Графовые нейронные сети и их применение при проектировании цифровых СБИС

Н.В. Желудков[1], К.А. Петров[2]

[1]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, nvgel@cs.niisi.ras.ru;

[2]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, petrovk@cs.niisi.ras.ru

Аннотация. В статье рассматривается метод машинного обучения на графах, представлены архитектуры современных графовых нейронных сетей, а также их применение для решения задач проектирования цифровых СБИС, в особенности при размещении стандартных ячеек на этапе топологического проектирования. Решение данной задачи с помощью методов машинного обучения является актуальной проблемой, так как стандартные алгоритмы размещения, используемые в современных САПР, сталкиваются со сложностями при работе с цифровыми схемами, число логических элементов в которых достигает 10^6 и более. Это приводит к длительному времени работы и неоптимальности полученных результатов по параметрам занимаемой площади и энергопотребления проектируемой СБИС.

Ключевые слова: машинное обучение, графовые нейронные сети, GNN, топологическое проектирование СБИС.

1. Введение

В настоящее время часто можно увидеть представление различных данных или информации как в быту, так и в научной сфере в виде графов – математической абстракции, представляющей собой совокупность двух множеств – множества вершин (объектов) и множества ребер (связи и соединения этих объектов). К данным, которые часто представляются в виде графов, относятся представление связи атомов в молекулах различных веществ, связи пользователей внутри социальной сети, множество научных работ с цитированием друг друга и т.д. В проектировании цифровых СБИС в виде графа можно представить нетлист схемы – список логических вентилей и их межсоединений. В таком случае первые будут представлены как узлы графа, а вторые – как его ребра, то есть соединены ли два логических вентиля друг с другом или нет.

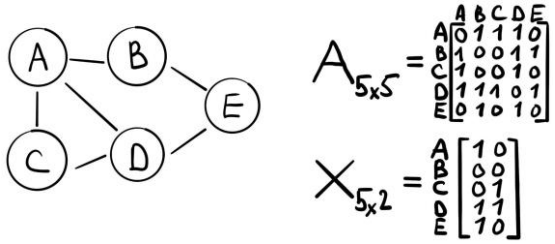

Рис. 1. Матрицы смежности A и признаков X

Очень часто сведения о графе представляют в матричной форме: в виде матрицы смежности A, где a_{ij} элемент равен числу ребер между i и j элементом графа, а также матрицей признаков X, где в i-й строке представлен вектор признаков i-го элемента графа (см. рис. 1).

Для данных, представленных в виде графа, также как для других видов представления информации (видео, изображения, таблицы, текст), можно применить методы машинного обучения для различного типа задач. К таким типам задач можно отнести:

- предсказание на уровне самого графа, то есть задача классификации графа, например, является ли данная молекула токсичной, на основе увиденных ранее экземпляров;
- предсказание на уровне ребер графа - предсказание возможных связей в графе, например, подбор рекламы под пользователя или рекомендации в социальных сетях;
- предсказание на уровне отдельных узлов, например, классификация отдельных узлов, на основе уже обработанных ранее;
- задача кластеризации, то есть задача объединения отдельных узлов графа в группы по набору каких-нибудь признаков.

Простейшая идея обучения на графах заключается в том, чтобы подать на вход полносвязной нейронной сети конкатенацию матрицы смежности A и матрицу признаков X и таким образом обучить модель [1]. Однако такой подход влечет за собой следующие фундаментальные проблемы:

- зависимость от размеров графа и числа его признаков, предложенную выше концепцию нейронной сети можно будет использовать только для графов с одним и тем же числом узлов, к тому же сложность модели будет зависеть от числа узлов, что при большом их числе сильно усложняет модель;
- изменение порядка нумерации узлов влияет на результат обучения; так как для графа нельзя

однозначно определить порядок узлов, то любое изменение в их нумерации влечет за собой изменение элементов матрицы смежности A, что не дает корректно провести обучение;

- граф обладает неевклидовой структурой [2], что, например, сказывается на отсутствие инвариантности к перестановкам узлов, и не позволяет использовать для него стандартные варианты сверточных нейронных сетей CNN.

