Boosting the Performance of Transistor-Level Circuit Simulation with GNN

ASP-DAC 2025

Jiqing Jiang, Yongqiang Duan, **Zhou Jin**

Super Scientific Software Laboratory,

China University of Petroleum-Beijing, China

Email: jinzhou@cup.edu.cn

Outline

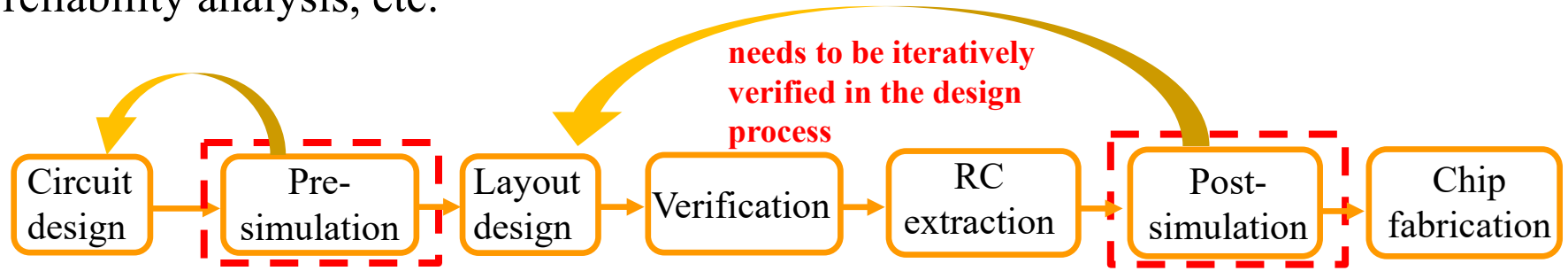
- Background and Motivation
- Proposed Method
  - GPTA Framework
  - Graph Representation
  - EnhanceSAGE
  - Layer-by-Layer Pooling and Prediction
- Experiment Results
- Conclusions and Future Work

Outline

- **Background and Motivation**
- Proposed Method
  - GPTA Framework
  - Graph Representation
  - EnhanceSAGE
  - Layer-by-Layer Pooling and Prediction
- Experiment Results
- Conclusions and Future Work

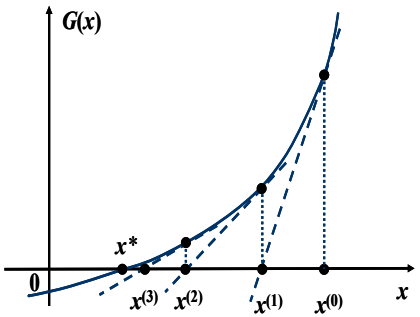
Background: Transistor-level Simulation

- **Transistor-level circuit simulation (SPICE simulation)** plays a crucial role in verifying circuit performance, and serving as the basis of timing, yield and reliability analysis, etc.

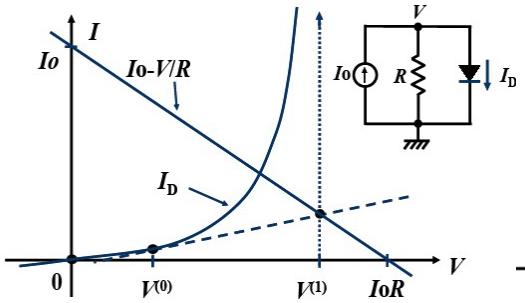


Design process of analog circuits

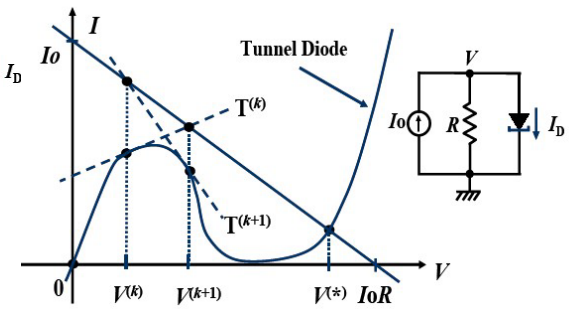
- **Nonlinear DC analysis** in SPICE is key for determining the DC operating point. Newton-Raphson (NR) is widely used, but convergence issues arise due to inaccurate initial guesses.



Convergence Situation



Overflow Situation

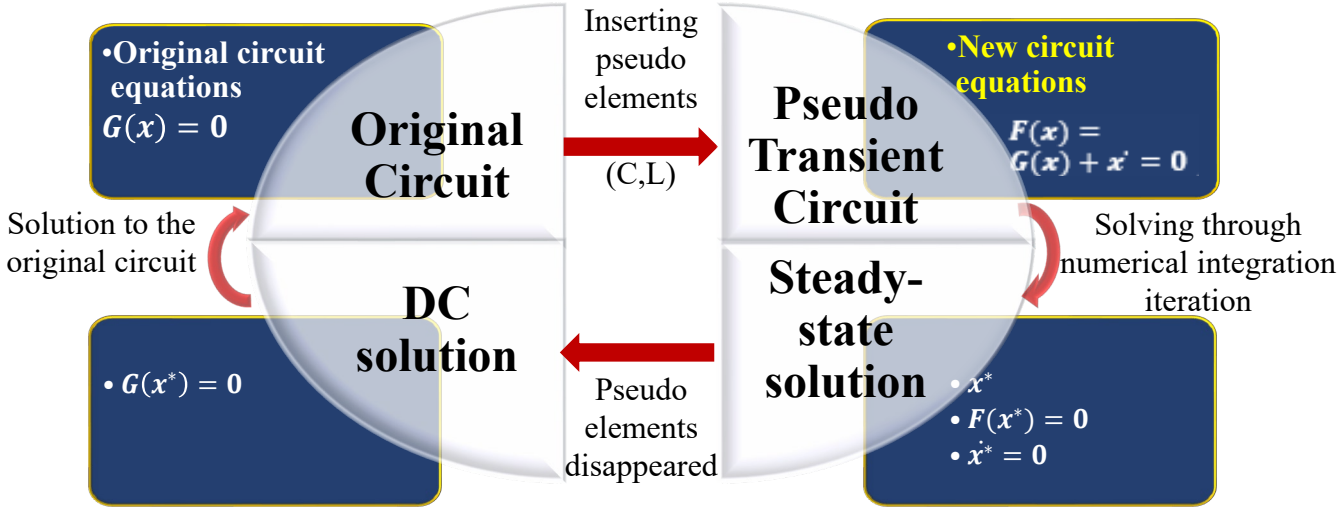


Loop Situation

$G(x) = b$
where $G(\cdot) : \mathbb{R}^m \rightarrow \mathbb{R}^m$
$\mathbf{x} = (v, i)^T \in \mathbb{R}^m$
v : node voltage
i : internal branch current

Background: PTA Algorithms

Pseudo Transient Analysis (PTA) is currently the **most powerful and promising** numerical solving algorithm in SPICE circuit simulation for DC analysis, as it is easy to implement and has **good continuity and convergence**.

