Boosting the Performance of Transistor-Level Circuit Simulation with GNN (Invited Paper)

Jiqing Jiang
SSSLab, Dept. of CST
China University of Petroleum-Beijing, China
jiqingjiang@student.cup.edu.cn

Yongqiang Duan
SSSLab, Dept. of CST
China University of Petroleum-Beijing, China
yongqiangduan@student.cup.edu.cn

Zhou Jin
SSSLab, Dept. of CST
China University of Petroleum-Beijing, China
jinzhou@cup.edu.cn

Abstract

Efficiently solving DC operating points for large-scale nonlinear circuits in SPICE simulation is both critical and challenging. Pseudo transient analysis (PTA) is a widely used and promising approach for DC analysis, with the pseudo element embedding strategy playing a key role in ensuring convergence and simulation efficiency. In this paper, we present GPTA, a graph neural network (GNN) enhanced PTA method that adaptively positions embeddings by considering circuit topology. GPTA transforms the nonlinear DC circuits into linearized graph representations and then integrates multi-head messaging, adaptive message filtering, and multi-scale information fusion in the GNN model to improve feature extraction. Additionally, a layer-by-layer pooling and prediction strategy effectively retains intermediate layer information, enhancing model expressiveness. Numerical results show that GPTA significantly improves the efficiency of DC analysis in terms of both convergence and simulation speed.

Keywords

SPICE simulation, DC analysis, Pseudo transient analysis, Graph neural network

ACM Reference Format:
Jiqing Jiang, Yongqiang Duan, and Zhou Jin. 2025. Boosting the Performance of Transistor-Level Circuit Simulation with GNN (Invited Paper). In *30th Asia and South Pacific Design Automation Conference (ASPDAC '25), January 20–23, 2025, Tokyo, Japan.* , 7 pages. https://doi.org/10.1145/3658617.3703149

1 Introduction

Direct current (DC) analysis is a pivotal step in SPICE simulation for finding the static operating points, which provides an initial solution for transient analysis and determines small-signal parameters for AC analysis[1]. The DC analysis process consists of solving a set of nonlinear algebraic equations generated from the circuit netlist using the modified nodal analysis (MNA) approach [2–10]. However, with the increasing integration and complexity of integrated circuits growing exponentially, the algebraic system to be solved in DC analysis becomes quite large and highly nonlinear. The basic Newton-Raphson (NR) [11] method is no longer suitable for these challenges. As a result, various continuation methods for DC analysis have been extensively investigated, including Gmin stepping [12], source stepping [13], homotopy methods [14–17], and pseudo transient analysis (PTA) [18].

Among the various continuation methods, PTA and its variants, including Pure PTA (PPTA) [19, 20], Damped PTA (DPTA) [21, 22], Ramping PTA (RPTA) [23, 24], and Compound Element PTA (CEPTA) [25, 26], have been proved to be the most promising algorithms for solving large scale strongly nonlinear DC analysis problem [27]. The PTA algorithm converts a nonlinear algebraic system into an ordinary differential equation (ODE) by inserting pseudo-elements into the circuit [18]. Subsequently, this ODE system can be solved iteratively using a numerical integration method (e.g., backward Euler). At each discrete time point, a set of nonlinear equations is solved using the NR method.

The efficiency of solving the ODE system is greatly influenced by three major factors: time-step control [28–30], initial parameter selection [31–33], and the positioning of pseudo element embeddings. There has been extensive prior research on time-step control. Traditional PTA algorithms, even those used in commercial EDA tools [34], utilize a simple iterative counting method to determine the time-step size [35]. X. Wu et al. [36] proposed an adaptive time-step control method based on switched evolution/relaxation (SER), which is a heuristic method that utilizes domain experience. Z. Jin et al. [33, 37–39] utilized advanced reinforcement learning and deep learning techniques to intelligently select the optimal time step to accelerate the solution of the ODE system. Regarding the initial parameter selection, W. Xing et al. [31] introduced Bayesian Optimization Accelerated PTA (BoA-PTA), which accelerates convergence by considering the relationship between PTA hyperparameters and the total number of NR iterations. J. Sun et al. [32] used Nerual ODE to model the solution curve of the PTA solution process and derived the gradient of the PTA hyperparameters to optimize the hyperparameters. However, there have been few studies on the optimal selection of pseudo element embedding artifacts. Unfortunately, for different circuits, the desired pseudo element embedding locations may vary significantly. The problem of selecting the optimal pseudo element embedding positions to minimize the total number of NR iterations (the main metric for evaluating PTA performance) for any given circuit does not have a clear answer.

To this end, we propose an adaptive pseudo element embedding positions selection method based on circuit topology, called GPTA. Specifically, different from the common way of constructing a graph, we linearize the nonlinear devices in the circuit so that the multiport devices become two-port devices to simplify the structure

ASPDAC '25, January 20–23, 2025, Tokyo, Japan

ACM ISBN 979-8-4007-0635-6/25/01
https://doi.org/10.1145/3658617.3703149

of the graph. Then we perform feature extraction on the circuit-constructed graph by GNN, which in turn intelligently predicts the optimal embedding locations of pseudo elements. To overcome the limitations of traditional GNN in feature extraction, we introduce the EnhanceSAGE model to enhance the diversity of information through multi-head message passing and adaptive filtering of important features. In addition, to fully utilize the features learned by GNNs at different layers, we use a layer-by-layer pooling and prediction strategy to ensure effective integration of intermediate layer features. The contributions of this work are as follows:

1) To the best of our knowledge, this is the first method in DC analysis that adaptively selects pseudo element embedding positions based on circuit topology.