Таким образом, для решения задачи требуется найти такое представление графа, в котором отражалась бы информация о его структуре и свойствах вершин. При этом такое представление должно вписываться в существующие методы машинного обучения. Такое представление получило название “представление узлов” (Node embedding) – узлам графа ставится в соответствие вектор размерности d (обычно, его размерность меньше числа признаков для узла) в евклидовом пространстве (см рис. 2). Эти представления должны содержать информацию о признаках самого узла и, некоторым образом, содержать в себе информацию о признаках его соседях и об общей структуре графа в окрестности рассматриваемого узла.

Такой аналог свертки для графов позволит применить к представлениям узлов стандартные методы машинного обучения. Естественным образом встает вопрос, каким образом получить данные представления узлов? В последние годы для этой задачи используются новая архитектура нейронных сетей – графовые нейронные сети.

Рис. 2. Представление графа в виде представления его узлов

2. Графовые нейронные сети

Новый подход к получению представлений узлов был предложен в работе [3]. Он заключается в том, что каждой вершине графа ставится в соответствие “скрытое представление” (hidden embedding), которое будет обновляться в зависимости от числа слоев. Нулевое скрытое представление вершины – это ее вектор признаков из матрицы X. Скрытое представление этого узла на первом слое будет представлять собой агрегацию данных от соседей этого узла, которые находятся на расстоянии в 1 вершину, самого скрытого представления рассматриваемого узла на предыдущем слое, а также функции “обновления”, которая вычисляет на основе описанных выше величин обновленное скрытое представление узла в этом слое. Такой подход к агрегации данных от соседей узла и обновлении скрытых представлений получил название нейронная пересылка сообщений (Neural message passing) [4] или сокращенно пересылкой сообщений. Под сообщениями, которыми обмениваются вершины графа, понимают скрытые представления этих вершин на определенном уровне (слое), обработанные определенным образом (выбранным методом или функцией). Проход с номером k (который также можно назвать уровнем или слоем) пересылки сообщений обновляет представление рассматриваемого узла в радиусе k-вершин в его локальном соседстве. В общем виде этот процесс описывается формулой (1):

$$\mathbf{h}_u^{(k+1)} = \mathrm{UPDATE}^{(k)}\left(\mathbf{h}_u^{(k)}, \mathrm{AGGREGATE}^{(k)}(\{\mathbf{h}_v^{(k)}, \forall v \in \mathcal{N}(u)\})\right) \qquad (1)$$

где $h_u^{(k+1)}$ – представление узла u в слое k+1, $\mathrm{UPDATE}^{(k)}$ – функция, обновляющая представление узла в слое k, $\mathrm{AGGREGATE}^{(k)}$ – функция агрегации данных от соседних узлов, $h_v^{(k)}$ – представление узлов v, которые являются соседями узла u (N(u)) на расстоянии в k узлов.

В зависимости от вида сообщений, способа их агрегации и функции обновления скрытого представления различают несколько видов нейронных сетей. Основной и классической архитектурой сети такого вида является графовая нейронная сеть (GNN, Graph Neural Network). В качестве сообщений здесь используются скрытые представления от соседних узлов в k-радиусе, деленные на степень соответствующего узла (чтобы уменьшить разброс между представлениями вершин, число соседей которых значительно отличается), идущие в финальную формулу с весовыми коэффициентами (которые и являются тренируемыми параметрами). Функция обновления скрытого состояния представляет собой одну из нелинейных функций, используемых в качестве функции активации нейронов в сети – это может быть сигмоидальная функция, функция ReLu и т.д. Формулы (2), (3) и (4) используются для вычисления представлений узлов в GNN:

$$\mathbf{h}_v^{(0)} = \mathbf{x}_v \qquad (2)$$

$$\mathbf{h}_v^{(k+1)} = \sigma\left(\mathbf{W}_k \sum_{u \in \mathrm{N}(v)} \frac{\mathbf{h}_u^{(k)}}{|\mathrm{N}(v)|} + \mathbf{B}_k \mathbf{h}_v^{(k)}\right), \forall k \in \{0..K-1\} \qquad (3)$$

$$\mathbf{z}_v = \mathbf{h}_v^{(K)} \qquad (4)$$

где $h_u^{(0)}$ – начальное представление узла u, равное x_u – u-строчке матрицы признаков, $h_u^{(k+1)}$ – представление узла u в слое k+1, W_k и B_k - веса нейронов в слое k, $|N(v)|$ – степень узла v, z_u – представление узла u в финальном слое K.