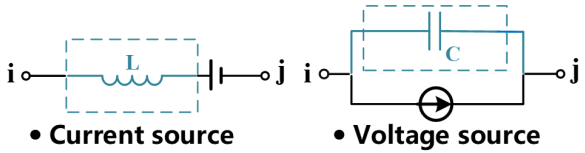


- Inserting pseudo capacitors/inductors can effectively address discontinuity issues, but it introduces oscillation problems and increases computation time.

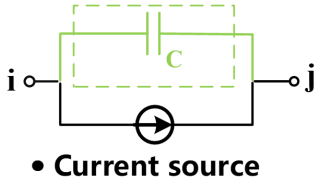
Background: PTA Algorithms

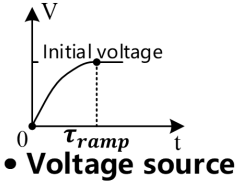
- **Pure PTA (PPTA) addresses solution discontinuities but introduces oscillations.**

PPTA[1]

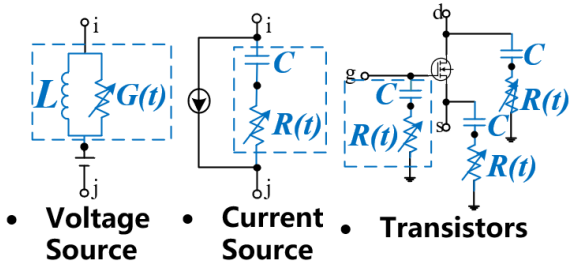


- **Variants [2][3][4] developed to overcome PTA limitations:**

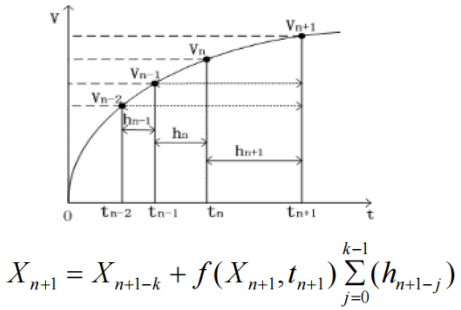




RPTA[2]



CEPTA[3]



$$X_{n+1} = X_{n+1-k} + f(X_{n+1}, t_{n+1}) \sum_{j=0}^{k-1} (h_{n+1-j})$$

DPTA[4]

[1] W. Weeks, A. Jimenez, G. Mahoney, D. Mehta, H. Qassemzadeh and T. Scott, Algorithms for ASTAP--A network-analysis program, in IEEE Transactions on Circuit Theory, 1973.

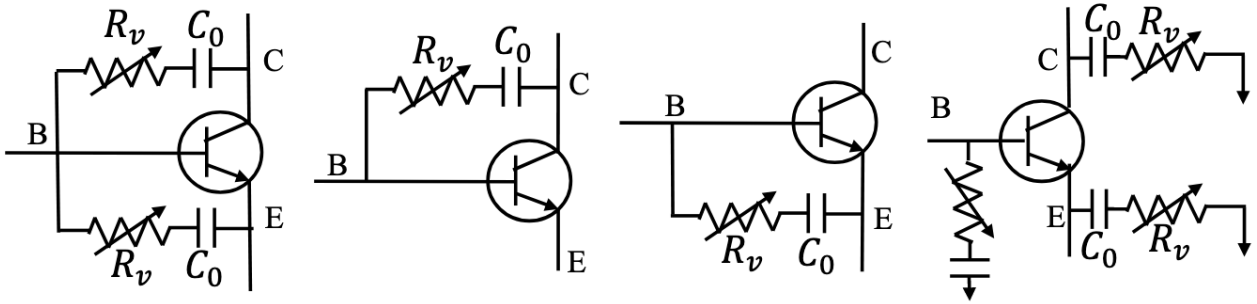
[3] Z. Jin, X. Wu, Y. Inoue, and N. Dan. A ramping method combined with the damped pta algorithm to find the dc operating points for nonlinear circuits. In ISIC, 2014.

[4] H. Yu, Y. Inoue, K. Sako, X. Hu, and Z. Huang. An effective spice3 implementation of the compound element pseudo-transient algorithm. In IEICE Trans. Fundam. Electron. Commun. Comput. Sci, 2007.

[4] Z. Jin, M. Liu and X. Wu, An Adaptive Dynamic-Element PTA Method for Solving Nonlinear DC Operating Point of Transistor. In MWSCAS, 2018.

Background: PTA Embedding Position

Beyond the PTA methods themselves, different **embedding strategies** play a critical role in influencing convergence and stability.

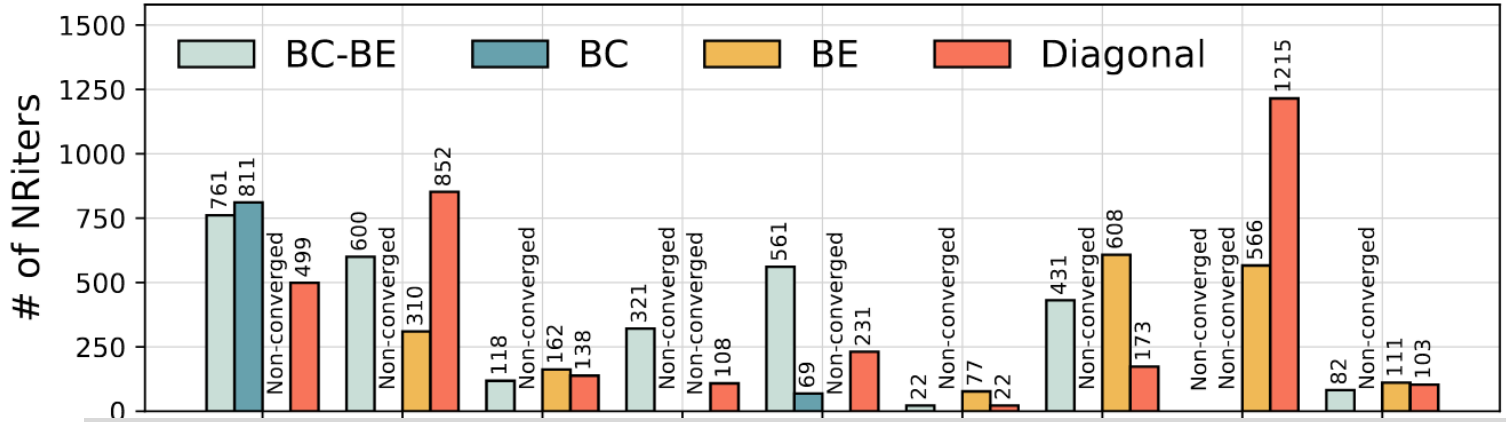


(a) BC-BE (b) BC (c) BE (d) Diagonal

- BE: insert pseudo-elements between base and emitter.
- BC: insert pseudo-elements between base and collector.
- BE-BC: insert pseudo-elements between both base-emitter and base-collector.
- Diagonal: insert pseudo-elements between node to ground.
- **BE**, **BC**, **BC-BE**, and **Diagonal** embedding methods influence the convergence and efficiency of the analysis.