2) GPTA employs a graph representation method based on equivalent circuits, simplifying the graph structure while preserving the integrity of the circuit topology.

3) We propose the EnhanceSAGE graph neural network layer, which improves feature extraction capabilities. Combined with a layer-by-layer pooling and prediction Readout strategy, this approach effectively leverages intermediate layer information in GNN, enhancing model expressiveness.

4) We demonstrate significant acceleration compared to state-of-the-art DC analysis algorithms. When extended to PPTA, DPTA, CEPTA, and RPTA, GPTA achieves speedup factors of 9.0x, 3.1x, 1.6x, and 2.4x, respectively.

2 Background

2.1 PTA algorithm and Its Embedding Position

The PTA algorithm determines the DC operating point of the nonlinear circuit by solving the nonlinear system $F(x) = 0$, where x includes the node voltages and internal branch currents [32]. The algorithm introduces pseudo inductance and capacitance to transform the system into an ODE [40], and enhances the convergence and simulation efficiency by employing time-varying components and extra branches [25, 26]. This dynamic adjustment mechanism effectively suppresses the oscillations caused by pseudo-transient components, and enables the circuit to gradually approach the steady state during the solution process, thus improving the stability of the simulation.

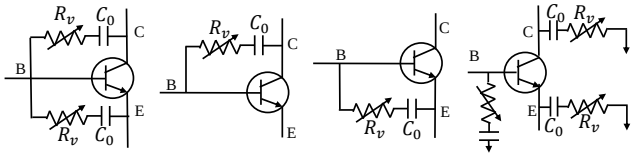

Figure 1: Embedding positions of compound pseudo capacitor for BJTs using CEPTA as an example.

In the PTA algorithm, the choice of embedding position is crucial for convergence and computational efficiency [25]. Common embedding strategies for BJTs include base-emitter (BE) embedding, base-collector (BC) embedding, base-collector and base-emitter (BC-BE) embedding, and diagonal embedding, as shown in Fig. 1. Similar strategies are also applicable for MOSFET transistors. BE embedding improves local convergence by introducing pseudo-elements between the base and emitter, while BC embedding effectively suppresses global oscillations by inserting pseudo elements between the base and collector. BC-BE embedding combines these strategies for more comprehensive convergence improvement. Diagonal embedding enhances numerical stability by introducing pseudo-elements at the diagonal positions of the Jacobian matrix. By appropriately selecting these embedding locations, the PTA algorithm achieves higher computational efficiency and simulation accuracy in circuits of varying sizes and complexities.

2.2 Graph Neural Network

A Graph Neural Network generates a vector representation (embedding) for each node in a graph, leveraging both node features and graph structure [41]. Given a graph $G = (V, E)$, where V is the set of nodes and $E \subseteq V \times V$ is the set of edges, and with node features represented by the matrix $X \in \mathbb{R}^{n \times d}$, where n is the number of nodes and d is the feature dimension, the adjacency matrix $A \in \{0, 1\}^{n \times n}$ encodes the connections between nodes. In GNNs, the embedding of a target node is updated by aggregating information from its neighbors through a message passing process. For each node v, the neighbors' features are aggregated as:

$$a_v^{(l)} = \mathrm{AGG}^{(l)}\left(h_{\mathcal{N}(v)}^{(l-1)}\right) \tag{1}$$

where $\mathcal{N}(v)$ denotes the neighbors of node v, and $h_{\mathcal{N}(v)}^{(l-1)}$ represents the features of those neighbors at layer $l-1$. The aggregated features $a_v^{(l)}$ are then used to update the target node's embedding as follows:

$$h_v^{(l)} = \mathrm{UPDATE}^{(l)}(h_v^{(l-1)}, a_v^{(l)}) \tag{2}$$

where $h_v^{(l)}$ is the updated embedding of node v at the l-th layer. After L layers of message passing, the embedding of each node encodes not only its own features but also structural information from its L-hop neighborhood within the graph.

3 Motivation

There is no fixed embedding location that gives the best performance for all circuits under all simulation conditions. As shown in Fig. 2, for each circuit netlist, the number of NR iterations varies when using four different embedding strategies. If an improper embedding location is chosen, it may lead to a significant increase in the number of iterations, which reduces the efficiency, or even a non-convergence of the solution, leading to failure. Since there is no universal optimal embedding location, it is necessary to make an adaptive selection based on the specific characteristics of the circuit to improve the convergence performance and ensure the success of the solution.

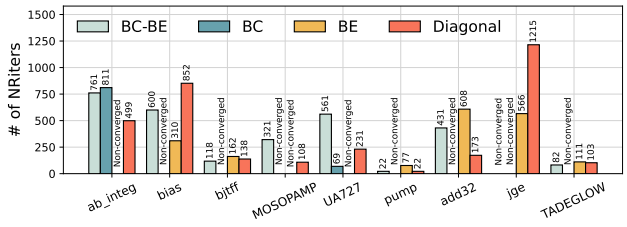

Figure 2: NR iteration comparison across embedding positions.