Число слоев K является гиперпараметром обучаемой сети и подбирается в зависимости от параметров графа и их особенностей.

После получения финального набора скрытых представлений на k-слое (после прямого распространения), встает задача обучения нашей моделей (параметров W_k и B_k из формулы (3)), чтобы уточнить полученные представления в контексте решаемой задачи. Вид обучения будет зависеть от имеющихся у нас данных (например, формат и тип матрицы признаков X) или типа задачи предсказания. Виды обучения графовой нейронной сети можно разделить на два типа: обучение с учителем и без него.

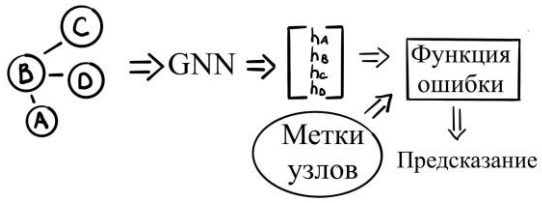

Рис. 3. Обучение с учителем

Обучение с учителем (supervised learning), когда известны, например, виды вершин (каждая вершина имеет метку своего класса, как на рис. 3), на которые надо классифицировать новые. В таких случаях функция потерь считает различие между фактическим значением типа вершины и тем, которое было получено исходя из ее скрытого представления. Такой тип обучения хорошо подходит для задачи предсказания на уровне узлов графа и его ребер.

Обучение без учителя (unsupervised learning) требуется применять в таких задачах, в которых нет меток вершин и ребер. В этом случае обучение можно строить на основе структуры самого графа. Интуитивным является визуализация графа таким образом, чтобы узлы, лежащие близко к друг другу, имели в евклидовом пространстве схожие скрытые представления (см. рис. 4).В данных задачах функция потерь будет оценивать разницу между “близостью” двух вершин на графе и “близостью” представлений этих вершин в пространстве скрытых представлений. В качестве меры близости узлов на графе можно использовать элементы матрицы смежности A или более комплексные оценки, базирующиеся на алгоритмах “случайных блужданий” (DeepWalk и node2vec). В качестве меры близости представлений вершин можно использовать также векторное расстояние между ними или скалярное произведение.

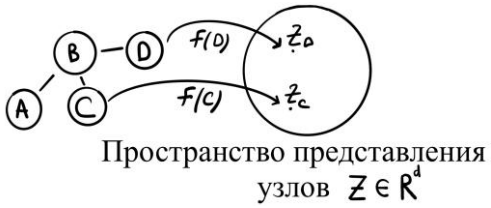

Рис. 4. Представление вершин графа

Отдельно уточним, что на выходе k-слоя графовой нейронной сети мы получаем представления вершин исследуемого графа, прошедших через обученную сеть. Данные представления можно передать на вход другой полносвязной нейронной сети или использовать в качестве входных данных для алгоритма.

Такая архитектура графовой нейронной сети и процесс ее обучения позволяет добиться следующих преимуществ перед примитивным подходом, который рассматривался выше (с подачей на вход сети матрицы смежности и матрицы признаков):

- инвариантность к нумерации узлов;

- позволяет подавать на вход графы любого размера, не требуется заново переучивать сеть.

3. GNN в проектировании цифровых СБИС

Как отмечалось ранее, многие данные при проектировании цифровых СБИС (в том числе нетлист схемы) можно представить в виде графа. Ввиду увеличения функциональной сложности схем и роста числа элементов в них, старые алгоритмы решения многих задач в области проектирования цифровых СБИС показывают неудовлетворительные результаты, а также затрачивают на свою работу много времени и ресурсов вычислительной машины. В последние годы значительно возросло число научных работ, посвященных применению различных методов машинного обучения при проектировании цифровых схем [5]. Особую роль в решении этих задач заняли графовые нейрон-

ные сети. Среди задач проектирования, в которых изучается применение графовых нейронных сетей можно выделить:

