ECO-GNN: Signoff Power Prediction Using Graph Neural Networks with Subgraph Approximation

YI-CHEN LU, Georgia Institute of Technology, USA
SIDDHARTHA NATH, Intel, USA
SAI PENTAPATI and SUNG KYU LIM, Georgia Institute of Technology, USA

Modern electronic design automation flows depend on both implementation and signoff tools to perform timing-constrained power optimization through Engineering Change Orders (ECOs), which involve gate sizing and threshold-voltage (V_{th})-assignment of standard cells. However, the signoff ECO optimization is highly time-consuming, and the power improvement is hard to predict in advance. Ever since the industrial benchmarks released by the ISPD-2012 gate-sizing contest, active research has been conducted extensively to improve the optimization process. Nonetheless, previous works were mostly based on heuristics or analytical methods whose timing models were oversimplified and lacked of formal validations from commercial signoff tools. In this article, we propose ECO-graph neural networks (GNN), a transferable graph-learning-based framework, which harnesses GNNs to perform commercial-quality signoff power optimization through discrete V_{th}-assignment. One of the highlights of our framework is that it generates tool-accurate optimization results *instantly* on unseen netlists that are not utilized in the training process. Furthermore, we propose a subgraph approximation technique to improve training and inferencing time of the proposed GNN model. We show that design instances with non-overlapping subgraphs can be optimized in parallel so as to improve the inference time of the learning-based model. Finally, we implement a GNN-based explanation method to interpret the optimization results achieved by our framework. Experimental results on 14 industrial designs, including a RISC-V-based multi-core system and the renowned ISPD-2012 benchmarks, demonstrate that our framework achieves up to 14× runtime improvement with similar signoff power optimization quality compared with *Synopsys PrimeTime*, an industry-leading signoff tool.

CCS Concepts: • **Hardware** → **Methodologies for EDA**; **Physical design (EDA);**

Additional Key Words and Phrases: Engineering Change Order (ECO), power prediction, Graph Neural Networks (GNNs)

ACM Reference format:
Yi-Chen Lu, Siddhartha Nath, Sai Pentapati, and Sung Kyu Lim. 2023. ECO-GNN: Signoff Power Prediction Using Graph Neural Networks with Subgraph Approximation. *ACM Trans. Des. Autom. Electron. Syst.* 28, 4, Article 55 (May 2023), 22 pages.
https://doi.org/10.1145/3569942

Authors' addresses: Y.-C. Lu, S. Pentapati, and S. K. Lim, Georgia Institute of Technology, Georgia, USA; emails: yclu@gatech.edu, sai.pentapati@gatech.edu, limsk@ece.gatech.edu; S. Nath, Intel, California, USA; email: siddhartha.nath@intel.com.

1084-4309/2023/05-ART55 $15.00
https://doi.org/10.1145/3569942

1 INTRODUCTION

Handheld and wearable devices are significantly proliferating in today's semiconductor markets and demand extremely low power dissipation while operating at voltages as low as 0.45 V. However, in advanced technology nodes, power optimization has become much more complicated than optimizations on other design metrics such as wirelength and delay, which is due to the dominance of leakage power and its complicated relation with the dynamic power [21]. It is well known that leakage power increases exponentially with the scaling of threshold voltage (V_{th}). Therefore, low-voltage design in advanced technology nodes such as 7 nm and 5 nm and below require design implementation tools to aggressively optimize leakage power at various stages of commercial **physical design (PD)** flows, which are often achieved by **Engineering Change Orders (ECOs)** that involve gate-sizing and V_{th}-assignment. In this article, we specifically focus on improving the power ECO at the signoff stage, which is more time-consuming than the power optimizations at other PD stages, because it requires a precise calculation of the delay budget.

In modern chip design flows, ECOs are performed extensively from synthesis to signoff with an aim to optimize **power, performance and area (PPA)** metrics. Every top semiconductor design company runs multiple iterations of signoff ECO to achieve the target PPA. However, in advanced technology nodes, power optimization has become much more complicated than optimizations on other design metrics such as wirelength and timing, which is mainly due to the dominance of leakage power and its complicated relation with the dynamic power [21]. Even though design implementation tools have developed various power optimization techniques throughout the years, designers still heavily rely on signoff tools to recover power at the signoff stage using ECO change-lists. Nonetheless, since power ECO in signoff tools requires accurate timing budget calculation (e.g., path-based timing analysis) during the optimization, it is extremely time-consuming and thus bottlenecks the chip design process. Therefore, in this work, we aim to develop a learning-based framework, ECO-**graph neural network (GNN)**, that has the ability to perform signoff power prediction to improve the chip design turnaround time.

Gate sizing and V_{th}-assignment are the two popular techniques to optimize design power consumption. However, since gate-sizing requires further legalization and routing to validate the design after the optimization, V_{th}-assignment is the preferred approach during signoff ECO, as it causes minimum disturbance to the overall placed and routed layout. In *Synopsys PrimeTime*, V_{th}-assignments during signoff ECO not only optimize the leakage power but also reduce the dynamic power simultaneously [28]. Nonetheless, this optimization conducted by *PrimeTime* is time-consuming, and the tool itself remains a blackbox for designers. Therefore, in this work, our goal is to develop a fast, explainable signoff power optimization framework that has the ability to perform commercial quality signoff power optimization instantly as well as the facility to explain the achieved optimization results.

V_{th}-assignment refers to assigning an appropriate V_{th} type for each design instance from a set of standard cell libraries to perform power optimization without violating timing constraints [17]. Note that for a given design instance, all the available V_{th} types have the same footprint, and the total number of the available types is limited to the discrete values of threshold voltages specified by the technology. This optimization problem is proven to be NP-hard [18], which means the optimality of a sizing solution is hard to be demonstrated and implies great opportunities to employ machine learning techniques for solving this problem.

Modern commercial signoff tools perform signoff power ECO based on sophisticated in-house timing models. The models precisely calculate the timing budget for every design instance to help the signoff engines conduct timing-constrained power optimization. The optimization results achieved by these tools are considered as golden QoR in the industry; however, there are two significant drawbacks in the current industrial signoff flows, namely

- **Extremely long runtime.** A signoff power ECO run often takes several days on an industrial scale design and requires human-in-the-loop for enhancement, which drastically bottlenecks the chip development process.
- **Obscure improvement.** The power improvement is unknown in advance. Designers tend to run multiple optimization configurations in parallel to select the best one in the end, which consumes significant amount of computing resources.
- **Partial netlist update.** In many real-world scenarios, designers would only want to perform the signoff power optimization on a few selected instances with positive slack margin to prevent severe timing degradation. However, the improvements that can be generated from these instances are often unclear until after spending significant amount of time in ECO iterations.

In this work, we overcome the above issues by presenting ECO-GNN, which is a graph-learning-based framework that leverages GNNs to perform V_{th}-assignments for fast signoff power optimization [15]. Specifically, we present two approaches to overcome the issues. First, we present a classification-based technique to predict the final V_{th}-assignment of each design instance that will be made by *PrimeTime* during the ECO using the information of the entire netlist. Second, we further propose a "subgraph approximation" technique to demonstrate the ability of our framework on predicting the actual power savings of targeted design instances. In summary, after performing supervised learning on several designs with the assignment ground-truths given by *Synopsys PrimeTime*, our framework has the ability to perform tool-accurate signoff power optimization on *unseen* designs instantly without degrading the performance or introducing new **design rule violations (DRVs)**. To validate our framework, we consider *Synopsys PrimeTime* as our baseline, and demonstrate that ECO-GNN achieves comparable optimization results with up to 14× runtime improvement on the ISPD-2012 benchmarks [19] and other real-world designs, including a RISC-V based multi-core system.

The goal of this work is to provide designers a fast and accurate signoff power optimization framework with high fidelity as the industry-standard commercial tool, *Synopsys PrimeTime*. The key contributions of this article are summarized as follows:

(1) Our first major finding is that ECO-GNN learns the behaviour of *Synopsys PrimeTime* effectively and generates comparable optimization results at inference time.
(2) Our second major finding is that ECO-GNN generally shows better power saving but worse timing saving compared with *Synopsys PrimeTime*. This indicates that ECO-GNN algorithms are more effective in power optimization.
(3) Unlike commercial tools or previous works (see Section 2) that require multiple iterations to assign appropriate V_{th} types, our framework ECO-GNN only needs one-pass to determine the final V_{th} type for every design instance.
(4) Rather than treating our learning-driven framework as a blackbox, we implement a GNN-based explanation method [32] to quantitatively interpret the V_{th}-assignment predictions made by our framework. Given a target node, the method identifies the influential local sub-graph that has high contribution to its V_{th}-assignment. We believe this interpretability would help designers understand the complicated characteristics of discrete sizing during signoff ECO.
(5) After demonstrating the effectiveness of using GNNs to model using whole netlist information, we propose a subgraph approximation technique to speed up the training and inferencing time of the proposed GNN model using local graph structures of targeted instances without sacrificing the accuracy.