Machine learning provides an effective tool for adaptive selection of embedding locations. Traditionally, the selection of embedding locations often relies on the experience of experts or a

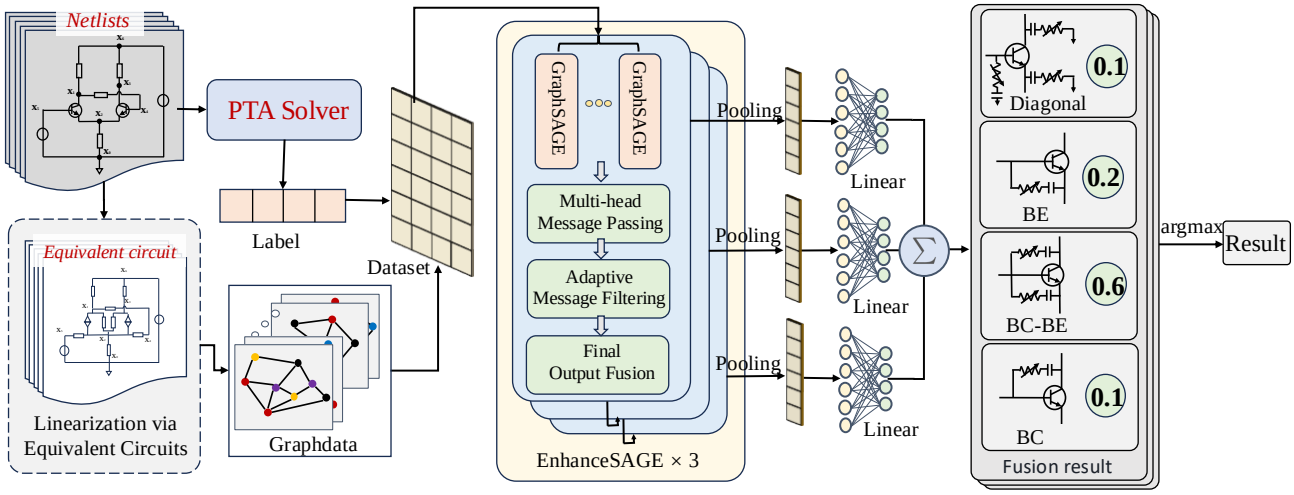

Figure 3: The overall workflow of our work.

large number of attempts, which is not only time-consuming and laborious, but also difficult to cope with the diversity of complex circuits. Graph Neural Network can effectively solve this problem by automatically extracting potential features in the circuit topology. GNN capture the topological relationships between circuit elements and adaptively select the optimal embedding locations based on the structural characteristics of the circuit, thus improving the efficiency and accuracy of DC analysis. This method does not need to rely on human experience, and can realize intelligent decision-making of embedding positions in different types of circuit netlists.

4 Proposed Method

4.1 The Overall Framework

As discussed, current methods heavily rely on expert experience, which can be difficult to generalize and transfer. In the absence of such experience, individuals often resort to repetitive trial-and-error, especially when confronted with new circuit topologies. To address these challenges, we propose the GPTA algorithm, which simplifies nonlinear circuits and utilizes a Graph Neural Network to predict optimal embedding positions. This approach eliminates the need for manual attempts, providing a more accurate and efficient solution for circuit analysis.

As shown in Fig. 3, we map the process of adaptively predicting the embedding position of PTA to a classical graph classification task. By employing a graph representation method based on equivalent circuits, we simplify the circuit structure while retaining essential information. We introduce the EnhanceSAGE layer to enhance topological feature extraction through multi-head message passing and adaptive message filtering. Combined with a layer-by-layer pooling and prediction strategy, this approach effectively aggregates features, improving the model's stability and accuracy.

4.2 Graph Representation

In DC circuit analysis, nonlinear devices such as MOSFETs and BJTs can impair the convergence of the Newton-Raphson method. Since not all ports are essential for DC analysis, extraneous connections can increase graph complexity, hinder GNN learning, and degrade model performance. To streamline the process, we linearize the circuit by converting BJTs into a two-port equivalent and transforming MOSFETs into a current-controlled source with a resistor. An example is shown in Fig 4. This approach simplifies the circuit diagram and eliminates unnecessary port interference.

We convert analog circuits into graph structures for processing with GNNs. Typical graph construction approaches either treat devices as vertices and their connections as edges or utilize both nodes and devices as vertices, with device ports as edges. Both methods have drawbacks: the first increases the edge count as the number of devices grows, resulting in a complex and inefficient graph; the second creates more challenging dichotomous graphs, which classic GNN models do not handle effectively. To address these issues, we adopt a graph construction method that represents circuit nodes as graph vertices and devices as graph edges.

To characterize each node, we assign a six-dimensional feature vector to each vertex, with each dimension indicating the number of different device types connected to the node, including resistors, capacitors, inductors, diodes, voltage sources, and current sources. Edge features are not included, as the device types are encoded within the node features. This graph construction method transforms analog circuits into a format suitable for GNN processing while preserving the circuit topology. It effectively captures the interdependencies among devices and provides the essential feature information required for GNN model training.