- оценка потребляемой мощности на основе GNN (на рис. 5 представлен вариант представления нетлиста в граф, а также нужных параметров узлов и ребер для расчета мощности), представленная в работе [6], где авторам удалось добиться с помощью графовой нейронной сети ускорения работы в 18,7 раза по сравнению со стандартным алгоритмом и ошибкой меньше 5,5%;

- анализ просадки напряжения на сетке земли-питания в работе [7]; оценка просадки питания в цифровом блоке из статьи стандартными средствами одного из коммерческих САПР занимает 3 часа, в то время как на основе GNN – 18 минут с точностью 94%;

- оценка паразитных характеристик при моделировании топологии схемы [8];

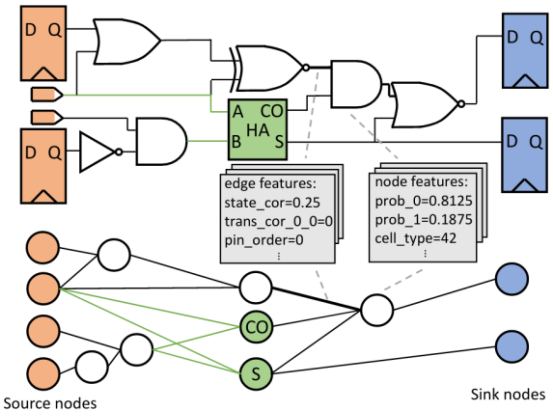

Рис. 5. Представление нетлиста и его параметров в виде графа для оценки мощности [6]

- модель на основе GNN, предсказывающая места на схеме, в которых могут быть проблемы с разводимостью, на основе плана размещения и нетлиста схемы [9]. На рис. 6 на изображении слева представлена реальная оценка разводимости, а справа - предсказанная.

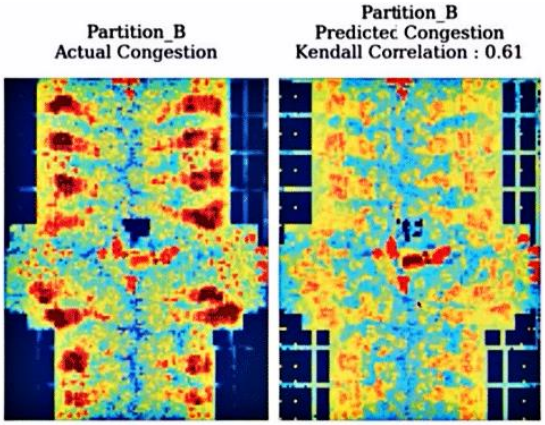

Рис. 6. Оценка разводимости с GNN [9]

К этому списку можно также отнести важную задачу размещения стандартных ячеек. Это задача является одним из наиболее вычислительно сложных этапов топологического проектирования СБИС. После создания плана размещения, включающего в себя определение размеров проектируемого блока, расстановки входных-выходных портов, размещения макроблоков (памятей, аналоговых блоков и т.д.) и построения сетки земли-питания, необходимо разместить стандартные ячейки, представленные в нетлисте. То, как будут размещены ячейки по площади блока, прямым образом отразится на будущее быстродействие схемы, трудности с разводимостью межсоединений на этапе трассировки, а также может затронуть потребляемую мощность и локальные просадки на сетке земли-питания. Этот этап проектирования является критически важным, поскольку от его результата зависят многие важные характеристики проектируемой СБИС.

Большинство алгоритмов, используемых для решения задачи размещения, в своей основе содержат механизм минимизации длины межсоединений ячеек. Годы использования таких алгоритмов показали высокую корреляцию между этой характеристикой и результатами этапа размещения. Например, в САПР с открытым исходным кодом OpenROAD для размещения ячеек используется алгоритм решения аналога электростатических уравнений для размещения методом Нестерова. Недостатком многих подобных методов, как уже упоминалось ранее, является большая длительность работы и качество результатов.