(6) In this article, we explore both classification and regression approaches to predict ECO power optimization results, with an ultimate goal of helping designers reduce the chip design turnaround time. We demonstrate that the proposed framework can not only predict the end V_{th}-assignment of each design instance with high F1-score but also deliver accurate estimations of the power saving from the optimization.
(7) We demonstrate that the proposed subgraph approximation technique can not only be utilized to solve the regression problem of predicting actual power saving but also be leveraged to improve the classification accuracy of the prediction of V_{th}-assignment.
(8) To the best of our knowledge, this is the first work that formulates signoff power optimization problem into a graph learning problem and validates the proposed framework using an industrial-leading commercial tool under an advanced technology node.

2 RELATED WORKS AND MOTIVATIONS

The literature in V_{th}-assignment for power optimization has been researched extensively throughout the past decade. Early works mainly focus on using analytical and heuristic (i.e., non-analytical) methods to improve the optimization; however, these methods demonstrate poor generalization results across different technologies and designs. Recently, **machine learning– (ML)** based approaches emerge as promising alternatives to tackle the problem, which often demonstrate better optimization results in much lesser runtime. In the following list, we summarize previous works into these three categories:

- Non-Analytical Methods: First, we introduce the heuristic-based algorithms. This category contains methods that leverage greedy-based [8, 17, 22], simulated annealing [7, 23], or dynamic programming [12, 14] algorithms to find feasible solutions. However, there are a few major drawbacks in these algorithms. First, these algorithms often demonstrate poor convergence results, which is because they often assume the design global optimum of power optimization can be achieved by iterative finding the local optimum of cell-based power saving. Second, these approaches are highly sensitive to heuristics and are often design or technology specific. Therefore, they are hard to be extended to never-seen designs or various technologies.
- Analytical Methods: Another popular category is the analytical-based approach. Algorithms in this category often formulate the power optimization problem into a convex optimization [24, 27] or a Lagrangian optimization problem [12, 20, 25, 26] whose objective is to maximize the power reduction through discrete sizing under certain timing constraints. These methods are considered to yield better and more reliable optimization results than the non-analytical methods. However, solving an optimization problem using numerical approaches is extremely time-consuming. Given that a real-world design can easily introduce tens of thousands of variables, these approaches are limited for real-world usage.
- Machine Learning: It is widely acknowledged that ML has emerged as a promising approach to solve the V_{th}-assignment problem with huge benefits in runtime saving, which is critical for productivity. The authors of Reference [2] leverage linear regression to find feasible solutions based on path slack estimation. Another work [21] utilizes support vector machine with lazy timing analysis to further enhance the optimization quality. However, these studies neglect that the final gate-type of each design instance highly depends on the characteristics of its neighbors. Therefore, they are not sufficient to perform the power optimization accurately without spending significant amount of time in feature engineering.

  To leverage the netlist graph information in solving the power optimization problem, recently, the authors of References [11, 15, 16, 29] propose graph learning-based frameworks to predict the leakage power saving of each design instance based on its local neighborhood information. These approaches demonstrate significant accuracy improvement compared with the above traditional ML-based approach while achieving similar runtime saving. Hence, it is proven that the sizing result of a targeted cell highly depends on the features of its neighbors. However, these GNN-based literature neglect the fact that the receptive field of their GNN models are only subject to the number of layers of the graph convolutional layers, which is less than 3 across all previous works. That is, if a GNN model has k convolutional layers, then the power saving prediction of an instance will only depend on the features within its local k-hop neighborhood structure and nothing beyond. In other words, the final gate-sizing prediction of a design instance will only depend on its local subgraph. Unlike previous works that rely on full-graph approaches to predict sizing solutions, in this article, we present a subgraph approximation technique find the solutions in much faster runtime and higher accuracy. Details of the proposed methodology will be discussed in Section 6.

Finally, besides the specific shortcomings raised in each of the category above, there exist several common drawbacks in most of the previous works. First, the timing models they leverage are oversimplified, which does not reflect real-world scenarios. In this article, we think the validation from commercial signoff engine is critical to the application of the proposed models. Second, the original ISPD-2012 benchmarks [19] that most of them leverage for evaluations are problematic. We analyze the benchmarks using *Synopsys PrimeTime* and discover that the original worst negative slack values across all the designs range from −1 to −8 ns, where all the target frequencies are less than 1 GHz. This simple fact makes previous works unrealistic, because the power optimization is meaningful only if the optimized designs are in signoff quality. Finally, none of the previous works interpret the optimization results achieved by their methods, where they all consider their optimization engines/models as blackboxes.

3 DESIGNING OF EXPERIMENTS

Inspired from the limitations and drawbacks of the previous works, in this article, we consider *Synopsys PrimeTime*, a leading industrial signoff tool, as our baseline and propose ECO-GNN, a transferable graph-learning-based signoff power optimization framework that can be easily integrated with any modern PD flow. To provide fair and meaningful comparisons with prior works, we re-synthesize the ISPD-2012 benchmarks using TSMC 28 nm technology node and demonstrate that our framework ECO-GNN performs commercial-quality signoff power optimization instantly on these designs. Furthermore, we leverage a GNN-based explanation method [32] to interpret the V_{th}-assignments made by our framework to ensure that our framework is reliable.

4 OVERVIEW OF ECO-GNN FRAMEWORK

Recently, GNNs have revolutionized many research areas, spanning from biology, social science, chemistry, and many others [6]. They perform effective graph representation learning, where the goal is to construct meaningful node embeddings that accurately characterize the nodes in the graph. In general, GNNs follow a message passing scheme, where a feature vector of a node can be considered as a message being iteratively transformed and passed to its neighboring nodes. At the end of the graph learning process, the initial node features are transformed into better representations that can be utilized in downstream tasks such as link prediction, node classification, and clustering [31].

Figure 1 presents a high-level view of our framework ECO-GNN. Since VLSI netlists can be naturally represented as hypergraphs, in this article, we leverage a specific variant of GNNs named GraphSAGE [6] to conduct graph representation learning directly on the netlist graphs. After getting the learned representations, we utilize a softmax-based classification model to predict the V_{th}-assignments that optimize the signoff power. Note that the entire learning is an end-to-end

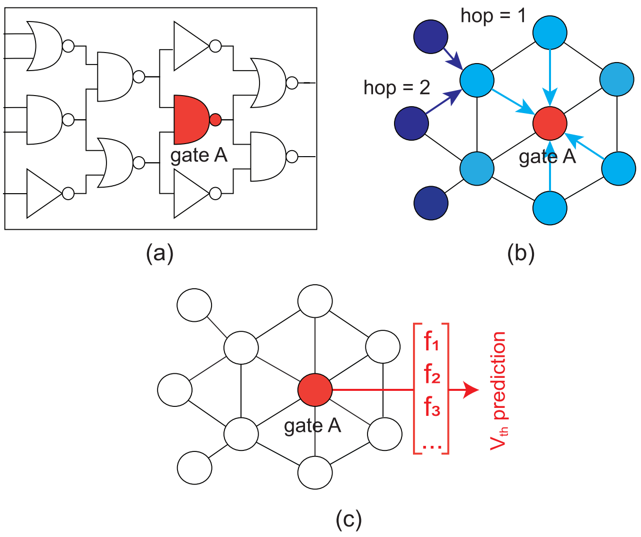

Fig. 1. High-level view of our ECO-GNN framework, (a) input netlist, (b) graph representation learning, and (c) V_{th} prediction. Note that (b) and (c) visualize the original netlist in a clique-based representation.

process. The classification loss that represents the cross-entropy between our predictions and the ground-truths from *Synopsys PrimeTime* is utilized to update the parameters inside GNN and the classification model through gradient descent.

The detailed learning process shown in Figure 1 works as follows. Given an input netlist as shown in Figure 1(a), to determine the V_{th}-assignment of the target cell (red-colored), we first leverage a GNN to sample and aggregate the features from its neighboring cells as shown in Figure 1(b). Then, we predict its V_{th}-assignment based on the aggregated representation vector as shown in Figure 1(c).

4.1 Our Objectives: Regression and Classification

The goal of this work is to construct a "general framework" that achieves *commercial-quality* signoff power optimization results *at inference time* (the testing time of the model). To achieve this goal, we explore two modeling approaches to solve different categories of problems: the regression and the classification problems. In this article, a regression-based model refers to the framework that predicts the "power difference" before and after ECO optimization, where a classification-based model refers to the framework that generates the sizing results (i.e., assigning a new V_{th} type for each design instance). These two types of problems are inherently different and subject to various real-world applications depending on the use cases. In Section 6, we present a "subgraph approximation" technique to solve the regression problem and leverage a node representation learning-based technique to solve the classification problem. Finally, note that our framework does not assume any pre-defined netlist structure, so it is generalizable to every design. After learning on a few designs, it has the facility to determine the V_{th}-assignments on the unseen ones that optimize the signoff power.