4.3 EnhanceSAGE

When dealing with graph data, the traditional GNN model, while performing well, in still has some limitations. Firstly, a single message passing mechanism may not be able to fully capture the diversity information among neighbouring nodes, which limits the model's ability to capture information in terms of extracting features. Secondly, the direct aggregation of neighbour node features in the standard model fails to dynamically screen out unimportant features, which may lead to excessive propagation of noisy information. In addition, the simple average or summation of aggregation ignores the effect of multi-scale information on node features, which limits the expressive ability of the model. In order to solve the above problems, we propose the EnhanceSAGE layer,

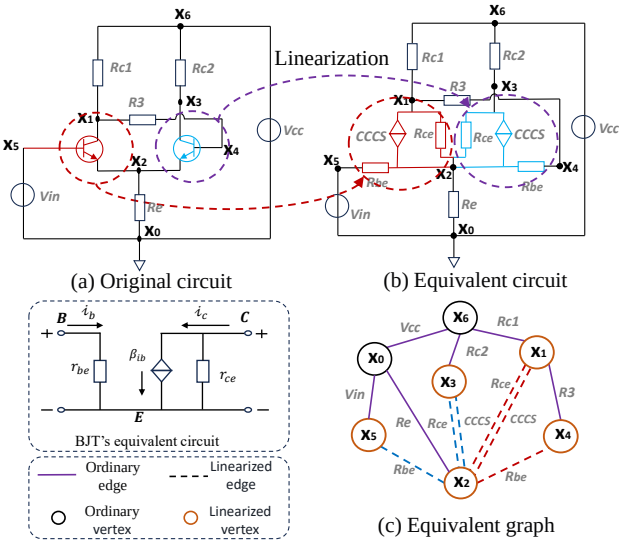

Figure 4: (a) Original circuit, (b) Linearized equivalent circuit, and (c) Graph representation with vertices for nodes and edges for devices, dashed lines indicating linearized elements. The inset shows the BJT's equivalent circuit, where ordinary vertices and edges are distinguished from linearized ones.

as shown in Fig 5. It contains three main parts: multi-head message passing, adaptive message filtering, and final output fusion.

Multi-head Message Passing allows the model to capture diverse subspace information by propagating messages through multiple independent heads, extending the receptive field and enhancing information diversity, thereby overcoming the limitation of relying on a single message-passing mechanism.

$$h'_{i,k} = \text{SAGEConv}_k\left(h_i^{(l)}, \{h_j^{(l)} | j \in N(i)\}\right) \tag{3}$$

Here, $h'_{i,k}$ denotes the output feature from the kth head, and SAGEConv$_k$ represents the standard GraphSAGE aggregation operation.

Following this, **Adaptive Message Filtering** is employed to selectively retain essential information by filtering the output features of each head. This is achieved through a gating mechanism, where a sigmoid function σ is applied to produce the filtered output $g'_{i,k}$, expressed as:

$$g'_{i,k} = \sigma\left(\text{gate}\left(h'_{i,k}\right)\right) \odot h'_{i,k} \tag{4}$$

In this equation, σ denotes the sigmoid function, and $\odot$ represents element-wise Hadamard product. This step dynamically adjusts the contribution of each head's output, thereby optimizing the effectiveness of information propagation.

Finally, the **Final Output Fusion** integrates the adaptively filtered multi-head outputs and applies a linear transformation through a fully connected layer. The filtered outputs $g'_{i,k}$ from all heads are aggregated into a final node representation $h_i^{(l+1)}$ by computing a weighted sum using learnable multi-scale weights α_k:

$$h_i^{(l+1)} = \text{FC}\left(\sum_{k=1}^{K} \alpha_k \cdot g'_{i,k}\right) \tag{5}$$

Here, K represents the number of heads, and α_k denotes the learnable weights that ensure multi-scale information is considered during the fusion process. The fully connected layer FC performs a linear transformation to extract high-level features, enhancing the model's representational power for subsequent tasks.

By integrating these steps, the EnhanceSAGE layer not only retains the efficient aggregation properties of the standard GraphSAGE model, but also introduces a multi-head mechanism, adaptive information filtering and multi-scale information fusion. This design greatly improves the model's ability to extract topological features from the graph structure.

4.4 Layer-by-Layer Pooling and Prediction

Classical GNN models, such as graph convolutional networks (GCN), graph attention networks (GAT), and GraphSAGE, are typically constructed by stacking multiple GNN layers. In the final layer, the model pools vertex features to obtain graph features and subsequently employs a multilayer perceptron (MLP) for classification. However, this reading mechanism overly relies on the output of the last layer and neglects the multilevel features extracted from the intermediate layers, thereby limiting the full utilization of graph features. To address this issue, we use a layer-by-layer pooling and layer-by-layer prediction strategy: pooling node features immediately after each GNN layer, predicting scores for four classes through linear layers, and ultimately summing the scores from each layer to produce the final classification result. This approach not only reduces the information loss but also fully leverages the graph feature representation at each layer, thereby enhancing the model's classification performance.

5 Experiments

5.1 Experimental Setup

Hardware and Software Configuration. All experiments are conducted on an AMD Ryzen 9 4900H CPU and an NVIDIA GeForce RTX 2060 GPU, running Windows 10.

Dataset. Our dataset consists of 67 circuit netlists sourced from benchmarks [42], challenging test cases [31, 37], and complex real-world circuits [38], representing a diverse range of circuit topologies. To assess the generalization capability of our model, we employ five-fold cross-validation. Optimal pseudo element embedding positions are determined using a SPICE-like simulator. For each circuit, we evaluate four PTA embedding strategies—BE, BC, BE-BC, diagonal and select the one with the fewest Newton-Raphson iterations as the ground truth label.