Motivation

Simulation efficiency (# of NR iterations) comparison for different embedding positions. Non-convergence observed in all circuits (#NR iterations > 10,000).



Therefore, it is essential to select the most suitable embedding strategy for solving any given circuit.

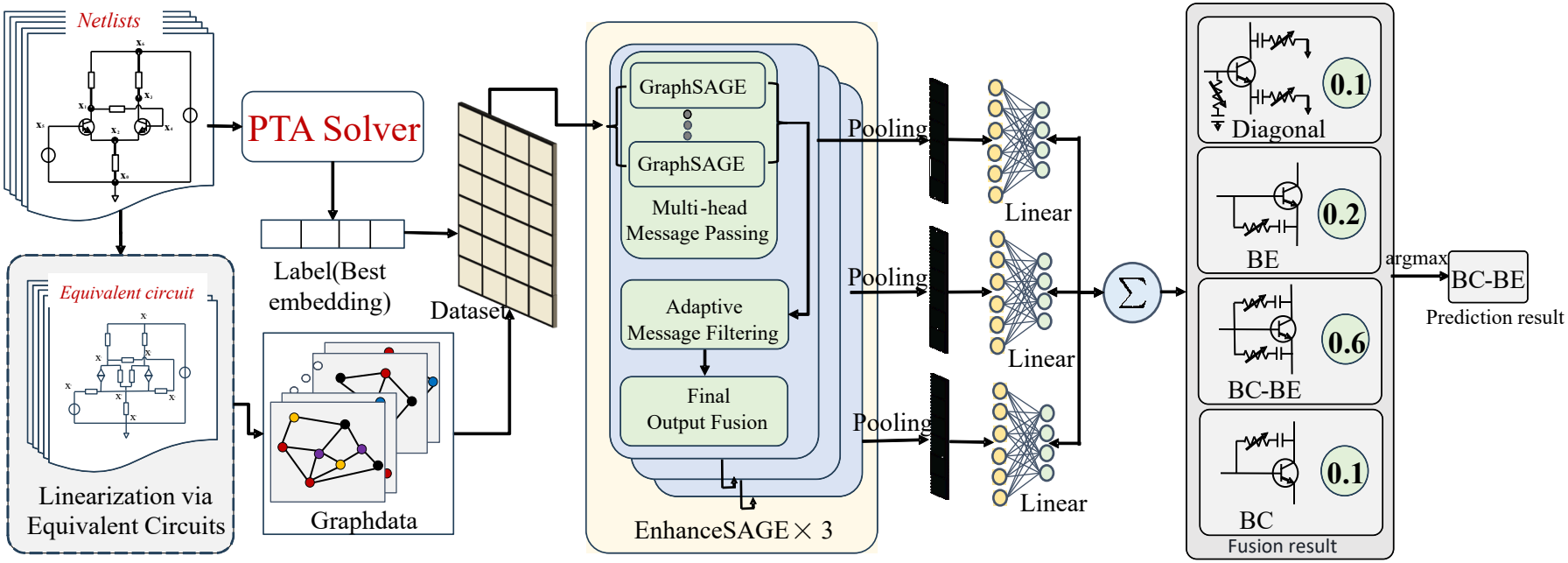
- Different embedding strategies (BC, BE, BC-BE, Diagonal) lead to varying NR iterations for each circuit.
- No single embedding position works best for all circuits.
- Improper selection can significantly increase NR iterations, reduce efficiency, even cause non-convergence.

Outline

- Background and Motivation
- **Proposed Method**
  - GPTA Framework
  - Graph Representation
  - EnhanceSAGE
  - Layer-by-Layer Pooling and Prediction
- Experiment Results
- Conclusions

GPTA Framework

We propose a new framework **GPTA**, which extracts circuit topology features using GNN and maps the embedding position selection to a classification task to **predict the optimal embedding strategy** for each circuit.

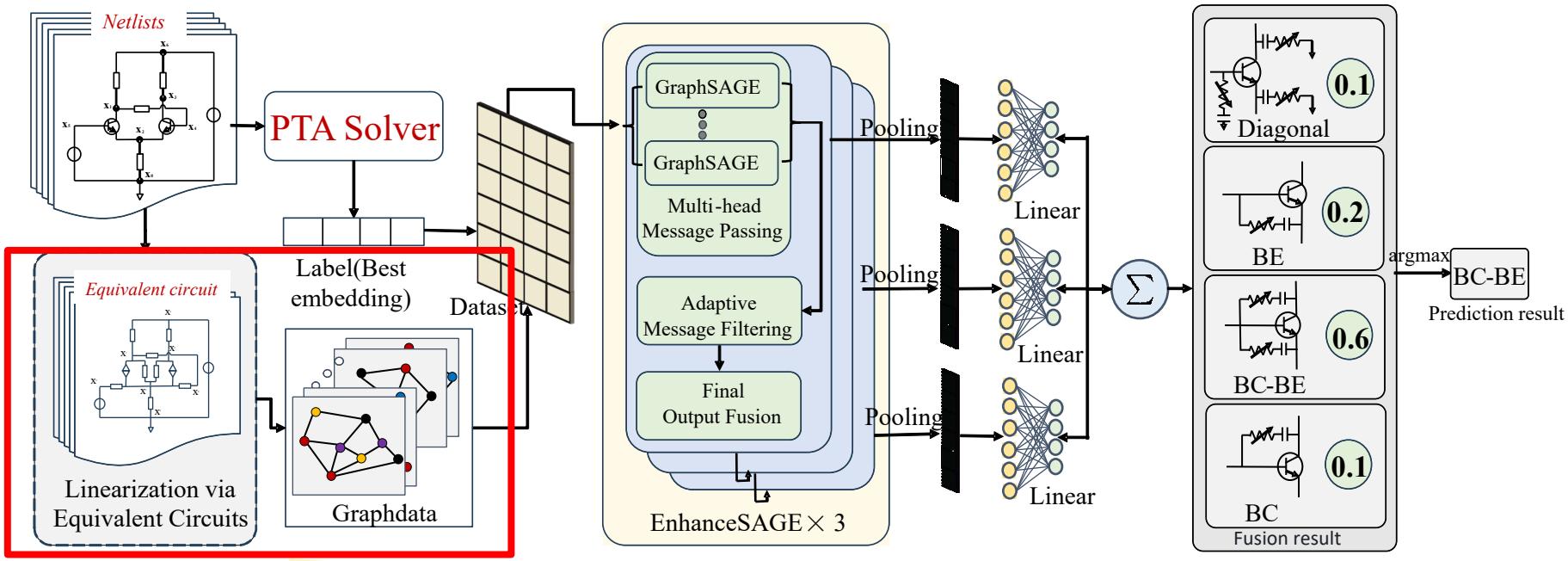


Our framework is composed of three main components:

Graph Representation, EnhanceSAGE, and Layer-by-Layer Pooling and Prediction.