5 DESIGN OF EXPERIMENTS

In this work, we follow the experimental setting of the ISPD-2012 power optimization contest as many previous works, where all the cells in a given design are initially in the lowest V_{th} type (tightest timing constraint). As mentioned in Section 2, we re-synthesize the ISPD benchmarks

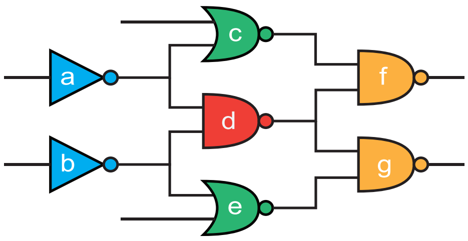

Fig. 2. Initial feature construction of cell d, where its fan-ins $\{a, b\}$, siblings $\{c, e\}$, and fan-outs $\{f, g\}$ are taken into consideration.

using a *TSMC 28 nm* technology node to ensure all the designs are in signoff performance before conducting the power optimization through V_{th}-assignments.

5.1 Problem Formulation

Given a netlist $G = (V, E)$, where V denotes the instances in the design and E represents the logical connections, assume that for each instance $v \in V$, there are n V_{th}-assignments available from the standard cell libraries. Let $x_v^j = 1$ if instance v is realized with jth V_{th} choice in the libraries and $x_v^j = 0$ otherwise. We formally define the signoff power optimization problem as follows:

$$\text{minimize} \sum_{i=1}^{|V|} \sum_{j=1}^{n} P(v_i^j) x_{v_i}^j, \tag{1}$$

where $P(v_i^j)$ represents the signoff power of instance v_i when jth choice of V_{th}-assignment is realized such that the **worst negative slack (WNS)** along with the **total negative slack (TNS)** do not degrade after the assignments, and no new DRVs are added.

5.2 Initial Node Features

Before leveraging GNN to conduct graph learning, we define an initial feature vector for each design instance as shown in Table 1. The term "initial" indicates that during the graph learning process, these original features are transformed to other representations that are more beneficial for the classification model to determine the appropriate V_{th}-assignments that optimize signoff power.

Features in Table 1 are extracted from technology files, SPEF files, and timing reports. These 20 features are chosen based on domain knowledge and parameter sweeping experiments. Most of them are related to timing, because during the signoff power optimization, an instance's V_{th}-assignment changes only if the *WNS* and *TNS* do not degrade and no DRV is introduced. Figure 2 further illustrates the feature construction process. To determine the initial features of a target instance d, we take the information of its fanins (instances $\{a, b\}$), siblings (instances $\{c, e\}$), and fanouts (instances $\{f, g\}$) into account. However, these manually engineered features are not sufficient to predict the V_{th}-assignments that optimize the design signoff power. To get better node representations, we leverage GNNs to perform the graph representation learning.

6 ECO-GNN ALGORITHM

6.1 Overview of the Algorithm

Figure 3 shows a detailed illustration of the learning process in the ECO-GNN framework. Given a netlist graph $G = (V, E)$, our framework first takes the initial node features defined in Table 1 as inputs. Then, it leverages GraphSAGE [6], a variant of GNNs to perform graph learning. The goal of graph learning is to obtain the node representations that better capture the underlying

Table 1. Twenty Initial Node Features Used in Our GNN

Type	# Dim.	Description
max output slew	1	max transition of output pin
max input slew	1	max transition of input pin(s)
wst output slack	1	worst slack of output pin
wst input slack	1	worst slack of input pin(s)
output cap limit	4	max driving cap of output pin per V_{th}
max leakage	4	max leakage per V_{th}
tot input cap	1	sum of input pin cap
tot fanout cap	1	output net cap + input pin cap of fan-outs
tot fanout slack	1	sum of worst slack of fan-outs
wst fanout slack	1	worst. slack of fan-outs
avg fanin cap	1	average cap of fan-ins
wst fanin slack	1	worst slack of fan-ins
tot sibling cap	1	sum of input pin cap of siblings
tot sibling slack	1	sum of worst slack of siblings

We obtain them using an initial PPA analysis.

Table 2. Dimension of Matrices Used in Our Work (See Figure 3)

Matrix	Meaning	Dimension
A	adjacency matrix of the netlist graph	$v \times v$
h_v^0	initial node features from PPA analysis	$v \times 20$
h_v^k	node embedding extracted by GNN	$v \times 128$
P	V_{th}-assignments from softmax function	$v \times 4$

v denotes the number of gates in the circuit.

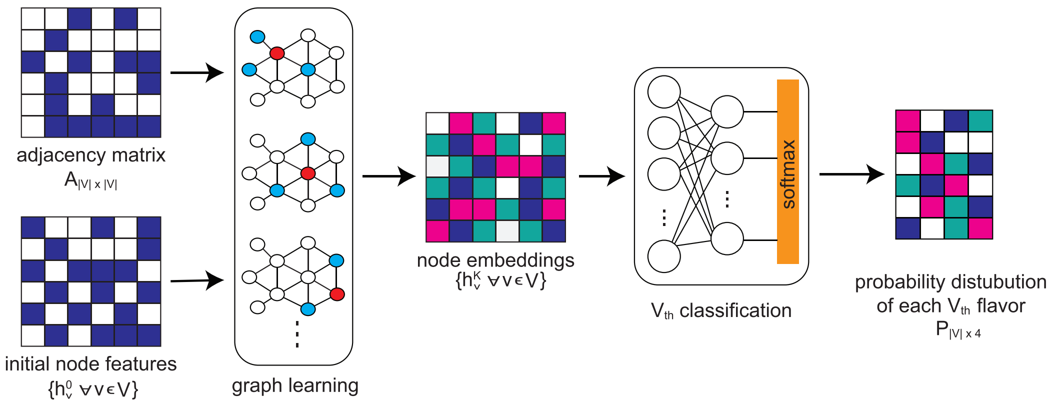

Fig. 3. Illustration of our ECO-GNN learning process. The inputs include a netlist graph represented in an adjacency matrix A and its initial features h_v^0 defined in Table 1. First, we perform graph learning to generate the node embeddings that represent the netlist better than the initial features. Figure 4 provides details of the GNN structure used. Next, with the learned node embeddings, we conduct softmax-based classification to determine the final V_{th}-assignment that optimizes the signoff power.

characteristics of the given netlist than the intial features. After graph learning, the learned representation vector of each node $v \in V$ is projected to a logit vector P_v through a softmax-based classification model, which is a neural network. The vector P_v represents the probability distribution of node v belonging to different V_{th} flavors that are available in the standard cell libraries.

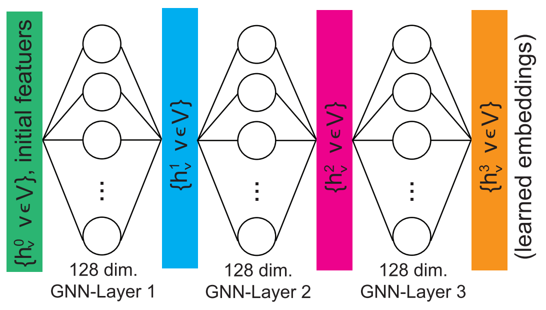

Fig. 4. Our GNN architecture that maps the initial node features (20) into learned embedding features (128).

Table 2 shows the size of matrices used in our framework. The adjacency matrix A represents the logical connections in the netlist, and the initial node features $\{h_v^0\ \forall v \in V\}$ are the cell attributes shown in Table 1. Note that the whole learning process, from graph learning to V_{th} classification, is end-to-end differentiable. Therefore, the parameters in the GNN and classification modules can be updated simultaneously using gradient descent.

6.2 GNN: Feature Aggregator

The goal of graph learning is to construct accurate node embeddings through effective feature aggregation. GNN functions as a feature aggregator that transforms the initial features h_v^0 for each node $v \in V$ into better representations h_v^K by *sampling and aggregating* the features within v's K-hop neighborhood. This aggregation process is performed iteratively, where for each hop $k \in \{1, \ldots, K\}$, a dedicated **neural network (NN)** $\mathbf{W}_k$ is developed to perform the transformation. These K dedicated NNs together form the GNN module in our framework as shown in Figure 4. Since the number of neighbors of a node scales exponentially as the hop-count increases, we fix the sampling size s_k at each hop k to improve the computational efficiency and to prevent overfitting.

Following the graph learning approach presented in Reference [6], in this work, for each node $v \in V$, we obtain its representation vector h_v^k at level[1] k by aggregating its representation h_v^{k-1} at the previous level with the features of its neighbors $N_k(v)$ sampled at k-hop as

$$\begin{aligned} h_{N_k(v)}^{k-1} &= \text{maxpool}\left(\{\mathbf{W}_k^{agg} h_u^{k-1}, \forall u \in N_k(v)\}\right), \\ h_v^k &= sigmoid\left(\mathbf{W}_k^{proj} \cdot \text{concat}[h_v^{k-1}, h_{N_v(v)}^{k-1}]\right), \end{aligned} \tag{2}$$

where W_k^{agg} and W_k^{proj} denote the aggregation and projection matrices, respectively, which together form the weights of the NN dedicated in *sampling and aggregating* features at the k-hop neighborhood. In the implementation, we set $k \in \{1, 2, 3\}$, and each NN (W_1, W_2, W_3) in the GNN module has an output dimension of 128. Note that the numbers 128 and 3 are chosen empirically based on parameter sweeping experiments.[2]

In summary, the initial feature vector h_v^0 for each node $v \in V$ is transformed to $h_v^{K=3}$ in R^{128}. The GNN model utilized in our framework can be considered as a "node filter," because it iterates

[1]Level is corresponding to the hop-count. When aggregating the features of a node at level k, the information within its k-hop neighborhood is considered.