Baseline. Our comparison is divided into three parts. First, we compare the number of iterations of the four PTA algorithms (PPTA, DPTA, CEPTA, and RPTA) under the settings of GPTA predicted embedding positions and default embedding positions. Second, we compare the performance of the EnhanceSAGE layer with the baseline GNN layer implementation provided in [43]. Finally, we conduct an ablation study of EnhanceSAGE to explore the impact of different key mechanisms on model performance.

5.2 Simulation Performance Comparison

We compare the performance of GPTA with CEPTA across multiple test circuits, as shown in Figure 6. GPTA offers significant optimization compared to the traditional CEPTA algorithm. We then

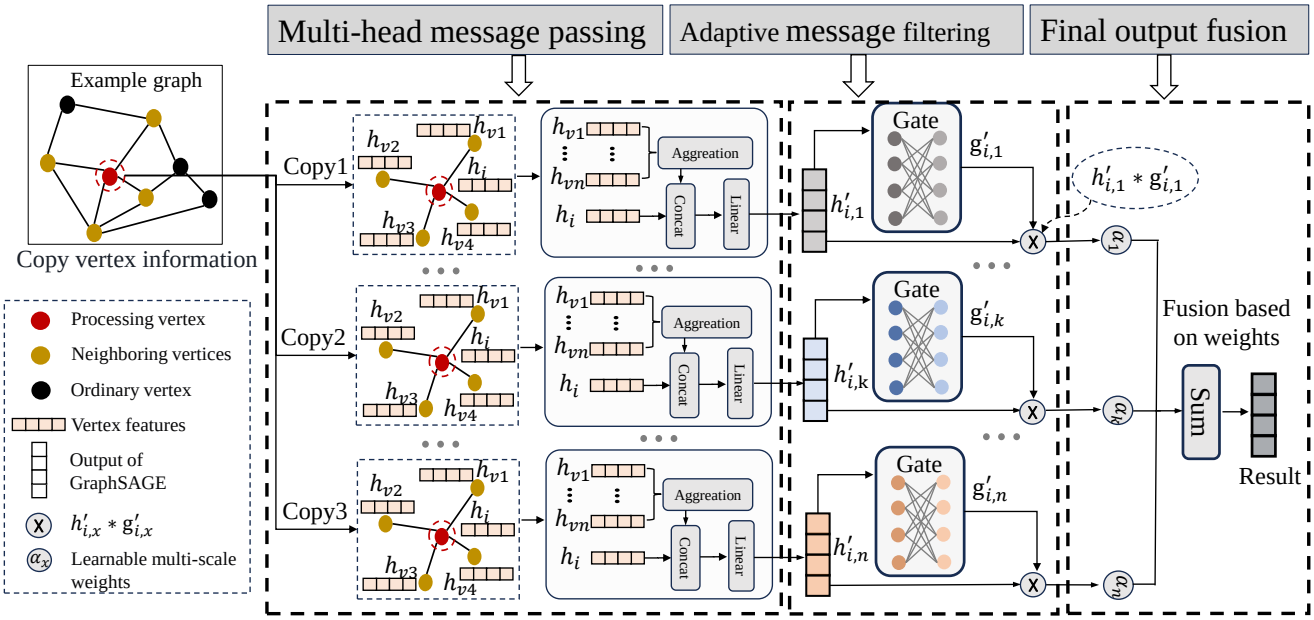

Figure 5: The diagram illustrates the node updating process in the EnhanceSAGE layer. First, the input graph is processed in parallel through multiple heads, where neighboring and central node features are aggregated. Next, these features are linearly transformed and filtered using a gating mechanism. Finally, the gated outputs are combined through a fusion of learnable weights, resulting in updated node features.

extend GPTA to other PTA algorithms and demonstrate some of the optimization effects on various circuits. As shown in Table 1, GPTA achieves average speedups of 9.0x, 3.1x, 1.6x, and 2.4x for PPTA, DPTA, CEPTA, and RPTA, respectively.

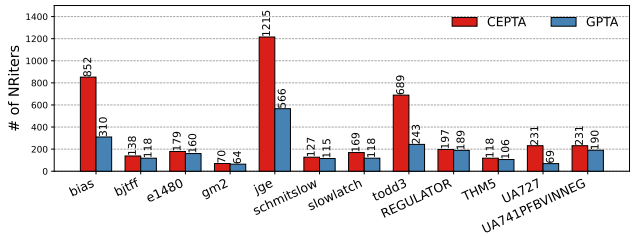

Figure 6: Simulation performance comparison: CEPTA vs GPTA.

Under the PPTA, the speedups for "mux8" and "nagle" reach 62.2x and 20.0x, respectively. This indicates that improper embedding strategies can severely impact the solving speed of certain circuits, and GPTA effectively mitigates this problem. The DPTA and CEPTA strategies also show significant acceleration, although in some cases the speedup is only 1.0x since the default embedding strategies for those circuits are already close to optimal, which GPTA successfully predicts. GPTA not only reduces the number of iterations but also improves the convergence of the circuits. Under the RPTA strategy, most non-converging circuits are able to achieve convergence with the help of GPTA.