Graph Representation

To make the circuit suitable for GNN processing, we first linearize it by replacing multi-port devices with their equivalent circuits and then represent it as a graph. This method simplifies the circuit while preserving key topological information.



Graph Representation:

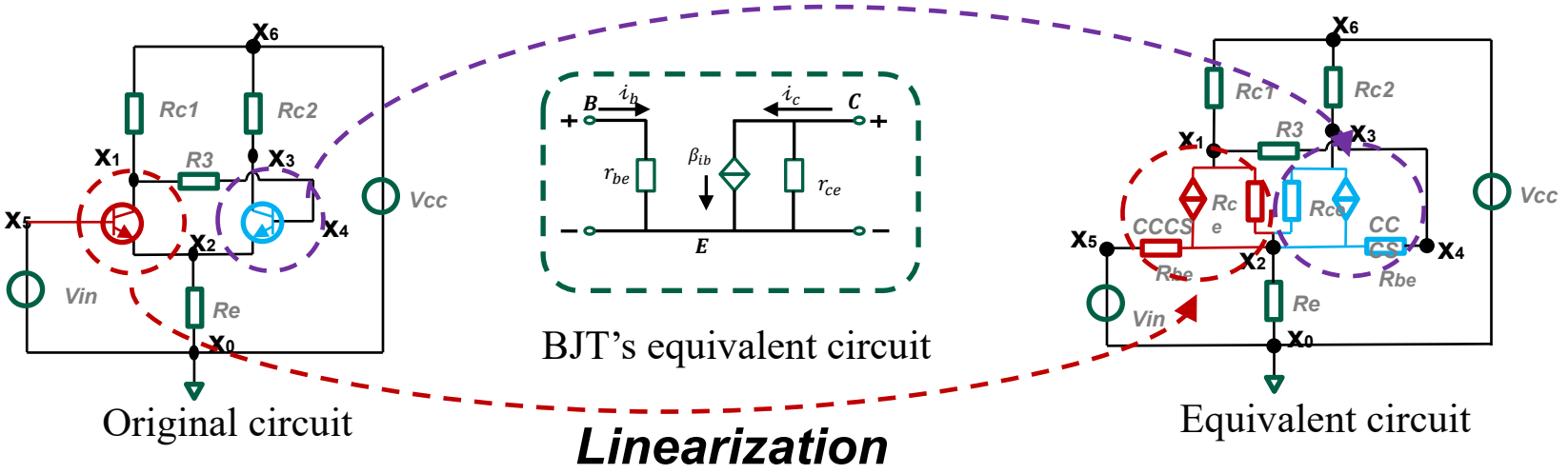
1. **Linearization**: simplify MOS and BJT into an easier-to-process equivalent.
2. **Graph construction**: converting netlist into graph.

Graph Representation: (1) Linearization

Nonlinear devices (MOSFETs, BJTs) influence Newton-Raphson convergence in DC analysis.

Not all connections between ports are essential for DC analysis. Extraneous connections:

- Increase graph complexity by adding many unnecessary connections between ports.
- Hinder GNN learning and degrade model performance.
- Do not effectively capture the behavior of the devices and the circuit.



- **BJTs**: Convert to two-port equivalent.
- **MOSFETs**: Transform into a current-controlled source with a resistor.
- Simplify the circuit, enhancing GNN performance by removing unnecessary connections.

Graph Representation: (2) Graph construction

Typical graph construction methods for circuit:

1.Devices as vertices, connections as edges.

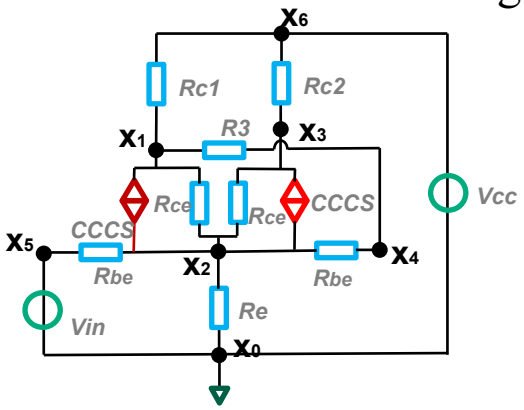
2.Nodes and devices as vertices, device ports as edges.

3.Nodes as vertices, devices as edges.

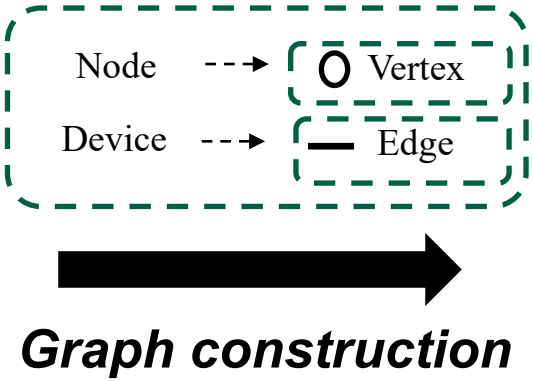
Issues:

Method 1: Increases edge count with device growth → complex, inefficient graphs.

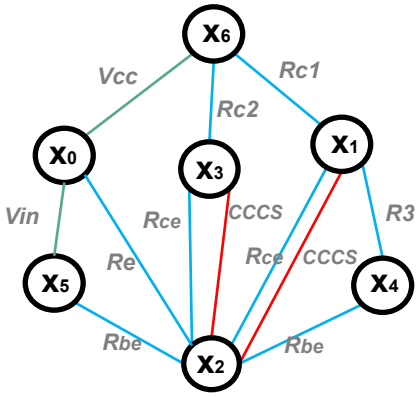
Method 2: Generates challenging bipartite graphs → poorly supported by GNNs.