[2]We varied them while monitoring the overall power saving vs. training time tradeoff. Due to the page limit, we omit the related experimental results. But, a general trend shows that the higher the values are, the more the power saving is at the cost of training time. But, the power saving saturates after some point.

through every design instance to find better node representations that can be utilized in the latter classification task of determining the V_{th}-assignments that optimize the design signoff power.

6.3 Loss Function

After leveraging GNN to perform graph representation learning, we take the learned node embeddings $\{h_v^K \in R^{128}, \forall v \in V\}$ as the inputs of our softmax-based classification model, which is a neural network, to determine the appropriate V_{th}-assignment for each design instance. As shown in Figure 3, the end of the classification model connects to a softmax function that outputs P, which is a $|V| \times n$ matrix denoting the probability of each node v belonging to n different V_{th} flavors, where $\forall v \in V, \sum_{c=1}^{n} P_{vc} = 1$. Note that n is limited to the discrete V_{th} values specified by the technology. The technology we utilize in this work is *TSMC 28 nm*, which has $n = 4$. A novelty of this work is that we map the discrete V_{th}-sizing problem into a multi-class classification problem, where the classification loss function is defined as

$$\mathcal{L} = -\sum_{i=1}^{|V|} \sum_{c=1}^{n} Y_{ic} log(P_{ic}), \tag{3}$$

where $Y \in R^{|V| \times n}$ denotes the V_{th}-assignments made by the *Synopsys PrimeTime* ECO engine, which are taken as ground-truths. Essentially, our loss function (Equation (3)) represents the cross-entropy between Y and P distributions. By minimizing Equation (3), we can update the parameters in the entire ECO-GNN framework.

6.4 Training Methodology

The training process of ECO-GNN is supervised, where we use the V_{th}-assignments obtained from *Synopsys PrimeTime* ECO engine as ground-truths and minimize a supervised loss function to update the GNN parameters. The main reason we discard traditional machine learning techniques as the ones used in previous works [2, 21] is that these techniques fail to consider the neighborhood information of an instance while determining the V_{th}-assignment, where the assignment certainly depends on the neighborhood structure such as the impacts of the propagated arrivals and transition effects.

Algorithm 1 summarizes the training process. Lines 3–10 illustrate the *sampling and aggregating* process in graph learning, where for each node $v \in V$, we aggregate its neighboring features at each hop $k \in K$ through Equation (2). Note that before performing each aggregation, we normalize the node representations at previous level as shown in Line 2 and Line 9. This normalization accelerates the overall training process by reducing the oscillation of gradient descent. Based on the learned representation vectors, in Lines 11–15 we calculate the cross-entropy loss (Equation (3)) from the softmax-based classification model and leverage a gradient descent optimizer named *Adam* [10] to update the parameters in the framework by minimizing the loss function. The overall training process takes about 12 hours on the nine training designs shown in Table 3 with a machine that has a 2.40-GHz CPU and a NVIDIA RTX 2070 graphic cards with 16 GB memory.

6.5 Complexity Analysis

The time complexity of ECO-GNN is linear with respect to the netlist size. Since the sampling size (s_k) at each aggregation level is constrained, GNN modules spend constant time in visiting every design instance and collecting features from its neighbors. Due to the large sparsity of the netlist adjacency matrix, we realize the adjacency matrix A shown in Table 2 in the compressed sparse row format [30]. Therefore, the space complexity of ECO-GNN is pseudo-linear rather than quadratic with respect to the netlist size, because it is mainly constrained by the number of nets of the underlying design. As shown in Algorithm 1, our framework conducts instance-based learning,

ALGORITHM 1: ECO-GNN training methodology.
We use default values of $K = 3, \alpha = 0.001, s_1 = 25, s_2 = 20, s_3 = 15, \beta_1 = 0.9, \beta_2 = 0.999$.

Input: (1) $G(V, E)$: netlist graph, (2) $A_{|V|\times|V|}$: adjacency matrix, (3) Y: tool optimization results, (4) n: number of available V_{th} flavors, (5) $\{h_v^0, \forall v \in V\}$: initial features. (6) K: depth of aggregation level, (7) $\{s_k, \forall k \in \{1, \ldots, K\}\}$: sampling size at k-hop neighborhood, (8) $\{\mathbf{W}_k, \forall k \in \{1, \ldots, K\}\}$: parameters of NN at hop k, (9) α: learning rate, (10) $\{\beta_1, \beta_2\}$: Adam parameters.
Output: $P_{|V|}$: V_{th}-assignment prediction of each instance.

```
1:  while $\{\mathbf{W}_k\}$ do not converge do
2:      $h_v^0 \leftarrow \frac{h_v^0}{\|h_v^0\|_2}, \forall v \in V$                ▹ initial features from Table 1
3:      for $k \leftarrow 1$ to $K$ do
4:          for $v \in V$ do                                  ▹ sample and aggregate by Equation (2)
5:              $N_k(v) \leftarrow$ Sample $s_k$ neighbors at $k$-hop
6:              $h^k_{N_k(v)} = \text{maxpool}(\{\mathbf{W}_k^{agg} h_u^{k-1}, \forall u \in N_k(v)\})$
7:              $h_v^k = sigmoid(\mathbf{W}_k^{proj} \cdot \text{concat}[h_v^{k-1}, h^k_{N_v(v)}])$
8:          end for
9:          $h_v^k \leftarrow \frac{h_v^k}{\|h_v^k\|_2}, \forall v \in V$                ▹ reduce gradient oscillation
10:     end for
11:     for $v \in V'$ do                                     ▹ minimize Equation (3)
12:         $p_v \leftarrow softmax(\mathbf{W}_k^{NN} \cdot f_v^K)$
13:         $g_v \leftarrow \nabla_\theta[\sum_{c=1}^n Y_{ic} log(p_{vc})]$
14:         $\{\mathbf{W}_k\} \leftarrow Adam(\alpha, \{\mathbf{W}_k\}, g_v, \beta_1, \beta_2)$
15:     end for
16: end while
```

where each instance (cell) in the design can be considered as a data point. Given a netlist $G = (V, E)$, therefore, after learning on the nine training designs presented in Table 3, which in total contain millions of data points, our framework achieves remarkable optimization results.

6.6 Handling Unseen Designs

A highlight of this work is that a trained ECO-GNN framework has the ability to perform commercial-quality signoff power optimization on *unseen* designs *at inference time*. This capability is independent of the netlist structure or the netlist size, because to determine the V_{th}-assignments that optimize the signoff power in an unseen design, we only need to take the initial features and the adjacency matrix as inputs, and ECO-GNN will determine the appropriate V_{th}-assignments through constant time inferencing. Unlike *PrimeTime* and previous works that require multiple iterations to determine the final V_{th}-assignments, our framework is a one-pass tool that generates tool-accurate results instantly.

6.7 A Regression Perspective: Subgraph Approximation for Fast Power Prediction

Up to now, we have presented a complete graph learning-based framework that can predict the final V_{th}-assignment of each design instance without actually running signoff power ECO, which is highly time-consuming. Specifically, we cast the ECO signoff power optimization prediction as a classification problem, where we present a GNN-based model that classifies each design instance to a specific V_{th} type. Nonetheless, in many real-world scenarios, designers are seeking a direct estimate of the actual power saving value (i.e., regression) before running any optimization or performing any netlist update using ECO change-lists. In addition, on certain occasions, designers would only want to conduct the ECO power optimization on a partial netlist rather than the whole

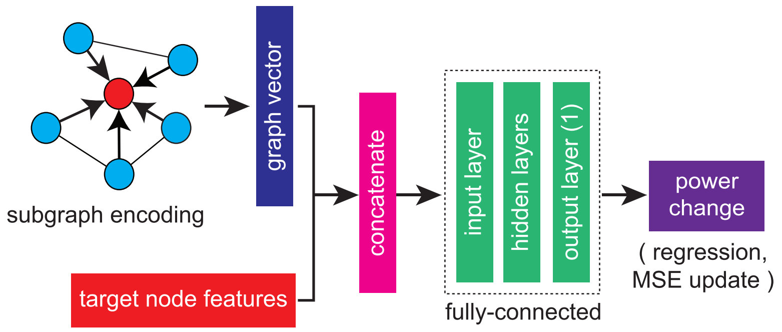

Fig. 5. Overview of subgraph approximation for power prediction of target instance (red-colored). Target nodes refer to the design instances that are selected for partial netlist update.

netlist. For instance, in modern industrial design flows, it is common to solely perform the signoff power optimization on the design instances whose slack values are larger than a pre-defined threshold, which is because the ECO changes made on these instances (with large positive slack values) will introduce the least timing impact to the overall placed and routed design. Therefore, in this section, we will present a new methodology that meets these needs of designers.