These results demonstrate GPTA's advantages in optimizing embedding locations, significantly improving the convergence and simulation efficiency of the circuits.

5.3 GNN Model Performance Comparison

In this study, we reformulate the PTA embedding strategy optimization task as a graph classification problem. To comprehensively evaluate the performance of our proposed EnhanceSAGE model against established GNN architectures including Graph Convolutional Networks (GCN) [44], Graph Attention Networks (GAT) [45], and GraphSAGE [46], we employ four standard classification metrics: Accuracy, F1-Score, Precision, and Recall.

EnhanceSAGE consistently outperforms the baseline models across all evaluated metrics, as shown in Table 2. This demonstrates that EnhanceSAGE provides a holistic enhancement in model performance rather than excelling in isolated aspects. The improvements can be attributed to the architectural innovations embedded within EnhanceSAGE. Specifically, the integration of multi-head message passing and multi-scale information fusion allows the model to capture diverse feature representations from various subspaces, enriching its understanding of complex circuit topologies. Furthermore, the adaptive message filtering mechanism ensures that only the most relevant information is retained during message aggregation, enhancing the model's focus on task-specific features and reducing noise propagation.

Despite these advancements, there is room for further performance improvements in EnhanceSAGE. The primary limitations include the relatively limited dataset size, comprising 67 distinct circuit netlists, which may constrain the model's ability to generalize across a wider variety of circuit topologies. Expanding the dataset with more samples and diverse configurations could enhance robustness and performance. Additionally, the hyperparameter tuning process is not exhaustive due to computational and

Table 1: Simulation performance comparison under PPTA, DPTA, CEPTA and RPTA. "—"denotes non-convergence.

Circuit	of NR_iters											
	PPTA			DPTA			CEPTA			RPTA		
	native	our	speedup	native	our	speedup	native	our	speedup	native	our	speedup
latch	153	79	1.9x	108	136	0.8x	91	80	1.1x	—	105	—
astabl	108	51	2.1x	81	81	1.0x	55	55	1.0x	112	79	1.4x
bjtinv	125	125	1.0x	155	133	1.2x	186	85	2.2x	—	143	—
gm6	—	94	—	110	110	1.0x	63	60	1.1x	101	101	1.0x
hussamp	—	248	—	209	209	1.0x	91	91	1.0x	—	209	—
latch	153	79	2.0x	108	136	0.8x	91	80	1.1x	—	105	—
mux8	8579	138	62.2x	156	156	1.0x	122	122	1.0x	164	111	1.5x
nagle	2440	122	20.0x	2093	138	15.2x	306	378	0.8x	—	138	—
rca	76	54	1.4x	104	65	1.6x	82	183	0.4x	173	71	2.4x
6stageLimAmp	69	48	1.4x	135	40	3.4x	72	39	1.8x	—	39	—
D1	208	164	1.3x	213	170	1.3x	163	189	0.9x	—	214	—
D2	69	67	—	90	70	1.3x	90	80	1.1x	—	68	—
D21	—	68	—	80	80	1.0x	78	68	1.1x	—	61	—
HVREF	—	57	—	96	62	1.5x	77	57	1.3x	—	61	—
TRISTABLE	56	51	1.1x	82	82	1.0x	45	45	1.0x	61	61	1.0x
UA709	311	311	—	2985	2985	1.0x	407	79	5.2x	—	83	—
UA733	100	44	2.3x	141	50	2.8x	121	46	2.6x	202	47	4.3x
UA741	399	143	2.8x	519	128	4.1x	320	285	1.1x	—	108	—
voter25	—	163	—	192	192	1.0x	133	133	1.0x	294	—	—
Average			9.0x			3.1x			1.6x			2.4x

time constraints. Employing more systematic hyperparameter optimization techniques could further refine the model's effectiveness. Lastly, the increased complexity from incorporating multi-head and multi-scale mechanisms may introduce challenges such as overfitting, especially with limited training data. Future work could explore advanced regularization techniques or more sophisticated training strategies to balance model complexity with generalization capabilities.

Table 2: Comparisons across various GNN models.

Model	Accuracy	F1-Score	Precision	Recall
GCN [44]	0.79	0.72	0.75	0.73
GAT [45]	0.77	0.68	0.75	0.68
GraphSAGE [46]	0.79	0.71	0.78	0.78
EnhanceSAGE (ours)	0.86	0.79	0.85	0.80

5.4 Ablation Experiments

To further validate the impact of each key mechanism in our proposed EnhanceSAGE network layer on the model performance, we conduct ablation experiments and the results are shown in Table 3. The contribution of these components to the model classification performance is evaluated by removing the multi-head message propagation, adaptive message filtering, and multi-scale weighted aggregation mechanisms in EnhanceSAGE, respectively. The evaluation metrics of the experiments still use Accuracy, F1-Score, Precision, and Recall, and the results are compared with the full EnhanceSAGE model.

The ablation study reveals that each component in the EnhanceSAGE layer is essential for model performance. Removing the multi-head mechanism results in the greatest drop in accuracy and F1-Score, underscoring its role in capturing complex node relationships across multiple feature spaces. The absence of adaptive message filtering also leads to a notable performance decrease, though its impact is smaller relative to other components. These findings highlight multi-head message passing and message filtering as the most critical features for EnhanceSAGE's classification effectiveness. Removing multi-scale weighted aggregation leads to a moderate decline.