Решение данной проблемы с использованием графовых нейронных сетей предлагается в

работе [10]. Алгоритм, представленный в статье, состоит из двух частей. В первой из оригинального нетлиста схемы необходимо получить граф, узлы которого будут иметь определенные (рассмотренные в дальнейшем) признаки; затем на основе графовой нейронной сети надо получить представление вершин, проведя обучение без учителя этой сети таким образом, чтобы полученные представления вершин имели схожесть с логической близостью элементов в нетлисте. Вторая часть алгоритма заключается в применении к полученным представлениям вершин метода K-средних (алгоритма кластеризации), для объединения “близких” в пространстве представлений вершин в кластеры, которые станут основой для групп-размещения в коммерческом САПР (в работе авторы используют САПР IC Compiler 2 компании Synopsys). Схематически эти два этапа представлены на рис. 7.

Рассмотрим подробнее, как авторы выбирают в своей работе признаки для вершин графа, а также как проводят обучение. Вектор начальных признаков для вершины графа, т.е. изначально для ячейки в нетлисте, формируется на основе иерархической информации – ячейки, находящиеся в одном иерархическом модуле, должны иметь одинаковый вектор признаков, а чем дальше в иерархии друг от друга находятся ячейки, тем сильнее должны отличаться их начальные представления.

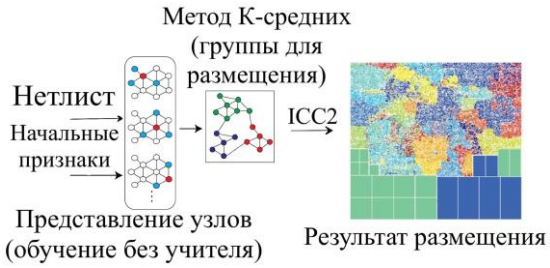

Рис. 7. Схема работы алгоритма [10]

Для кодирования иерархической информации о ячейках используется иерархическое дерево, представленное на рис. 8.

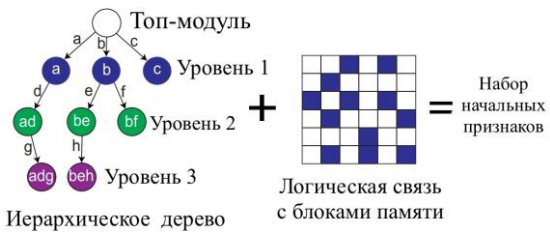

Рис. 8. Задание начальных признаков [10]

Также к общим иерархическим признакам добавляется информация о том, насколько далеко рассматриваемая ячейка находится от блоков памяти (в общем случае – от всех макроблоков в схеме) в нетлисте. Авторы работы аргументируют такой выбор признаков тем, что ячейки, находящиеся в близких иерархических модулях, имеют больше межсоединений между собой, и, если размещать их рядом – это должно позитивно отразиться на качестве этапа размещения. Добавление же к вектору признаков информации о близости к ячейкам памяти обусловлено тем, что пути между триггерами и портами памятей часто являются критическими в плане быстродействия. Добавление данных об этом может помочь провести размещение ячеек с учетом этой особенности.

Для нахождения представлений вершин используется одна из разновидностей архитектур графовых нейронных сетей Graph-SAGE. В отличие от базового варианта GNN, представленного на рис. 4, в котором агрегация сообщений от соседних вершин суммируется с представлением рассматриваемой вершины с некоторыми коэффициентами, в Graph-SAGE агрегация сообщений и представление узла конкатенируются. В своей работе авторы используют 2 слоя такой сети с 128 нейронами в каждом слое.

Для испытания своего метода размещения ячеек авторы использовали два цифровых блока СБИС по технологии TSMC 28nm с числом ячеек около 200 тыс. Новый подход (по сравнению с алгоритмом используемым в САПР ICC2) с использованием графовой нейронной сети позволил сократить суммарную длину проводов на 3,9%, потребляемую мощность на 2,8% и увеличить запас по времени установки (показатель, определяющий общее быстродействие схемы) на 85,7 %.