Although we can apply Algorithm 1 to overcome the above problem of partially optimizing the netlist by performing a full-graph (i.e., full-netlist) inference on every design instance, this computation will introduce excessively unnecessary computational resources and runtime given that the optimization is targeted on a few instances. To overcome this issue, we propose a new methodology, named "subgraph appoximation," where instead of using the information of the entire netlist to predict the final gate types of a few instances, we leverage GNN to encode the features from their local subgraphs.

Given a target instance $v \in G$, the subgraph sG_v of this instance v refers to its local three-hop neighborhood graph structure of the underlying netlist. For each selected instances V that meet the slack threshold for signoff power optimization, we leverage GNNs to perform subgraph encoding, where the goal is to construct a meaningful graph-level feature representations that capture the characteristics of the targeted instances. Note that the philosophy behind this subgraph-based approach is fundamentally different from the previous approach. The current approach focuses on learning "graph-level" vectors that characterize the information related to optimizing single design instance, where the previous approach (i.e., Algorithm 1) dedicates on learning the "node-level" embeddings that capture the interaction among all instances in the netlist. Based on the initial node features defined in Table 1, we first leverage Equation (2) to transform the initial features to high-dimensional representations. Then, for each node v in a subgraph sG, we obtain the graph-level vector s through

$$s = \text{concat}\left[mean\_pool\left(\left\{h_v^{k=K}\right\}\right), feat(v)\right], \tag{4}$$

where $feat(v)$ denotes the underlying features of the target instance v that is selected for the power optimization, which includes the current power consumption (internal and driving net switching power), cell capacitance (input pin cap), driving strength, and the worst transition values of input and output pins. The subgraph vector s, which characterizes the "local" information of the target instance v that is related to its power optimization, will be taken as the input of fully connected layers to directly predict the change of power consumption.

Finally, Figure 5 demonstrates an overview of the subgraph-based power prediction flow, where the goal is to predict the power difference before and after performing power optimization on the

ALGORITHM 2: Assisting V_{th}-assignment with subgraph approximation.

```
Input: (1) $G(V,E)$: netlist graph, (2) $A_{|V|\times|V|}$: adjacency matrix, (3) $t$: target instance, (4) $\mathbf{W}_{ECO-GNN}$:
    weights of ECO-GNN, (5) $\mathbf{W_{nn}}$: weights of feedfoward neural networks, (6) $Y$: tool optimization results
Output: $\{p_t\}$: $V_{th}$-assignment of instance $t$.
1: while {W} do not converge do
2:     $s_t \leftarrow$ 3-hop local subgraph of instance $t$
3:     $g_t \leftarrow$ subgraph encoding of $s_t$                    ▷ graph-level vector as in Figure 5
4:     $h_t \leftarrow$ ECO-GNN$(G, A; \mathbf{W}_{ECO-GNN})$        ▷ node embeddings of cell $t$ from full-graph learning
5:     $c_t \leftarrow$ concat$[g_t, h_t]$                ▷ concatenate graph-level vector with node embeddings
6:     $p_t \leftarrow softmax(\mathbf{W}_{nn} \cdot c_t)$
7:     $g_t \leftarrow \nabla_\theta[\sum_{c=1}^{4} Y_{ic} log(p_{tc})]$
8:     $\{\mathbf{W}\} \leftarrow Adam(\alpha, \{\mathbf{W}\}, g_t)$
9: end while
```

target instance colored in red. The detail steps are as follow. First, we leverage GNN to preform node representation learning on the local three-hop neighborhood subgraph of the target instance, which includes the fanout/fanin cells up to three levels and the sibling cells up to two levels. This node representation learning will transforms the initial features of each cell in the subgraph into high-dimensional representations (128 dimensions). Then, a global mean pooling is performed across each cell in the subgraph to obtain a graph-level vector, which is expected to represent the ECO-related characteristics of the target instance. Finally, the graph-level vector, along with the initial feature vector of the targeted instance, is fed to a downstream feed-forward neural network to predict the actual power saving and mean-squared-error is utilized as the loss function to train the entire framework.

6.8 Assisting V_{th}-Assignment with Subgraph Approximation

The V_{th}-assignment problem is a classification task by nature as the available gate sizes are discrete, which we solve by leveraging GNNs to encode full-netlist information as shown in Algorithm 1. However, as aforementioned, the purpose of the proposed subgraph approximation technique is to perform regression-based power prediction by merely using partial-netlist information. Therefore, to assist the V_{th}-assignment task with the subgraph approximation technique, a learning methodology needs to be developed to bridge the gap between the classification and the regression tasks.

Algorithm 2 summarizes how the subgraph approximation technique can be leveraged to assist the traditional V_{th}-assignment task, where the key idea is to leverage the encoded graph-level vector from subgraph approximation as additional information to help predict the final V_{th} type of the target instance. The detail steps are as follow. First, we leverage the subgraph approximation technique to encode the local three-hop neighborhood subgraph of the target instance t to obtain a graph-level vector g_t (Lines 2 and 3). Then, we leverage the aforementioned ECO-GNN framework to perform node representation learning and obtain the learned embeddings h_t of the target instance (Line 4). Finally, we concatenate the graph-level vector (describing the local neighborhood structure) and the learned embeddings as input to the downstream feedforward classification network to predict the final V_{th} assignment. Note that the entire algorithm is end-to-end differentiable.

7 EXPLAINING PREDICTION RESULTS

Understanding the reasons behind the predictions of ML models can give users better trust in the models. In this work, we explore explanation techniques that provide insights of the V_{th}-assignment predictions made by the proposed framework ECO-GNN. Given a targeted instance for explanation, we will explore its local subgraph to determine what are the important factors in terms of neighboring nodes and connected nets that drive the prediction of the proposed graph learning-based framework.

7.1 Inner Workings of GNN Predictions

Unlike previous works who consider their optimization engines as blackboxes, in this article, we implement a GNN-based explanation method [32] to interpret the V_{th}-assignment predictions made by our framework ECO-GNN. Given a set of target instances $\{v\} \in V$ in a netlist graph $G = (V, E)$, the goal is to find an influential sub-graph $G_S = (V_S, E_S)$ that has high contribution to the decision of $\{v\}$'s V_{th}-assignments. The objective of finding such sub-graph G_S can be quantitatively formulated as maximizing the **mutual information (MI)** between the original graph G and the sub-graph G_S as

$$\max_{G_S} MI(G, G_S) = H(Y) - H(Y|G = G_S), \tag{5}$$

where $H(\cdot)$ denotes the entropy of the given distribution and Y represents the V_{th} prediction distribution of the target instances. Since $H(Y)$, the entropy of the prediction distribution based on the original graph, is a constant, maximizing Equation (5) is equivalent to minimizing the conditional entropy $H(Y|G = G_S)$, which can be formulated as

$$H(Y|G = G_S) = -\mathbb{E}_{Y|G=G_S}\left[log\left(P_\theta(Y|G = G_S)\right)\right], \tag{6}$$

where θ denotes the parameters of the trained ECO-GNN framework. Note that due to the fact that the number of neighbors of the target nodes increases exponentially as the hop-count increases, in the implementation, we constrain G_S to search within the one-hop neighbors of the target instances $\{v\}$. In the context of the actual netlist, G_S represents the cells that are either the fanins, fanouts, or siblings of $\{v\}$ as well as the message passing flows (edge connectivities) that demonstrate how important features are aggregated. We believe this interpretability would give designers precious insights on what the framework has learned and whether the V_{th}-assignments are reliable or not.

8 EXPERIMENTAL RESULTS

In this section, we demonstrate the achievements of our ECO-GNN framework, which is implemented in *Python3* with *Tensorflow 1.0* library. We leverage 7 designs from the ISPD-2012 benchmark [19] and 7 other industrial designs to conduct the experiments. All 14 designs are synthesized under *TSMC 28 nm* technology node by *Synopsys Design Compiler 2015* and placed and routed using *Cadence Innovus v18.1*. To validate the signoff power optimization results of ECO-GNN, we use *Synopsys PrimeTime 2018* to perform timing and power analysis and consider the *PrimeTime* ECO engine as the baseline across all experiments.

8.1 Benchmarks Details and Timing Corners

As mentioned in Section 2, due to the unrealistic nature of the ISPD-2012 benchmark that the worst negative slacks in the original designs range from −1 ns to −8 ns, we re-implement all seven ISPD designs using *TSMC 28 nm* technology node and commercial PD tools. Aside from the ISPD benchmarks, we introduce 7 other renowned industrial designs, including JPEG, TATE, LDPC, AES-128, NOVA, ECG from *OpenCores.org*, and RocketCore [1], which is a RISC-V-based multi-core system. To substantiate the generality of our framework, we utilize 9 designs in the training process and perform the validations on the 5 *unseen* ones. The characteristics of these 14 designs are shown in Table 3. In the table, we also demonstrate the graph-related statistics, which

Table 3. Our Benchmarks and Their Attributes in *TSMC 28 nm*

Design Name	# Nets	# FFs	# Cells	MPL	SR	RCC	Usage
RocketCore	93,812	16,784	90,859	68	381	7	training
AES-128	90,905	10,688	113,168	17	68	12	
NOVA	138,171	29,122	136,537	32	185	3	
ECG	85,058	14,018	84,127	29	76	8	
LDPC	42,018	2,048	39,377	23	229	14	
DMA	10,898	2,062	10,215	15	29	52	
PCI_BRIDGE	1,381	310	1,221	24	33	307	
DES_PERF	48,523	8,802	48,289	17	28	29	
B19	34,399	3,420	33,784	35	29	22	
TATE	185,379	31,409	184,601	31	26	5	testing
JPEG	231,934	37,642	219,064	26	173	4	
VGA_LCD	56,279	17,054	56,194	24	25	19	
LEON3MP	341,263	108,724	341,000	48	28	3	
NETCARD	317,974	87,317	316,137	37	31	4	

MPL denotes the maximum path length of timing paths, SR denotes the spectral radius of the adjacency matrix, and RCC denotes the Rich Club Coefficient (10^-4).