Table 3: Comparison of classification metrics after removing individual EnhanceSAGE components.

Model	Accuracy	F1-Score	Precision	Recall
No MultiHead	0.78	0.67	0.76	0.66
No MultiScaleWeight	0.83	0.73	0.81	0.75
No Message Filter	0.81	0.70	0.78	0.71
EnhanceSAGE	0.86	0.79	0.85	0.80

6 Conclusion

In this paper, we propose a novel graph neural network enhanced PTA method, GPTA, for adaptively selecting pseudo element embedding positions in PTA based on circuit topology. By incorporating the EnhanceSAGE model, we improve feature extraction capabilities and optimized the embedding position selection process. Evaluations on benchmark and practical circuits demonstrate that GPTA outperforms traditional PPTA, DPTA, CEPTA, and RPTA methods by achieving speedups of 9.0×, 3.1×, 1.6×, and 2.4×, respectively. Furthermore, GPTA is orthogonal to step size control and initial value prediction methods, allowing these approaches to be used together to further enhance performance.

Acknowledgments

This work is supported in part by the National Key R&D Program of China (Grant No.2022YFB4400400), NSFC (Grant No.62204265, 92473107, 62234010, U23A20301) and Science Foundation of CUPB (No.2462024BJRC002). Zhou Jin is the corresponding author of this paper.

References

[1] J. Deng, K. Batselier, Y. Zhang, and N. Wong. An efficient two-level dc operating points finder for transistor circuits. In *Design Automation Conference (DAC)*, 2014.

[2] C. W. Ho, A. Ruehli, and P. Brennan. The modified nodal approach to network analysis. *IEEE Transactions on Circuits and Systems (TCAS)*, 1975.

[3] C. Li, B. Zhang, Y. Duan, Y. Li, Z. Ye, W. Liu, D. Tao, and Z. Jin. Masc: A memory-efficient adjoint sensitivity analysis through compression using novel spatiotemporal prediction. In *Design Automation Conference (DAC)*, 2024.

[4] Y. Zhao, X. Yang, Y. Bai, L. Zeng, D. Niu, W. Liu, and Z. Jin. Csp: Comprehensively-sparsified preconditioner for efficient nonlinear circuit simulation. In *International Conference on Computer Aided Design (ICCAD)*, 2024.

[5] G. Feng, H. Wang, Z. Guo, M. Li, T. Zhao, Z. Jin, W. Jia, G. Tan, and N. Sun. Accelerating large-scale sparse lu factorization for rf circuit simulation. In *European Conference on Parallel Processing (Euro-Par)*, 2024.

[6] Z. Jin, W. Li, Y. Bai, T. Wang, Y. Lu, and W. Liu. Machine learning and gpu accelerated sparse linear solvers for transistor-level circuit simulation: A perspective survey. In *Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2024.

[7] T. Wang, W. Li, H. Pei, Y. Sun, Z. Jin, and W. Liu. Accelerating sparse lu factorization with density-aware adaptive matrix multiplication for circuit simulation. In *Design Automation Conference (DAC)*, 2023.

[8] J. Zhao, Y. Wen, Y. Luo, Z. Jin, W. Liu, and Z. Zhou. Sflu: Synchronization-free sparse lu factorization for fast circuit simulation on gpus. In *Design Automation Conference (DAC)*, 2021.

[9] Y. Chen, H. Pei, X. Dong, Z. Jin, and C. Zhuo. Application of deep learning in back-end simulation: Challenges and opportunities. In *Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2022.

[10] Z. Jin, H. Pei, Y. Dong, X. Jin, X. Wu, W. W. Xing, and D. Niu. Accelerating nonlinear dc circuit simulation with reinforcement learning. In *Design Automation Conference (DAC)*, 2022.

[11] Y. L. Kuo and M. L. Liou. Computer-aided analysis of electronic circuits: Algorithms and computational techniques. *Proceedings of the IEEE*, 1977.

[12] K. S. Kundert and P. Gray. *The Designer's Guide to Spice and Spectre.* 1995.

[13] T. M. Najibi. Continuation methods as applied to circuit simulation. *IEEE Circuits and Devices Magazine*, 1989.

[14] C. E. Lemke. Pathways to solutions, fixed points, and equilibria. *SIAM Review*, 1984.

[15] Z. Jin, T. Feng, X. Wu, D. Niu, Z. Zhou, and C. Zhuo. Msh: A multi-stage hiz-aware homotopy framework for nonlinear dc analysis. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2024.

[16] E. Yilmaz and M. M. Green. Some standard spice dc algorithms revisited: why does spice still not converge? In *International Symposium on Circuits and Systems (ISCAS)*, 1999.

[17] Z. Jin, T. Feng, Y. Duan, X. Wu, M. Cheng, Z. Zhou, and W. Liu. Palbbd: A parallel arclength method using bordered block diagonal form for dc analysis. In *Great Lakes Symposium on VLSI (GLSVLSI)*, 2021.

[18] C. T. Kelley and D. E. Keyes. Convergence analysis of pseudo-transient continuation. *SIAM Journal on Numerical Analysis*, 1998.

[19] W. Weeks, A. Jimenez, G. Mahoney, D. Mehta, H. Qassemzadeh, and T. Scott. Algorithms for astap–a network-analysis program. *IEEE Transactions on Circuit Theory*, 1973.