4. Заключение

В статье рассмотрены особенности представления информации в виде графов (в том числе данных, используемых при проектировании цифровых СБИС) в контексте решения задач машинного обучения, рассмотрены современные архитектуры графовых нейронных сетей, процесс их обучения и достоинства перед другими методами обучения на графах. Проведен обзор современных тенденций в использовании графовых нейронных сетей для решения

различных задач при проектировании цифровых СБИС. Особое внимание уделено работе, затрагивающей использование графовых нейронных сетей на этапе размещения стандартных ячеек во время топологического проектирования СБИС. Использование данного подхода позволило получить лучшие результаты по быстродействию и потребляемой мощности, по сравнению с результатами с использованием стандартного алгоритма в САПР Synopsys ICC2. Основываясь на обзоре описанных выше работ наиболее перспективным направлением исследований в этой области являются работы по решению задачи размещения ячеек:

- исследовать новые способы задания начальных признаков вершин, для более точного соответствия с нетлистом;

- применить разные виды обучения моделей, рассмотреть варианты использования GNN с большим числом слоев для более глубокого анализа;

- найти применение полученным с помощью GNN представлениям узлов для более точной расстановки ячеек, чем при кластеризации.

Исходя из анализа работ на данную тему можно сделать вывод, что данное направление, связанное с использованием графовых нейронных сетей и смежных методов машинного обучения в проектировании СБИС, актуально, его изучение и активное использование позволяет сократить время на разработку интегральных схем, а также получить лучшие характеристики проектируемых СБИС по сравнению со стандартными алгоритмами.

Публикация выполнена в рамках государственного задания ФГУ ФНЦ НИИСИ РАН по теме FNEF-2022-0008.

Graph neural networks and their application in the design of digital VLSI

N.V. Zheludkov, K.A. Petrov

Abstract. The article discusses the method of machine learning on graphs, presents the architecture of modern graph neural networks, as well as their application for solving digital VLSI design problems, especially in placing standard cells at the layout design stage. The solution of this problem using machine learning methods is an urgent problem, since the standard placement algorithms used in modern CAD systems face difficulties when working with digital circuits, the number of logic elements in which reaches 10^6 and more. This leads to a long operating time and non-optimality of the results obtained in terms of the occupied area and power consumption of the designed VLSI.

Keywords: machine learning, graph neural networks (GNN), layout design of VLSI.

Литература

1. W.L. Hamilton, R. Ying, J. Leskove. Representation Learning on Graphs: Methods and Applications. «Bulletin of the IEEE Computer Society Technical Committee on Data Engineering», Vol. 40 (2017), №3, 52-74.
2. M.M. Bronstein, J. Bruna, Y, LeCun, A.Szlam. Geometric Deep Learning: Going beyond Euclidean data. “IEEE Signal Processing Magazine”, Vol. 34 (2017), №6, 18-42.
3. T.N. Kipf, M. Welling. Semi-supervised Classification with Graph Convolutional Networks. “5th International Conference on Learning Representations 2017”, France, Toulon, ICLR, 2017, 66-80.
4. W.L. Hamilton. Graph Representation Learning. Synthesis Lectures on Artificial Intelligence and Machine Learning, Vol. 14 (2019), №3, 1-159.
5. G. Huang, J. Hu, Y. He. Machine Learning for Electronic Design Automation: A Survey. “ACM Transactions on Design Automation of Electronic Systems”, Vol. 26 (2021), №5, 1-46.
6. Y. Zhang, M. Ren, B. Khailany. GRANNITE: Graph neural Network Interference for Transferable Power Estimation. “Design Automation Conference (DAC) 2020”, USA, San Francisco, IEEE, 2020.
7. V. Zhang, M. Ren, B. Keller, B. Khailany. MAVIREC: ML-Aided Vectored IR-Drop Estimation and Classification. “Design, Automation & Test in Europe Conference & Exhibition, 2021”, France, Grenoble, 2021, IEEE, 2021.

8. M. Ren, G. Kokai, W. Turner, T. Ku. ParaGraph: Layout Parasitics and Device Parameter Prediction using Graph Neural Networks, “Design Automation Conference (DAC) 2020”. USA, San Francisco, IEEE, 2020.
9. R. Kirby, S. Godil, R. Roy, B. Catanzaro. CongestionNet: Routing Congestion Prediction Using Deep Graph Neural Networks. “IFIP/IEEE 27th International Conference on Very Large Scale Integration (VLSI-SoC) 2019”, Peru, Cuzco, IEEE, 2019.
10. Y. Lu, S. Pentapati, S.K. Lim. VLSI Placement Optimization using Graph Neural Networks. “IEEE/ACM International Conference On Computer Aided Design (ICCAD)”, USA, San Diego, 2020.