Table 4. Subgraph Approximation Prediction Results on Unseen Benchmarks

Unseen design		VGA	JPEG	TATE	LEON
NRMSE %		2.2	2.8	1.7	1.4
CC		0.96	0.97	0.97	0.98
Total power (mW)	Before	212.7	376.8	345.0	576.6
	After	197.9	342.3	328.7	552.4
WNS (ps)	Before	−3.4	−13.1	−2.4	−16.3
	After	−3.1	−12.8	−2.3	−16.2
TNS (ps)	Before	−14.1	−228.7	−3.2	−246.0
	After	−12.4	−202.5	−3.0	−239.3

CC denotes the Pearson correlation coefficient and is calculated against the *ICC2* optimization results.

include the maximum path length of timing paths, the spectral radius (i.e., maximum eigenvalue of adjacency matrix), and the Rich Club Coefficient (an indicator of connectivity).

Following the experimental settings of the ISPD-2012 contest where all the designs are synthesized with one timing corner and one V_{th} flavor that has the tightest timing constraint, in this work, we synthesize all the designs using typical corner and ultra-low V_{th} flavor (tightest timing constraint) in *TSMC 28 nm* for fair comparisons. In the *PrimeTime* ECO for signoff power optimization, each design instance is enabled to be swapped into one of the three other V_{th} flavors, which are low, high, and ultra-high types, or remain as the ultra-low type (four choices in total). Therefore, the solution space of our V_{th}-assignment problem is $4^{|V|}$, which is almost impossible for designers to perform design exploration in an exhaustive manner.

8.2 Subgraph Approximation Results

As aforementioned, in this article, we not only develop a classification-based model to predict the final V_{th} type of each design instance using entire netlist information, we also present the subgraph approximation technique to estimate the power recovery of individual instances during signoff ECO. Table 4 demonstrates the prediction results on four unseen designs. In this article,

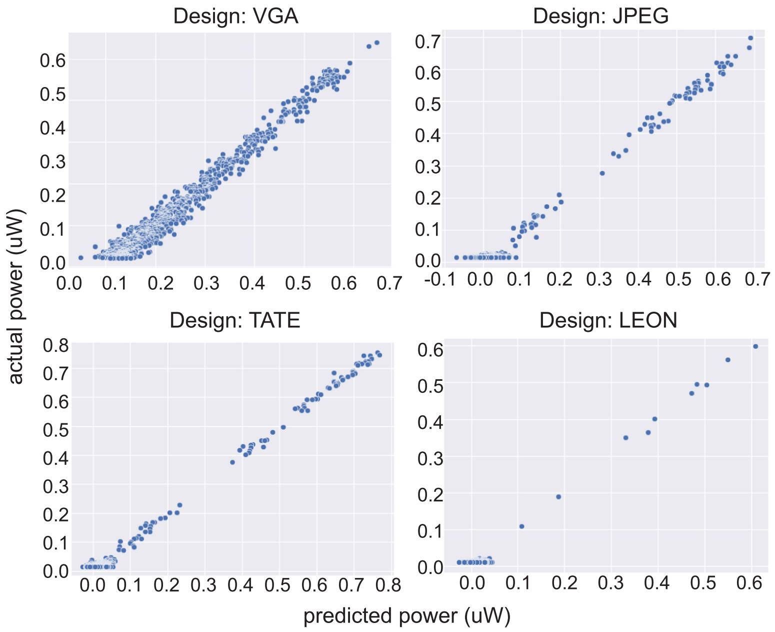

Fig. 6. Prediction results of subgraph approximation. We validated the subgraph model on 4 unseen designs. Each dot in the plots represents a design instances whose initial slack is above 200 ps before ECO power optimization.

we specifically focus on two metrics to evaluate the model: **normalized root-mean-squared error (NRMSE)** and the **correlation coefficient (CC)**. Note that NRMSE is calculated by normalizing the RMSE, which inherently comes with a "unit" (e.g., mW) by the difference between the maximum and minimum ground-truth values (i.e., $NRMSE = \frac{RMSE}{power_{max} - power_{min}}$). NRMSE is a popular comparison metric that removes the effect of unit scale. As shown in the figure, we observe that the proposed model consistently delivers highly accurate prediction results across the unseen benchmarks. Finally, Figure 5 shows the scatter distribution of the prediction results, where each dot represents an actual subgraph whose worst slack value before the optimization is 200 ps.

8.3 Prediction Results of V_{th}-Assignment with Subgraph Approximation

In this experiment, we demonstrate the effectiveness of using the proposed subgraph approximation technique to improve the prediction task of V_{th}-assignment. Table 5 demonstrates the detailed prediction results in the format of confusion matrices. The upper table shows the results without using the subgraph approximation technique, and the lower table demonstrates the results achieved with subgraph approximation using Algorithm 2. In the table, we observe that the subgraph approximation technique can indeed boost the prediction accuracy, which is expected as more information (e.g., the graph-level vector) is curated for the model to make better predictions.

8.4 Discussion of Subgraph Approximation

One of the highlights of this article is the proposed concept of subgraph approximation, which enables fast and accurate prediction of power optimization. The rationale behind the proposed subgraph approximation technique is twofold. First, given that power and timing are often interrelated with each other and the power recovery in ECO often comes under the sacrifice of timing degradation, the final V_{th} type of a target instance v will depends on not only its direct one-hop

Table 5. Confusion Matrix Comparison of V_{th}-assignment with and without Subgraph Approximation on the VGA Benchmark

		Predictions (accuracy: 0.94)				
	w.o. subgraph	ultra-low	low	high	ultra-high	Total
Ground -truths	ultra-low	429	17	53	82	581
	low	22	1557	144	202	1,925
	high	14	18	5026	166	5,224
	ultra-high	45	111	189	9110	9,455
	Total	510	1703	5412	9560	17,185

		Predictions (accuracy: 0.96)				
	w. subgraph	ultra-low	low	high	ultra-high	Total
Ground -truths	ultra-low	518	26	23	14	581
	low	12	1645	188	80	1,925
	high	21	93	4973	137	5,224
	ultra-high	37	19	124	9275	9,455
	Total	588	1783	5308	9506	17,185

Each count represents an instance whose slack value is greater than 200 ps before the *PrimeTime* ECO optimization.

neighbors but also other neighbors that may or may not locate on the same timing paths. Therefore, we leverage GNNs to encode such neighboring information for the final prediction of its power recovery. Second, the reason we do not select a huge number of hops to perform the subgraph encoding is because the QoR impact of a single gate sizing move on a design instance to the overall netlist diminishes quickly as the hop count increases.

8.5 Optimization Results on Unseen Designs

In this experiment, we compare the signoff power optimization results achieved by our framework ECO-GNN with the commercial tool *Synopsys PrimeTime*. To substantiate the generality of ECO-GNN, we only use nine designs for training, and perform validations on the five unseen ones as shown in Table 3. Note that to perform meaningful and reasonable signoff power optimization, each design is originally implemented in signoff frequency, where the WNS is close to 0. The optimization constraints are that the WNS and TNS do not degrade and no violation is introduced after the optimization.

Table 6 demonstrates the optimization results. Compared with *PrimeTime*, ECO-GNN achieves up to 14× runtime improvement with similar optimization quality. Unlike previous works that do not utilize commercial signoff tools for validations, we demonstrate that our framework performs tool-accurate signoff power optimization without degrading the original signoff performance of each unseen design. Note that each design in Table 6 has different target frequencies, which proves that the optimization achieved is not confined by design characteristics. The inference time of ECO-GNN is measured on a machine with 2.40-GHz CPU and a NVIDIA RTX 2070 graphics card with 16-GB memory, where *Synopsys PrimeTime* is ran on a machine with 2.50-GHz CPU and 8 cores enabled.

Table 6 also reports the micro F1-score as the evaluation metric of the classification task, owing to the fact that our framework ECO-GNN is performing supervised learning that we take the V_{th}-assignments from *Synopsys PrimeTime* as ground-truths in the training process. Note that micro F1-score represents the accuracy of multi-class classification. In the table, we observe that ECO-GNN performs the the V_{th}-assignments in high fidelity as *PrimeTime*.