[20] L. Goldgeisser, E. Christen, M. Vlach, and J. Langenwalter. Open ended dynamic ramping simulation of multi-discipline systems. In *International Symposium on Circuits and Systems (ISCAS)*, 2001.

[21] X. Wu, Z. Jin, and Y. Inoue. Numerical integration algorithms with artificial damping for the pta method applied to dc analysis of nonlinear circuits. In *International Conference on Communications, Circuits, and Systems (ICCCAS)*, 2013.

[22] X. Wu, Z. Jin, D. Niu, and Y. Inoue. A pta method using numerical integration algorithms with artificial damping for solving nonlinear dc circuits. *Institute of Electronics, Information and Communication Engineers (IEICE)*, 2014.

[23] Z. Jin, X. Wu, Y. Inoue, and D. Niu. A ramping method combined with the damped pta algorithm to find the dc operating points for nonlinear circuits. In *International Symposium on Integrated Circuits (ISIC)*, 2014.

[24] Z. Jin, X. Wu, D. Niu, X. Guan, and Y. Inoue. Effective ramping algorithm and restart algorithm in the spice3 implementation for dpta method. *Nonlinear Theory and Its Applications, IEICE*, 2015.

[25] Z. Jin, X. Wu, D. Niu, and Y. Inoue. Effective implementation and embedding algorithms of cepta method for finding dc operating points. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 2013.

[26] H. Yu, Y. Inoue, K. Sako, X. Hu, and Z. Huang. An effective spice3 implementation of the compound element pseudo-transient algorithm. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 2007.

[27] Z. Jin, M. Liu, and X. Wu. An adaptive dynamic-element pta method for solving nonlinear dc operating point of transistor circuits. In *International Midwest Symposium on Circuits and Systems (MWSCAS)*, 2018.

[28] D. Niu, Y. Tao, Z. Jin, Y. Dong, C. Wang, and C. Sun. Islu: Indexing-efficient sparse lu factorization for circuit simulation on gpus. In *International Conference on Computer Aided Design (ICCAD)*, 2024.

[29] Y. Dong, D. Niu, Z. Jin, C. Zhang, C. Sun, and Z. Zhou. Ispt-net: A novel transient backward-stepping reduction policy by irregular sequential prediction transformer. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2024.

[30] Y. Dong, D. Niu, Z. Jin, C. Zhang, Q. Li, and C. Sun. Adaptive stepping pta for dc analysis based on reinforcement learning. *IEEE Transactions on Circuits and Systems II: Express Briefs (TCAS-II)*, 2022.

[31] W. W. Xing, X. Jin, T. Feng, D. Niu, W. Zhao, and Z. Jin. Boa-pta: A bayesian optimization accelerated pta solver for spice simulation. *ACM Trans. Des. Autom. Electron. Syst. (TODAES)*, 2022.

[32] J. Sun, X. Zha, C. Wang, X. Wu, D. Niu, W. Xing, and Z. Jin. Pseudo adjoint optimization: Harnessing the solution curve for spice acceleration. In *International Conference on Computer Aided Design (ICCAD)*, 2024.

[33] D. Niu, Y. Dong, Z. Jin, C. Zhang, Q. Li, and C. Sun. Ossp-pta: An online stochastic stepping policy for pta on reinforcement learning. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2023.

[34] K. S. Kundert and P. Gray. *The Designer's Guide to Spice and Spectre.* Kluwer Academic Publishers, 1995.

[35] F. N. Najm. *Circuit Simulation.* Wiley-IEEE Press, 2010.

[36] X. Wu, Z. Jin, D. Niu, and Y. Inoue. An adaptive time-step control method in damped pseudo-transient analysis for solving nonlinear dc circuit equations. *Institute of Electronics, Information and Communication Engineers (IEICE)*, 2015.

[37] Z. Jin, H. Pei, Y. Dong, X. Jin, X. Wu, W. W. Xing, and D. Niu. Accelerating nonlinear dc circuit simulation with reinforcement learning. In *Design Automation Conference (DAC)*, 2022.

[38] Y. Dong, D. Niu, Z. Jin, C. Zhang, Q. Li, and C. Sun. Adaptive stepping pta for dc analysis based on reinforcement learning. *IEEE Transactions on Circuits and Systems II: Express Briefs (TCAS-II)*, 2023.

[39] X. Zha, H. Pei, D. Niu, X. Wu, and Z. Jin. Deep learning enhanced time-step control in pseudo transient analysis for efficient nonlinear dc simulation. In *International Symposium of EDA (ISEDA)*, 2023.

[40] K. S. Kundert and P. Gray. *The Designer's Guide to Spice and Spectre.* 1995.

[41] L. Alrahis, S. Patnaik, M. Shafique, and O. Sinanoglu. Embracing graph neural networks for hardware security. In *International Conference on Computer Aided Design (ICCAD)*, 2022.

[42] J. A. Barby and R. S. Guindi. Circuitsim93: A circuit simulator benchmarking methodology case study. *Annual IEEE International Conference and Exhibit ASIC*, 1993.

[43] D. Prakash Dwivedi, C. K. Joshi, A. T. Luu, T. Laurent, Y. Bengio, and X. Bresson. Benchmarking graph neural networks. *Journal of Machine Learning Research*, 2023.

[44] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[45] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[46] W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems*, 2017.