Equivalent circuit



Graph construction



Graph Representation

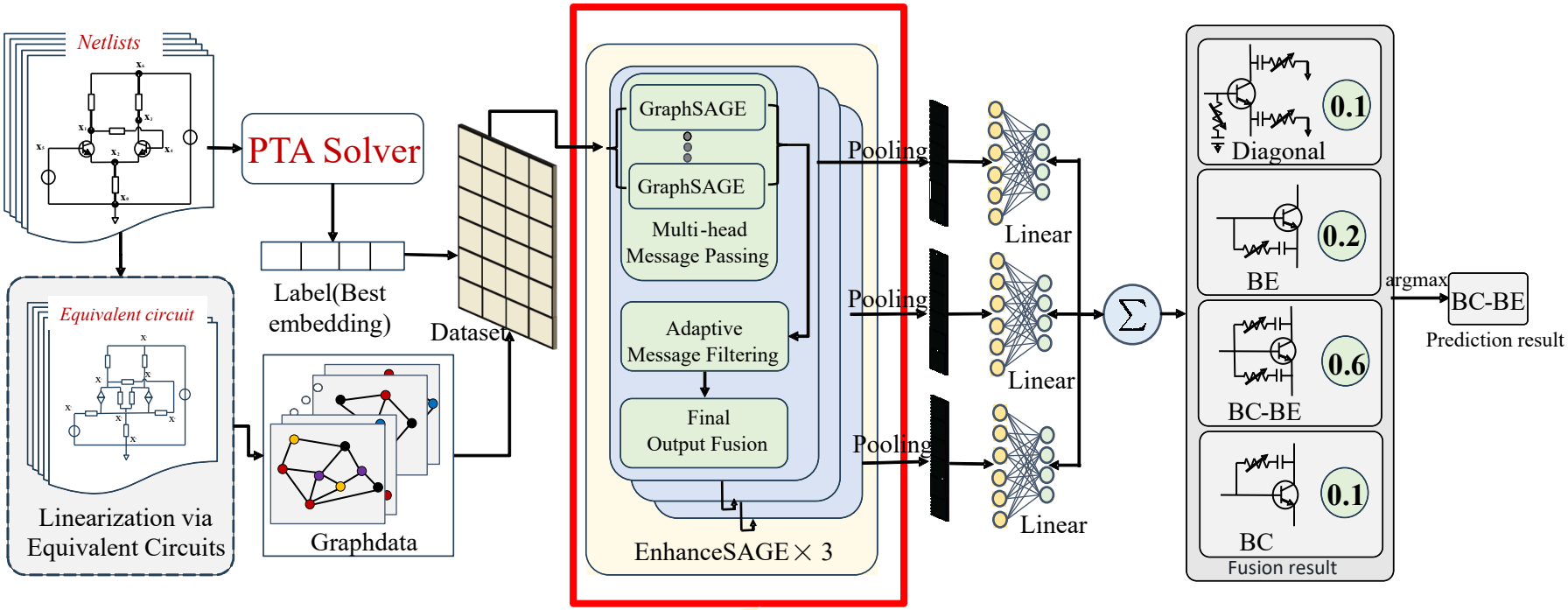
Therefore, we choose the third method:

•Circuit nodes as vertices, devices as edges.

•Nodes are assigned a 6-dimensional initial node feature: Number of connected **resistors, capacitors, inductors, diodes, voltage sources, and current sources**.

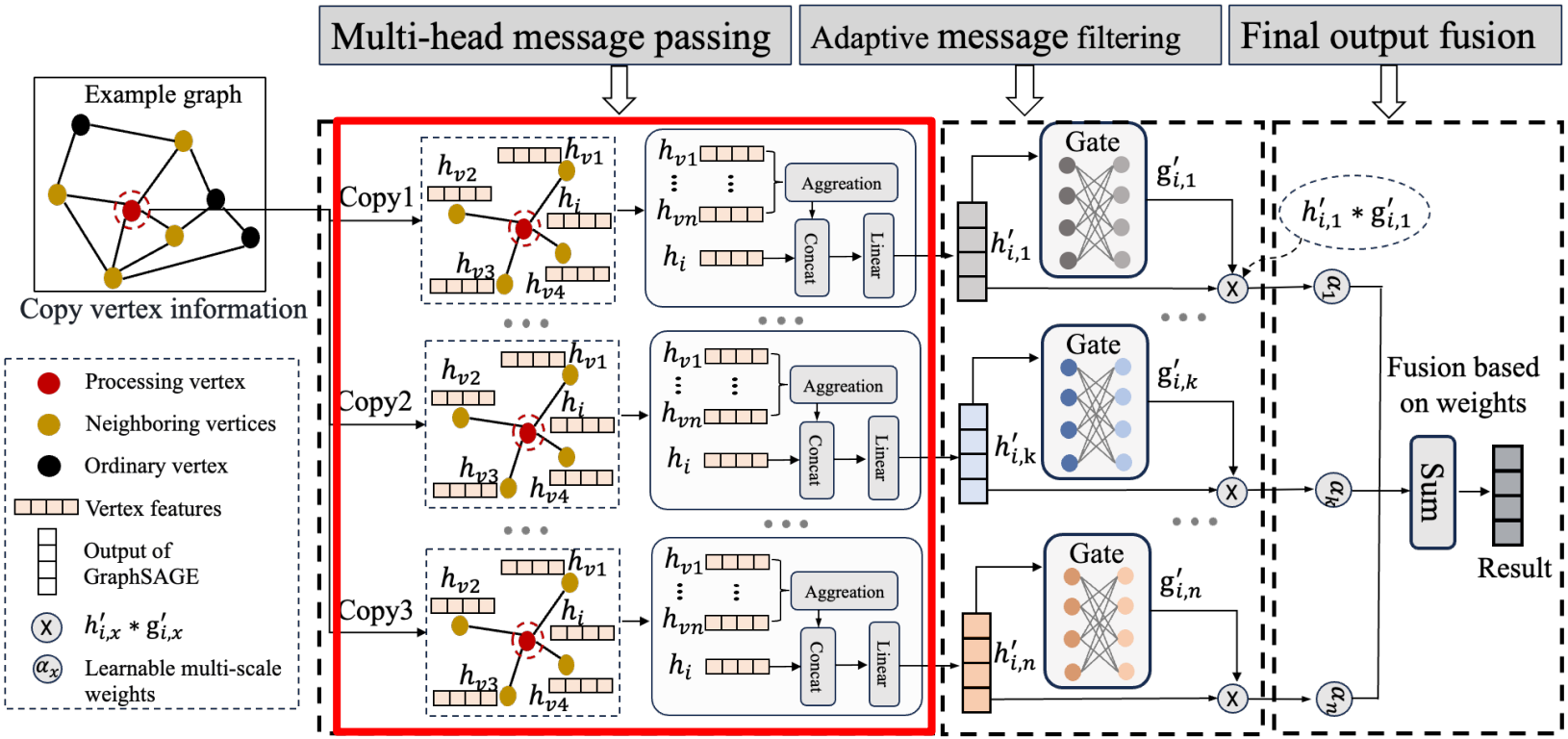
EnhanceSAGE

To address the challenges in traditional GNN models, such as uniform treatment of nodes in GCN, limited focus on local neighbors in GAT, and sampling inconsistencies in GraphSAGE, we propose EnhanceSAGE with the following key enhancements: **Multi-Head Message Passing**, **Adaptive Message Filtering**, and **Final Output Fusion**.