Table 6. V_{th} Re-assignment Impact on Power, Timing, and Runtime between ECO-GNN and *Synopsys PrimeTime*

Design	Target frequency	Optimization Engine	Leakage power (*mW*)	Total power (*mW*)	WNS (*ps*)	TNS (*ps*)	Runtime (*sec*)	F1-score (*micro*)
TATE	1.2 GHz	Before Opt.	38.3	345.0	−2.4	−3.2	—	0.90
		PrimeTime	1.84	282.7	−0.5	−0.6	141	
		ECO-GNN	1.72	280.6	−0.9	−1.8	16 (9×)	
JPEG	1.1 GHz	Before Opt.	57.5	376.8	−13.1	−228.7	—	0.85
		PrimeTime	3.4	294.6	−5.7	−69.6	120	
		ECO-GNN	3.8	296.9	−11.4	−182.4	15 (8×)	
VGA_LCD	1.8 GHz	Before Opt.	18.0	212.7	−3.4	−14.1	—	0.89
		PrimeTime	3.7	184.6	−2.6	−4.4	69	
		ECO-GNN	3.5	183.3	−3.2	−11.8	5 (14×)	
LEON3MP	700 MHz	Before Opt.	101.4	576.6	−16.3	−246.0	—	0.88
		PrimeTime	15.1	459.8	−8.4	−76.7	341	
		ECO-GNN	12.2	454.9	−12.8	−209.3	28 (12×)	
NETCARD	1 GHz	Before Opt.	78.1	651.5	−2.4	−4.2	—	0.86
		PrimeTime	9.9	544.3	−0.8	−1.1	302	
		ECO-GNN	6.9	537.6	−1.2	−2.7	26 (12×)	

Selected designs are *unseen* during training. Note that both leakage and total power reduce from V_{th} re-assignment, because our initial designs before ECO optimization are using ultra-low V_{th} only as suggested in Reference [19]. Timing also improves because of the gate capacitance reduction from higher V_{th}.

Table 7. Sweeping Experiments on Maximum Number of Aggregation Level (K) of GNN

Designs (F1-score)	$K = 1$	$K = 2$	$K = 3$	$K = 4$	$K = 5$
TATE	0.39	0.78	0.90	0.82	0.73
JPEG	0.33	0.74	0.85	0.81	0.70
VGA_LCD	0.42	0.81	0.89	0.85	0.74
LEON3MP	0.36	0.84	0.88	0.79	0.66
NETCARD	0.41	0.69	0.86	0.83	0.72

The entry represents the F1 score of the classification results.

Finally, due to the fact that V_{th}-assignments directly optimize the design leakage power, Figure 8 further shows the instance-based leakage power consumption maps of the unseen designs, which are corresponding to the optimization results presented in Table 6. In the figure, we compare the leakage power consumption of each instance in the original designs with the ones after using ECO-GNN to perform signoff power optimization. Across all designs, we observe that ECO-GNN effectively reduces the overall leakage power without introducing extra hotspots.

8.5.1 Sweeping Experiments on GNN Aggregation Level. In the realm of graph learning using GNNs, choosing a right number of the maximum aggregation level is critical to the success of representation learning. Nonetheless, it is known that GNNs tend to suffer from the over-smoothing problem [3], which is an issue that the representations among different nodes become indistinguishable and thus the prediction accuracy becomes worse. Therefore, for the majority of GNN applications, the number of aggregation level is empirically set to a number between 2 and 4 (inclusive) [33]. In this article, we validate the hypothesis by providing sweeping experiments over the GNN aggregation level as shown in Table 7, where we clearly observe that in the classification task of predicting final V_{th} type for each design instance, the best accuracy occurs when the number of aggregation level is set to 3.

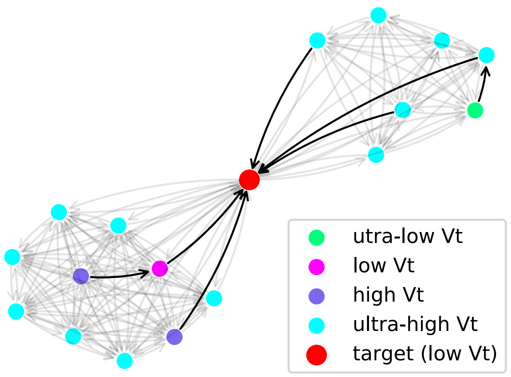

Fig. 7. Graph learning explanation on b19 benchmark. The majority of the neighbors are ultra-high V_{th}, but cells with lower V_{th} types have higher importance to the target node. As a result, low V_{th} is assigned to the target node.

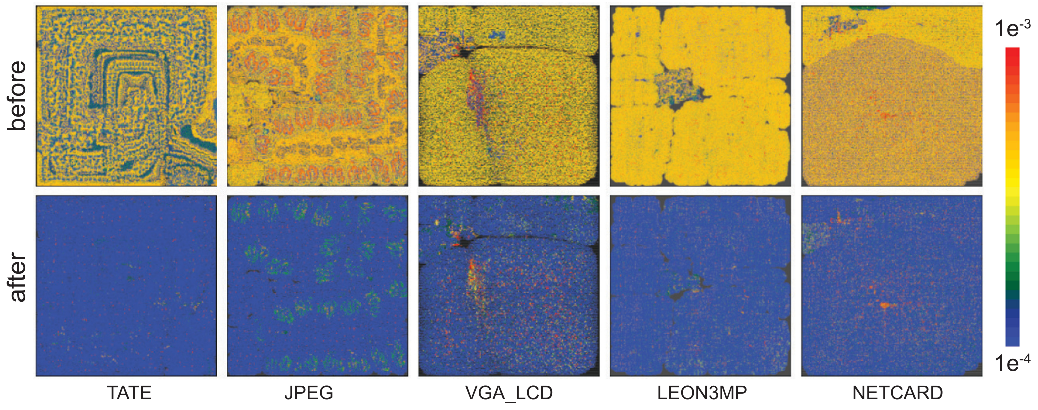

Fig. 8. Leakage power consumption of each design instance before and after using ECO-GNN to perform optimizations. The designs are *unseen* during training, and the unit is mW.

8.6 Discussion of Optimization Results

Power Perspective. As shown in Table 6, the optimizations through V_{th}-assignments achieved by our framework and *PrimeTime* improve both leakage power and total signoff power. This is because we follow the experimental settings from the ISPD-2012 contest [19] as many previous works [8, 17, 21–23, 25, 26]. The setting suggests all the cells to be in ultra-low V_{th} (tightest timing constraint) before the optimization. Therefore, for a design instance, a swap from ultra-low V_{th} to other V_{th} types in *TSMC 28 nm* not only improves its static power (leakage) but also the dynamic power as the capacitance load is reduced.

Timing Perspective. As shown in the table, we observe that the WNS and TNS get improved as well. This comes from the fact that although *PrimeTime* will not upsize the V_{th} type of the cells that are on critical (negative slack) paths, the driving load of such cells may still be reduced if some of its fanout cells that are not on critical paths are swapped to higher V_{th} types, which in the end improves the overall timing as a by-product.

8.7 GNN Explanation

Instead of viewing our framework ECO-GNN as a blackbox, we validate our optimization results by explaining the V_{th}-assignments made by our framework. Figure 7 demonstrates the explanation results on the b19 design, where we plot the graph learning computational graph centered on the target node colored in red along with its neighbors using force-directed placement drawing [4]. Note that although we present single-instance explanation in this experiment for clarity, the proposed explanation method can be leveraged to perform the explanation of multi-instance as well.

To explain the V_{th}-assignment on the target node (red), we identify the important message passing flows within the local sub-graph. As mentioned in Section 7.1, we constrain the explanation method to search within the one-hop neighborhood. Therefore, every neighboring node in Figure 7 is either the fanin, fanout, or sibling of the target node. However, as shown in the figure, the influential features may not be passed directly from the neighbors to the target even though they are one-hop neighbors. This is because the message passing scheme in graph learning is bi-directional. For better illustration, in Figure 7, we plot two directed edges for each bi-directional edge in the graph learning computational graph to show how the influential features are being passed.

Figure 7 shows that the V_{th}-assignment made by ECO-GNN on the target node is reliable, because we observe that the final V_{th} type of the target node is more influenced by its minority neighbors who are in lower V_{th} types rather than the majority neighbors that are in the ultra-high V_{th} type. This aligns well with common design knowledge. Since cells in lower V_{th} types have larger capacitance, tighter constraints will be imposed on their drivers compared with cells in high-level V_{th} types. Therefore, we conclude that the V_{th}-assignment made by ECO-GNN on the target cell is reliable.

8.8 Why Does ECO-GNN Work?

In the experiments, we demonstrate that ECO-GNN achieves commercial quality signoff power optimization results with negligible runtime compared with *Synopsys PrimeTime*. The achievements of our framework can be accounted by two reasons. First, the initial modeling features (Table 1) accurately capture the underlying characteristics of each design instance that are related to the signoff power optimization. Specifically, the timing related features provide solid information for our framework to select appropriate V_{th}-assignments that optimize signoff power with the consideration of timing budget. Second, GNNs are highly powerful for solving the optimization problems on graphs. The final V_{th}-assignment of an instance highly depends on the information of its neighborhood structure. Therefore, unlike previous works [2, 21] who use traditional machine learning techniques to predict the V_{th}-assignment of an instance solely based on its handcrafted features, our framework acts as a graph filter that aggregates an instance's neighboring information to more accurately determine its final V_{th}-assignment through the classification model. Finally, with the validations from the explanation method, we conclude that this work successfully presents a solution to the long-lasting V_{th}-assignment problem.

