# 可行性报告：GNN驱动的调度友好图切分

### 一 研究背景

EMU最终的调度性能（如总调度步数）在前期图切分的过程中无法感知，相对而言是一个黑盒优化问题，因此借助传统的启发式算法，很难在图切分阶段提前优化调度性能。相较而言，最近兴起的机器学习技术擅长捕捉潜在的依赖关系和深层信息，在很多黑盒问题的求解中展现了卓越的性能。因此，本研究希望能借助机器学习图神经网络（GNN）技术的优势，在不进行实际调度的情况下，提前预测感知调度信息，进而利用调度信息提升切分性能以及最终的调度性能。

### 二 前人研究概述

#### 图神经网络（GNN）

图神经网络(GNN)是最近几年兴起的一种基于图结构的神经网络结构，因为其独特的表示能力，而受到广泛学者的关注与研究[[1]](#endnote-1)。传统深度学习模型如卷积神经网络（CNN）在欧几里得空间数据(语言，图像，视频等)上取得了不错的成绩，但是在对非欧几里得空间数据(如电路网表，社交网络等)进行处理上却存在一定的局限性。针对该问题，研究者们引入了图论中抽象意义上的图(Graph)来表示非欧几里得结构化数据，并利用图神经网络对来图数据进行处理，以深入发掘其特征和深层信息。

目前最流行的图神经网络大多遵循迭代的消息传递机制[[2]](#endnote-2)，以捕获一个节点邻域内的结构信息。具体地，令表示一个图，其中是图的节点集合，表示图的边集。现在考虑一个 K 层 GNN，那么其第 k 层的传播可以表示为：

其中是节点v在第k层的表示向量。表示v的邻居节点，AGGREGATE是用于从节点的邻居收集消息的函数，COMBINE 用于将节点的先前表示与其邻域消息组合的函数。

因为在电子自动化设计（EDA）领域中很多原始数据都可以表示成图的形式，因此近年来有越来越多的研究者致力于利用图神经网络来解决EDA里的问题[[3]](#endnote-3)，如在逻辑综合阶段，有研究者提出了一种定制化的图神经网络D-SAGE[[4]](#endnote-4)，该网络可以学习从HLS到FPGA块的映射；在布局规划阶段，谷歌团队提出了Edge-GNN[[5]](#endnote-5),并在其基础上搭建了一个强化学习框架，可以高效高质量地完成芯片宏布局；在布局阶段，北京大学团队[[6]](#endnote-6)受静态时序分析的启发，提出了一个新颖的图神经网络架构，该网络可以快速地完成对绕线后时序表现的评估；在测试阶段，马宇哲教授等人[[7]](#endnote-7)提出可以用图神经网络在芯片设计中插入较少的最佳测试点，同时最大限度地提高故障覆盖率。总的来说，我们希望可以看到图神经网络在EDA领域中的更多应用。

#### 2.2 机器学习驱动的图切分

近年来，随着机器学习相关技术的快速发展，有很多研究者开始尝试用机器学习技术来解决图切分问题。来自谷歌的研究者们首先提出了一个可泛化的近似图切分框架（GAP）[[8]](#endnote-8)，该框架将图切分问题转化为图学习里的节点分类问题，同时提出了一个定制化的损失函数来指导图神经网络参数学习过程。在该损失函数的基础上，来自伯克利大学和剑桥大学的研究者们进一步构造了一个多层图神经网络用于预测图切分结果[[9]](#endnote-9)。该神经网络的灵感来自于多层图切分框架，先利用池化层将原始图不断地粗化到更小规模，直到图的规模小于给定阈值。在获取的最小规模图上，研究者们用图神经网络来生成节点的低维表示向量，然后不断地将节点向量映射回更细化的层次上，同时在每一个粗化层次利用图神经网络来对映射的节点向量做光滑处理，直到映射回原始图。这时网络已经为原始图中每个节点生成了一个低维向量表示，该低维向量会被进一步用于节点分类，从而判断每个节点所属的切分块。总的来说，之前的GNN驱动的图切分方法大多关注一般的图切分问题，目前还没有方法关注调度性能，因此无法直接运用到本项目中，但里面的一些方法思想/网络设计还是值得借鉴的。

### 三 可行性分析

我们在对EMU架构有了一个系统性了解之后，基于我们对机器学习图神经网络技术的认识，提出了两种可能的技术路线以帮助提升调度性能：时步预测和拥塞预测。下面将分别介绍两种路线的任务定义，基本方法，以及可能遇到的挑战或难点等。

#### 3.1 时步预测

时步预测指的是预测每一个节点在最终调度时的执行时步，通俗地讲即预测所有节点最终的调度顺序。总的来说，时步预测的ground-truth信息较为容易获取，在执行完调度之后可以很方便地获取，预测难度也在合理范围之内。

该技术路线的难点在于如何利于预测得到的调度信息（节点的执行时步）以提升最终调度性能。目前，我们有两种可能的利用思路。第一种思路是获取可能发生拥塞的时步，如统计每个时步的节点个数，将节点个数超过上限或给定阈值的时步判定为拥塞。获取到可能发生拥塞的时步信息之后，可以设计一个后处理过程，将拥塞时步中的节点交换到其他P64中，以期减小原P64内的拥塞，进而提升调度性能。这里需要设计一个代价函数，综合考虑拥塞，流通量，是否处于关键路径等信息，综合考量交换后对最终性能带来的可能影响。第二种思路和第一种思路比较接近，都是利用拥塞时间步信息，不过通过一种新的反馈途径。我们会基于拥塞节点，进一步预测拥塞边，然后为拥塞边/节点赋予更小的权重，再将该信息反馈到图切分阶段，重新进行P512->P64的切分，以期可以生成更好的切分结果。