EnhanceSAGE: Better capture richer features

EnhanceSAGE

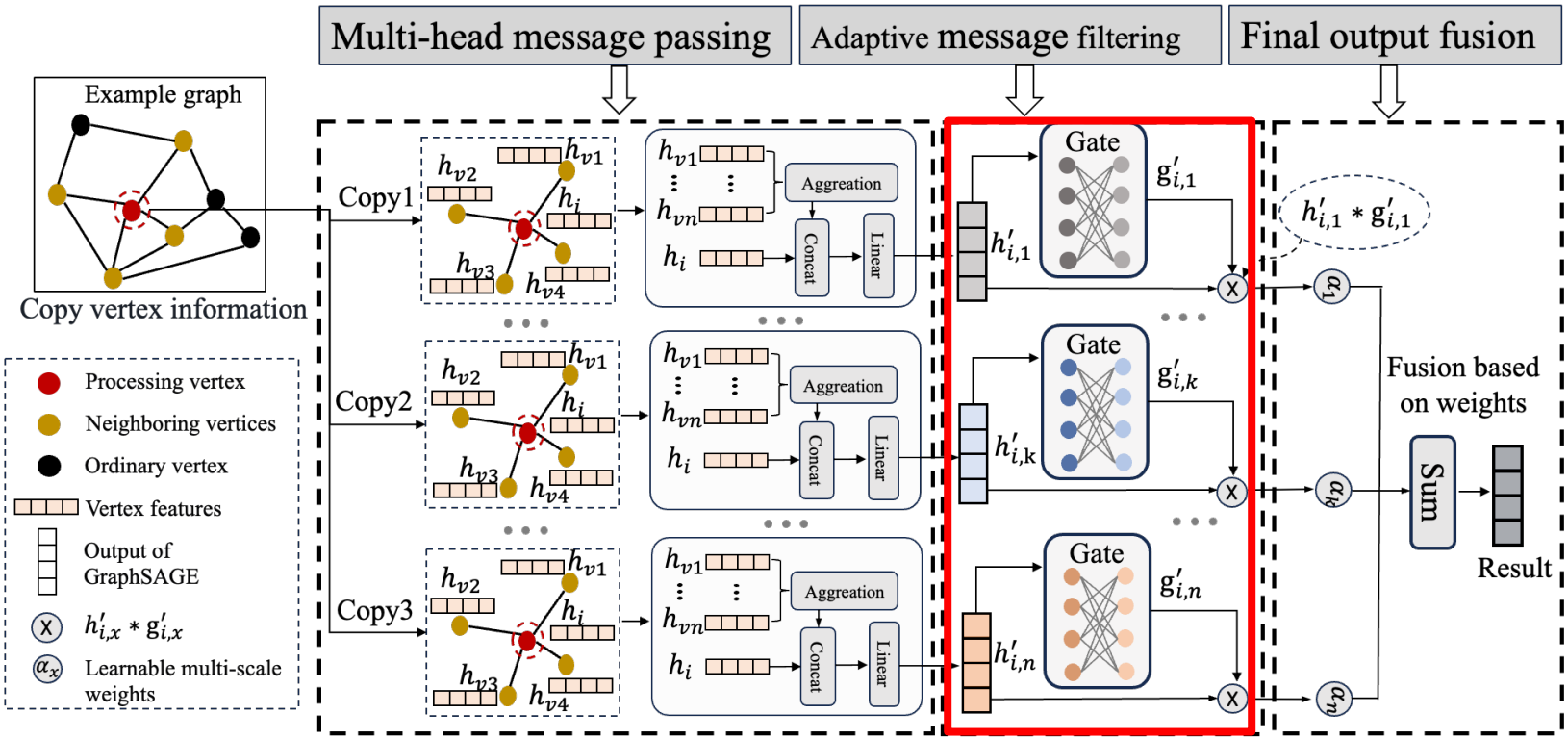


Multi-Head Message Passing: $$h'_{i,k} = \mathrm{SAGEConv}_k\left(h_i^{(l)}, \{h_j^{(l)} | j \in N(i)\}\right)$$

Objective: Enhance information diversity from neighboring nodes using multiple independent message-passing heads.

Mechanism: Each head performs a separate GraphSAGE convolution, allowing the model to learn diverse feature representations.