In spite of the superior performance achieved, we still see some limitations of the proposed framework. We observe that *Synopsys PrimeTime* consistently delivers better timing results, and ECO-GNN does not consistently improve the signoff power from the commercial tool. In fact, this is resulted from the modeling errors occurred in the learning process. Although we take the V_{th}-assignments from *Synopsys PrimeTime* as the ground-truths, there always exists a gap between the predictions of our framework and the actual assignments made by the tool. Nonetheless, the goal of this work is *not to replace* commercial signoff tools, but to provide PD engineers a fast, accurate, and reliable estimation of the amount of power recovery to expect from the signoff tools.

8.9 Related Works on Improving Chip Design Turnaround Time

Although the main focus of this work is to improve the turnaround time of signoff power optimization, related works have been developed to improve chip design productivity in different endeavors. Recently, as the hardware resources become more powerful, GPU-accelerated algorithms have been developed to significantly improve runtime of different tasks [9]. A previous work [5] develops a novel static timing analysis engine on a GPU-CPU hybrid system that greatly achieves a 3.6× speed-up on designs with over million of cells. Another work [13] further leverages GPU and deep learning toolkit, PyTorch, to advance placement, where the runtime is improved by 30× without degrading solution quality. As the technology scaling continuously increases the design complexity, in the future, methodologies to improve design turnaround time will be ever-critical.

9 CONCLUSION

In this article, we have proposed two different ML modeling approaches in classification and regression aspects to overcome the inherent challenges of the current signoff power optimization flow. The proposed framework, ECO-GNN, not only can provides commercial-quality tool-accurate signoff power optimization results *instantly* on unseen designs but also has the ability to estimate the power savings of targeted instances using information obtained from local graph structure. Furthermore, we present a GNN-based explanation method to demonstrate the reliability of our framework, which gives designers better reasonings about the predictions of the models.

REFERENCES

[1] Krste Asanovic, Rimas Avizienis, Jonathan Bachrach, Scott Beamer, David Biancolin, Christopher Celio, Henry Cook, Daniel Dabbelt, John Hauser, Adam Izraelevitz, et al. 2016. *The Rocket Chip Generator*. EECS Department, University of California, Berkeley, Technical Report UCB/EECS-2016-17.

[2] Shuhan Bao. 2010. *Optimizing Leakage Power Using Machine Learning*. Stanford University.

[3] Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. 2020. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 3438–3445.

[4] Thomas M. J. Fruchterman and Edward M. Reingold. 1991. Graph drawing by force-directed placement. *Softw.: Pract. Exp.* 21, 11 (1991), 1129–1164.

[5] Zizheng Guo, Tsung-Wei Huang, and Yibo Lin. 2020. Gpu-accelerated static timing analysis. In *Proceedings of the 39th International Conference on Computer-Aided Design*. 1–9.

[6] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*. 1024–1034.

[7] Masanori Hashimoto, Hidetoshi Onodera, and Keikichi Tamaru. 1998. A power optimization method considering glitch reduction by gate sizing. In *Proceedings of the International Symposium on Low Power Electronics and Design*. 221–226.

[8] Jin Hu, Andrew B. Kahng, SeokHyeong Kang, Myung-Chul Kim, and Igor L. Markov. 2012. Sensitivity-guided metaheuristics for accurate discrete gate sizing. In *Proceedings of the International Conference on Computer-Aided Design*. 233–239.

[9] Tsung-Wei Huang, Dian-Lun Lin, Chun-Xun Lin, and Yibo Lin. 2021. Taskflow: A lightweight parallel and heterogeneous task graph computing system. *IEEE Trans. Parallel Distrib. Syst.* 33, 6 (2021), 1303–1320.

[10] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. arXiv:1412.6980. Retrieved from https://arxiv.org/abs/1412.6980.

[11] Wonjae Lee, Yonghwi Kwon, and Youngsoo Shin. 2020. Fast ECO leakage optimization using graph convolutional network. In *Proceedings of the Great Lakes Symposium on VLSI*. 187–192.

[12] Li Li, Peng Kang, Yinghai Lu, and Hai Zhou. 2012. An efficient algorithm for library-based cell-type selection in high-performance low-power designs. In *Proceedings of the International Conference on Computer-Aided Design*. ACM, 226–232.

[13] Yibo Lin, Shounak Dhar, Wuxi Li, Haoxing Ren, Brucek Khailany, and David Z. Pan. 2019. Dreamplace: Deep learning toolkit-enabled gpu acceleration for modern vlsi placement. In *Proceedings of the 56th Annual Design Automation Conference 2019*. 1–6.

[14] Yifang Liu and Jiang Hu. 2010. A new algorithm for simultaneous gate sizing and threshold voltage assignment. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst.* 29, 2 (2010), 223–234.

[15] Yi-Chen Lu, Siddhartha Nath, Sai Surya Kiran Pentapati, and Sung Kyu Lim. 2020. A fast learning-driven signoff power optimization framework. In *Proceedings of the IEEE/ACM International Conference on Computer Aided Design (ICCAD'20)*. IEEE, 1–9.

[16] Uday Mallappa and Chung-Kuan Cheng. 2021. GRA-LPO: Graph convolution based leakage power optimization. In *Proceedings of the 26th Asia and South Pacific Design Automation Conference (ASP-DAC'21)*. IEEE, 697–702.

[17] Santiago Mok, John Lee, and Puneet Gupta. 2013. Discrete sizing for leakage power optimization in physical design: A comparative study. *ACM Trans. Des. Autom. Electr. Syst.* 18, 1 (2013), 15.

[18] Wing Ning. 1994. Strongly NP-hard discrete gate-sizing problems. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst.* 13, 8 (1994), 1045–1051.

[19] Muhammet Mustafa Ozdal, Chirayu Amin, Andrey Ayupov, Steven Burns, Gustavo Wilke, and Cheng Zhuo. 2012. The ISPD-2012 discrete cell sizing contest and benchmark suite. In *Proceedings of the ACM International Symposium on International Symposium on Physical Design*. ACM, 161–164.

[20] Muhammet Mustafa Ozdal, Steven Burns, and Jiang Hu. 2011. Gate sizing and device technology selection algorithms for high-performance industrial designs. In *Proceedings of the International Conference on Computer-Aided Design*. IEEE Press, 724–731.

[21] SLPSK Patanjali, Milan Patnaik, Seetal Potluri, and V. Kamakoti. 2018. MLTimer: Leakage power minimization in digital circuits using machine learning and adaptive lazy timing analysis. *J. Low Power Electr.* 14, 2 (2018), 285–301.

[22] Mohammad Rahman and Carl Sechen. 2012. Post-synthesis leakage power minimization. In *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE'12)*. IEEE, 99–104.

[23] Tiago Reimann, Gracieli Posser, Guilherme Flach, Marcelo Johann, and Ricardo Reis. 2013. Simultaneous gate sizing and vt assignment using fanin/fanout ratio and simulated annealing. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS'13)*. IEEE, 2549–2552.

[24] Sanghamitra Roy, Weijen Chen, Charlie Chung-Ping Chen, and Yu Hen Hu. 2007. Numerically convex forms and their application in gate sizing. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst.* 26, 9 (2007), 1637–1647.

[25] Subhendu Roy, Derong Liu, Junhyung Um, and David Z. Pan. 2015. OSFA: A new paradigm of gate-sizing for power/performance optimizations under multiple operating conditions. In *Proceedings of the Design Automation Conference*. ACM, 129.

[26] Ankur Sharma, David Chinnery, Sarvesh Bhardwaj, and Chris Chu. 2015. Fast Lagrangian relaxation based gate sizing using multi-threading. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD'15)*. IEEE, 426–433.

[27] Jaskirat Singh, Vidyasagar Nookala, Zhi-Quan Luo, and Sachin Sapatnekar. 2005. Robust gate sizing by geometric programming. In *Proceedings of the 42nd Design Automation Conference*. IEEE, 315–320.

[28] S. Sirichotiyakul, T. Edwards, C. Oh, and J. Zuo. 2005. PrimeTime User Guide: Fundamentals.

[29] Kai Wang and Peng Cao. 2022. A graph neural network method for fast ECO leakage power optimization. In *Proceedings of the 27th Asia and South Pacific Design Automation Conference (ASP-DAC'22)*. IEEE, 196–201.

[30] James B. White and Ponnuswamy Sadayappan. 1997. On improving the performance of sparse matrix-vector multiplication. In *Proceedings of the 4th International Conference on High-Performance Computing*. IEEE, 66–71.

[31] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S. Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* (2020).

[32] Rex Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. 2019. Gnn explainer: A tool for post-hoc explanation of graph neural networks. arXiv:1903.03894. Retrieved from https://arxiv.org/abs/1903.03894.

[33] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2020. Graph neural networks: A review of methods and applications. *AI Open* 1 (2020), 57–81.

Received 16 April 2022; revised 7 September 2022; accepted 12 October 2022