#### 3.2 拥塞预测

拥塞指的是到达硬件中某一部分的消息数量过多，超出该部分硬件资源可处理上限，使得部分消息来不及处理，以致引起这部分乃至整个系统性能下降的现象。具体到我们的任务场景中，以用于信息中转的多路选择器（mux）为例，因为mux容量有限，当同一时间步需要通过某mux发送/接收的消息超出上限时，会在该mux处形成拥塞，导致部分通过该mux发送/接收的消息的延迟增加。若被阻塞的消息位于关键路径上，可能会进一步降低整体的调度性能。

拥塞是影响最终调度性能的关键因素之一，因此若能在图切分过程中提前感知到拥塞信息，就可以实施对应技术减轻拥塞，从而优化最终的调度性能。我们将这一任务定义为拥塞预测，即不在进行实际调度的情况下，预测到可能发生拥塞的区域（如mux节点）。

拥塞预测本身是一个非常困难的任务，主要的难点在于在我们的目标场景中拥塞是动态的，拥有时间属性。具体而言，拥塞情况会随时间步随时变化，一个Mux在某个时间步发生拥塞，并不意味着该Mux在其他时间步也会发生拥塞。因此，要进行拥塞预测，需要先对节点的调度顺序有一个预估，从而对每个Mux在每个时步下可能的拥塞情况进行建模。对两种方式可以对节点的调度顺序进行预估，第一种是3.1中提到的时步预测方法，第二种是预调度方法，即不考虑带宽等硬件约束下进行一次快速调度。在获得每个节点的调度时步之后，我们可以设计方法预测给定时步下的拥塞情况。

拥塞预测的另外一个难点体现在ground-truth获取。要标注拥塞区域，我们需要知道在实际调度过程中每个时步下的每个mux的带宽使用情况，但这一点在熟悉代码接口（如Srcmux2MuxValid）的情况下是可以完成的。

拥塞预测还有一个难点在于如何结合网表信息和硬件架构信息。目前我们的初步思路是构造一个融合异构图，其包含两类节点：网表节点和硬件资源节点（如mux）。对于一个给定的硬件资源，比如一个mux，其在每个时步下都有一个对应的节点，该节点会连接到对应的网表节点。硬件资源节点之间也会按照硬件架构进行连接，再加上网表中本来的连接关系，就构成了一个新的融合异构图。之后，我们可以利用图神经网络，为（每个时间步下）每个硬件资源节点生成低维表征（embedding），然后基于embedding进行二分类，判定是否出现拥塞。

在获取到拥塞信息之后，有两种反馈路线，和3.1中比较类似。第一种路线是设计后处理过程，将拥塞区域的节点交换到其他p64，以期减小拥塞，进而优化调度;第二种路线是将拥塞信息转换为节点、边权重，然后重新进行切分（如p512->p64），以期获得更好的切分结果。

### 四 可行性资源保障

为了保证项目的可行性，需要一些文档资料或者平台权限，罗列如下：

|  |  |  |
| --- | --- | --- |
| 资料/权限 | 是否已获取 | 用途 |
| 切分算法设计文档 | 是 | 帮助获得对切分/调度的整体流程的系统性认识，进而帮助更好地完成算法设计 |
| Escompile代码 | 是 | 帮助熟悉切分及调度算法的细节，帮助完成数据ground-truth标注。 |
| Benchmark平台 | 否 | 用于测试所设计的算法，验证其有效性 |
| 红区 | 否 | 提供真实的大规模网表 |
| GPU平台 | 否 | 提供运行图神经网络所必须的平台资源 |

### 五 参考文献

1. Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks, IEEE transactions on neural networks and learning systems,2020. [↑](#endnote-ref-1)
2. Hamilton W, Ying Z, Leskovec J. Inductive representation learning on large graphs. Advances in neural information processing systems, 2017. [↑](#endnote-ref-2)
3. Lopera D S, Servadei L, Kiprit G N, et al. A survey of graph neural networks for electronic design automation. ACM/IEEE 3rd Workshop on Machine Learning for CAD, 2021. [↑](#endnote-ref-3)
4. Ustun E, Deng C, Pal D, et al. Accurate operation delay prediction for FPGA HLS using graph neural networks. Proceedings of the 39th International Conference on Computer-Aided Design. 2020. [↑](#endnote-ref-4)
5. Mirhoseini A, Goldie A, Yazgan M, et al. A graph placement methodology for fast chip design, Nature, 2021. [↑](#endnote-ref-5)
6. Guo Z, Liu M, Gu J, et al. A Timing Engine Inspired Graph Neural Network Model for Pre-Routing Slack Prediction, 59th ACM/IEEE Design Automation Conference, 2022. [↑](#endnote-ref-6)
7. Ma Y, Ren H, Khailany B, et al. High performance graph convolutional networks with applications in testability analysis. Proceedings of the 56th Annual Design Automation Conference 2019. [↑](#endnote-ref-7)
8. Azade Nazi, Will Hang, Anna Goldie, Sujith Ravi, and Azalia Mirhoseini. A Deep Learning Framework For Graph Partitioning. In Seventh International Conference on Learning Representations, 2019. [↑](#endnote-ref-8)
9. Gatti A, Hu Z, Smidt T, et al. Deep Learning and Spectral Embedding for Graph Partitioning. Proceedings of the 2022 SIAM Conference on Parallel Processing for Scientific Computing. Society for Industrial and Applied Mathematics, 2022. [↑](#endnote-ref-9)