EnhanceSAGE

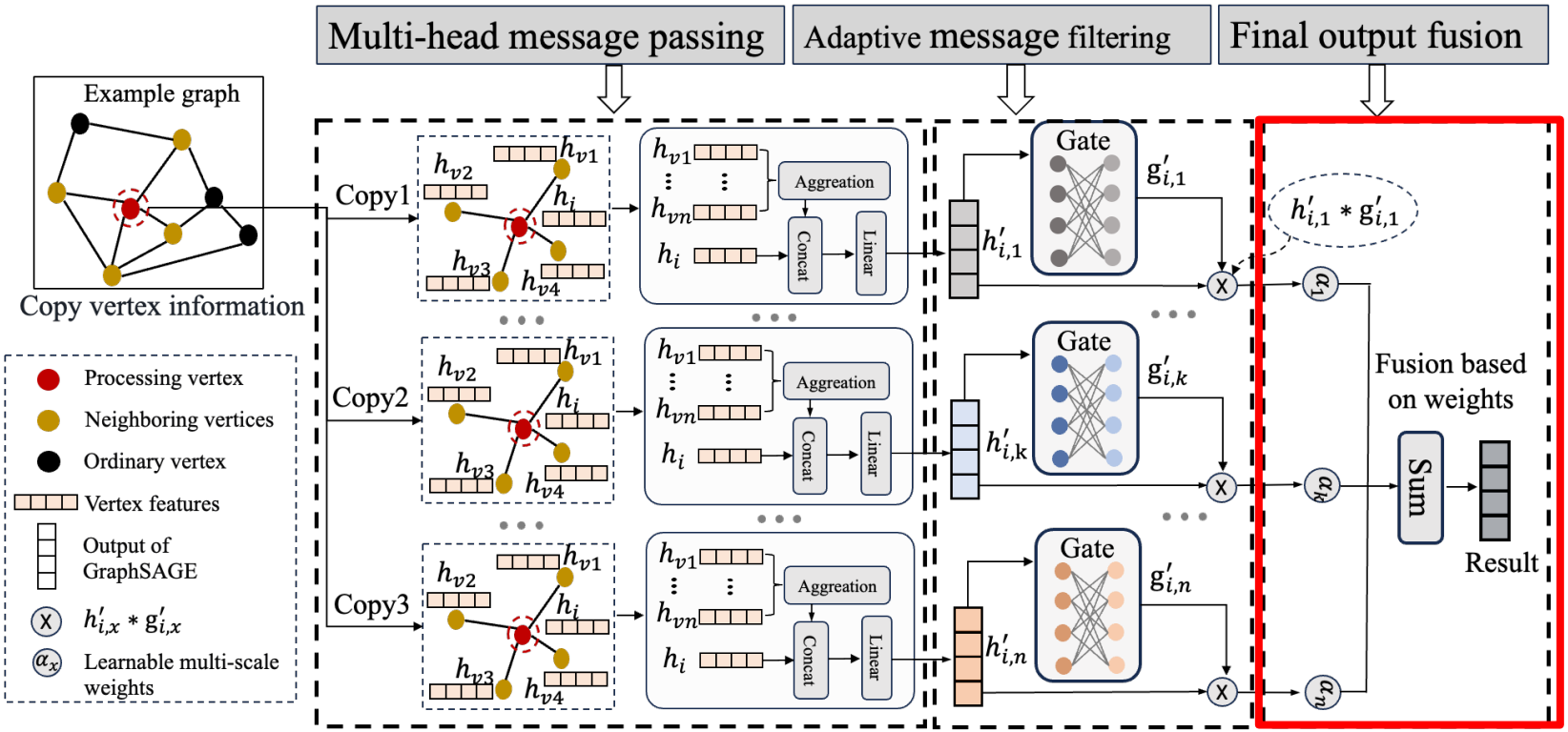


Adaptive Message Filtering: $g'_{i,k} = \sigma\left(\text{gate}\left(h'_{i,k}\right)\right) \odot h'_{i,k}$

Objective: Dynamically retain essential information and suppress noisy or irrelevant features during message aggregation.

Mechanism: Utilize a gating mechanism with a sigmoid function to filter the aggregated features from each head.

EnhanceSAGE



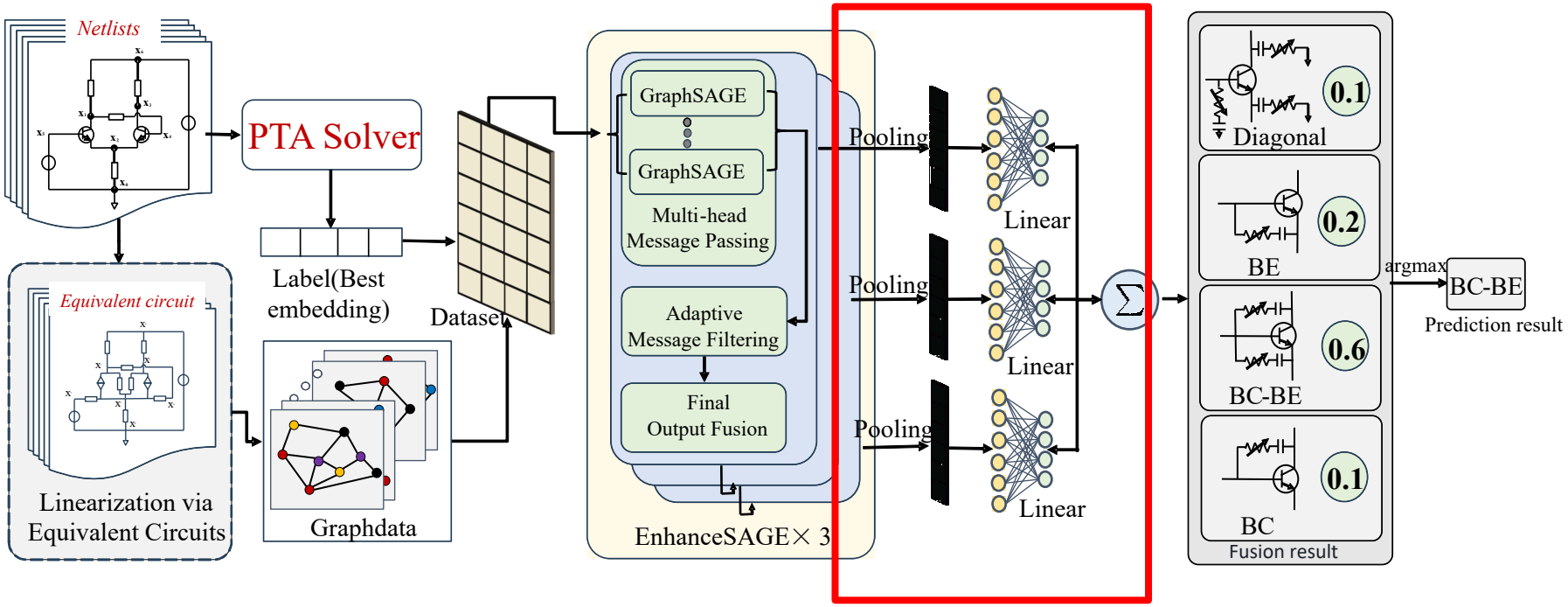
Final Output Fusion: $$h_i^{(l+1)} = \mathrm{FC}\left(\sum_{k=1}^{K} \alpha_k \cdot g'_{i,k}\right)$$

Objective: Integrate the filtered outputs from all heads and produce the final updated node representation.

Mechanism: Combine the adaptively filtered multi-head outputs using learnable multi-scale weights, followed by a fully connected layer for linear transformation.

Layer-by-Layer Pooling and Prediction

To overcome the limitations of information loss in the intermediate layers of traditional GNNs, we propose a comprehensive layer-by-layer pooling and prediction approach that fully leverages multilevel graph representations for enhanced classification.



Layer-by-Layer Pooling and Predict: Utilize intermediate features

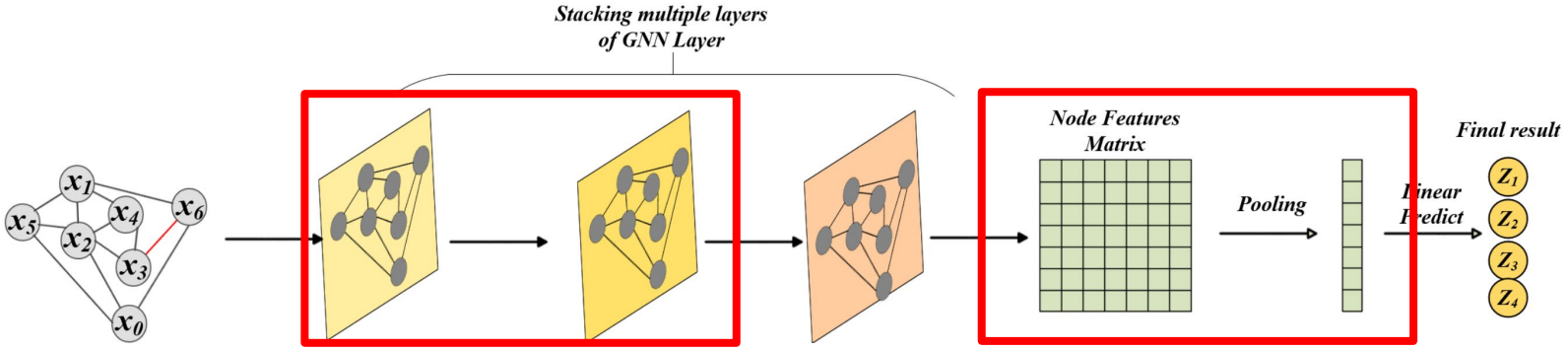
Layer-by-Layer Pooling and Prediction

Classical GNN Limitations

Traditional GNNs stack multiple layers and use pooling at the final layer to obtain graph features for downstream tasks.

