Yu FR, Huang T, *et al*. Distributed energy consumption management in green content-centric networks via dual decomposition. IEEE Systems Journal, 2017, 11(2): 625–636. [doi: 10.1109/JSYST.2015.2454231]

8 Chen XZ, Zhang SG, Zhang LJ, *et al*. Determining redundant links of multiagent systems in keeping or improving consensus convergence rates. IEEE Systems Journal, 2022, 16(4): 6153–6163. [doi: 10.1109/JSYST.2021.3130591]

9 Mosk-Aoyama D. Maximum algebraic connectivity augmentation is NP-hard. Operations Research Letters, 2008, 36(6): 677–679. [doi: 10.1016/j.orl.2008.09.001]

10 Kim Y, Mesbahi M. On maximizing the second smallest eigenvalue of a state-dependent graph Laplacian. IEEE Transactions on Automatic Control, 2006, 51(1): 116–120.

[doi: 10.1109/TAC.2005.861710]
11 Rafiee M, Bayen AM. Optimal network topology design in multi-agent systems for efficient average consensus. Proceedings of the 49th IEEE Conference on Decision and Control. Atlanta: IEEE, 2010. 3877–3883. [doi: 10.1109/CDC.2010.5717719]
12 Dai R, Mesbahi M. Optimal topology design for dynamic networks. Proceedings of the 50th IEEE Conference on Decision and Control and European Control Conference. Orlando: IEEE, 2011. 1280–1285. [doi: 10.1109/CDC.2011.6161026]
13 Ghosh A, Body SP. Growing well-connected graphs. Proceedings of the 45th IEEE Conference on Decision and Control. San Diego: IEEE, 2006. 6605–6611.
14 Kim Y. Bisection algorithm of increasing algebraic connectivity by adding an edge. IEEE Transactions on Automatic Control, 2010, 55(1): 170–174. [doi: 10.1109/TAC.2009.2033763]
15 Sydney A, Scoglio C, Gruenbacher D. Optimizing algebraic connectivity by edge rewiring. Applied Mathematics and Computation, 2013, 219(10): 5465–5479. [doi: 10.1016/j.amc.2012.11.002]
16 Wei P, Chen L, Sun D. Algebraic connectivity maximization of an air transportation network: The flight routes' addition/deletion problem. Transportation Research Part E: Logistics and Transportation Review, 2014, 61: 13–27. [doi: 10.1016/j.tre.2013.10.008]
17 Li G, Hao ZF, Huang H, *et al*. Maximizing algebraic connectivity via minimum degree and maximum distance. IEEE Access, 2018, 6: 41249–41255. [doi: 10.1109/ACCESS.2018.2857411]
18 Trimble J, Pack D, Ruble Z. Connectivity tracking methods for a network of unmanned aerial vehicles. Proceedings of the 9th IEEE Annual Computing and Communication Workshop and Conference. Las Vegas: IEEE, 2019. 440–447.
19 Scarselli F, Gori M, Tsoi AC, *et al*. The graph neural network model. IEEE Transactions on Neural Networks, 2009, 20(1): 61–80. [doi: 10.1109/TNN.2008.2005605]
20 Kipf TN, Welling M. Semi-supervised classification with graph convolutional networks. arXiv:1609.02907, 2017.
21 Hamilton WL, Ying Z, Leskovec J. Inductive representation learning on large graphs. Proceedings of the 31st International Conference on Neural Information Processing Systems. Long Beach: Curran Associates Inc., 2017. 1025–1035.
22 Veličković P, Cucurull G, Casanova A, *et al*. Graph attention networks. Proceedings of the 6th International Conference on Learning Representations. Vancouver: ICLR, 2018.
23 You JX, Ying R, Leskovec J. Design space for graph neural networks. Proceedings of the 34th International Conference on Neural Information Processing Systems. Vancouver: Curran Associates Inc., 2020. 1427.
24 Ying R, He RN, Chen KF, *et al*. Graph convolutional neural networks for Web-scale recommender systems. Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. London: Association for Computing Machinery. 2018. 974–983. [doi: 10.1145/3219819.3219890]
25 You JX, Liu BC, Ying R, *et al*. Hierarchical temporal convolutional networks for dynamic recommender systems. Companion Proceedings of the 2019 World Wide Web Conference. San Francisco: ACM, 2019. 2236–2246.
26 Qu ZG, Liu XZ, Zheng M. Temporal-spatial quantum graph convolutional neural network based on Schrödinger approach for traffic congestion prediction. IEEE Transactions on Intelligent Transportation Systems, 2023, 24(8): 8677–8686. [doi: 10.1109/TITS.2022.3203791]
27 Gao JL, Lyu T, Xiong F, *et al*. Predicting the survival of cancer patients with multimodal graph neural network. IEEE/ACM Transactions on Computational Biology and Bioinformatics, 2022, 19(2): 699–709. [doi: 10.1109/TCBB.2021.3083566]
28 Aslan Hİ, Ko H, Choi C. Classification of vertices on social networks by multiple approaches. Mathematical Biosciences and Engineering, 2022, 19(12): 12146–12159. [doi: 10.3934/mbe.2022565]
29 Zavlanos MM, Egerstedt MB, Pappas GJ. Graph-theoretic connectivity control of mobile robot networks. Proceedings of the IEEE, 2011, 99(9): 1525–1540. [doi: 10.1109/JPROC.2011.2157884]
30 Nagarajan H, Rathinam S, Darbha S, *et al*. Algorithms for synthesizing mechanical systems with maximal natural frequencies. Nonlinear Analysis: Real World Applications, 2012, 13(5): 2154–2162. [doi: 10.1016/j.nonrwa.2012.01.010]
31 Glover F. Tabu search —Part I. ORSA Journal on Computing, 1989, 1(3): 190–206. [doi: 10.1287/ijoc.1.3.190]
32 Jafarizadeh S. Fastest mixing reversible Markov chain on friendship graph: Trade-off between transition probabilities among friends and convergence rate. Systems & Control Letters, 2019, 130: 13–22.
33 Jafarizadeh S. Fastest mixing reversible Markov chain: Clique lifted graphs and subgraphs. IEEE Transactions on Signal and Information Processing over Networks, 2020, 6: 88–104. [doi: 10.1109/TSIPN.2020.2964211]

(校对责编: 孙君艳)