Limited Feature Utilization: Intermediate layers, which capture progressively larger neighborhoods, are not fully exploited.

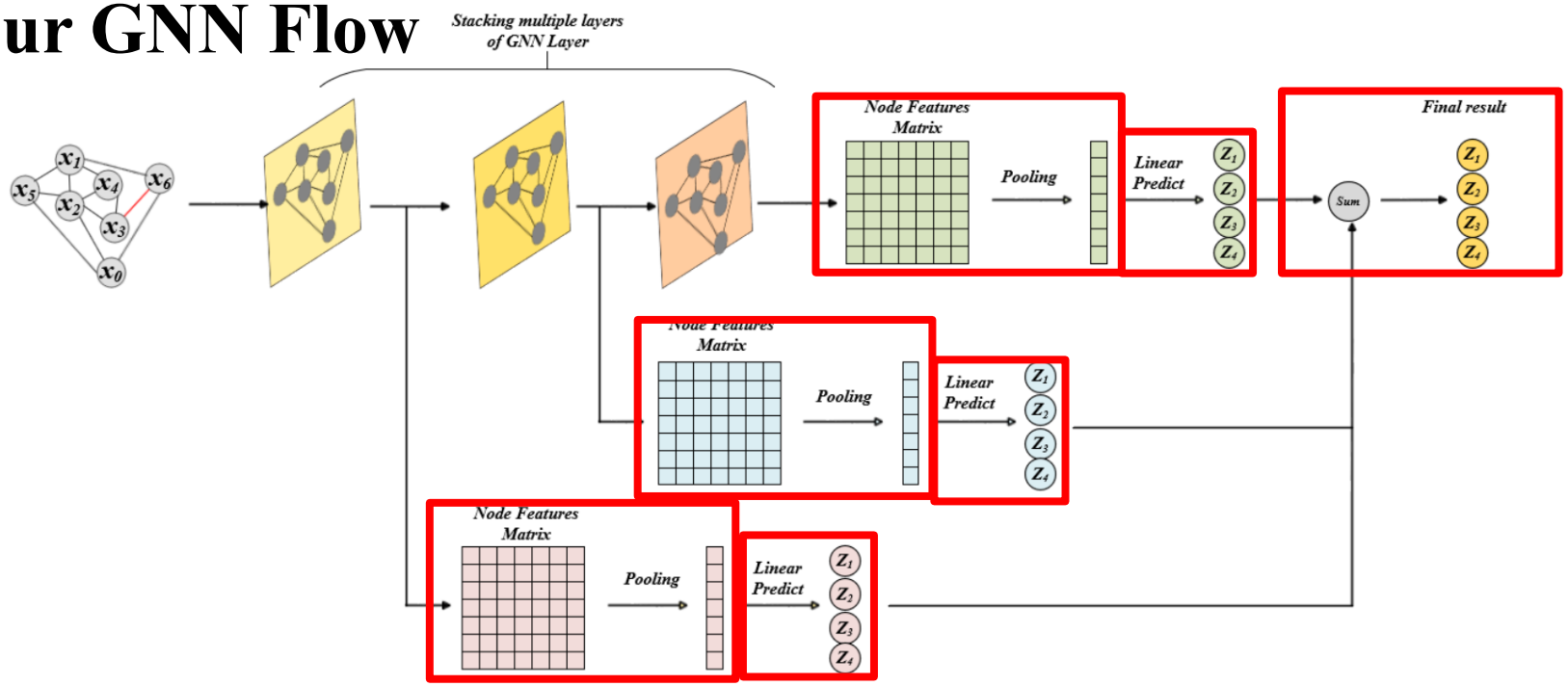
Pooling at Final Layer Only: Only the last layer's features are used for downstream tasks, ignoring intermediate layers.



Classical GNN Flow

Layer-by-Layer Pooling and Prediction

Our GNN Flow



Layer-by-Layer Pooling: Pooling node features after each layer to capture multilevel graph information.

Layer-by-Layer Prediction: Predict class scores at each layer to utilize different levels of feature representation.

Final Classification: Sum scores from all layers to obtain the final classification, integrating local and global features.

Outline

- Background and Motivation
- Proposed Method
  - GPTA Framework
  - Graph Representation
  - EnhanceSAGE
  - Layer-by-Layer Pooling and Prediction
- **Experiment Results**
- Conclusions and Future Work

Experiment Results

- The performance of GPTA under four different PTA methods

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

GPTA achieves average speedups of 9.0x, 3.1x, 1.6x, and 2.4x for PPTA, DPTA, CEPTA, and RPTA, respectively.

Experiment Results

➢ The performance of GPTA under four different PTA methods

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

GPTA achieves significant speedups for circuits like mux8 (62.2x) and nagle (20.0x) under PPTA, as the solver shows significant variation with different parameters, and our method selects the optimal ones.

Experiment Results

➢ The performance of GPTA under four different PTA methods

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

GPTA enables convergence in most previously non-converging circuits, demonstrating improved robustness across all PTA strategies.

Experiment Results

- GNN Model Performance Comparison

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

- EnhanceSAGE outperforms other models, achieving the best results across all metrics.
- **Model design innovation:** EnhanceSAGE integrates multi-layer feature aggregation with Layer-by-layer Pooling and Prediction.

Experiment Results

- Ablation Experiments

Classification Metrics After Component Removal

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

1. **Multihead mechanism**: Its removal causes the largest drop in performance, particularly in Accuracy and F1-Score, highlighting its importance in capturing complex node relationships.
2. **Adaptive message filtering**: Removing it leads to a noticeable performance decrease, especially in Precision and Recall, but its impact is smaller compared to the multi-head mechanism.
3. **Final Output Fusion**: Its absence results in a moderate performance decline, underlining its role in aggregating multi-scale information for better model generalization.

Outline

- Background and Motivation
- Proposed Method
  - GPTA Framework
  - Graph Representation
  - EnhanceSAGE
  - Layer-by-Layer Pooling and Prediction
- Experiment Results
- **Conclusions and Future Work**

Conclusions

- We introduce GPTA, a DC solver in SPICE Simulation that leverages GNN to adaptively select optimal pseudo-element embedding positions by extracting circuit topology features.

- The framework integrates EnhanceSAGE, which captures circuit features through multi-head message passing, adaptive filtering, and multi-scale fusion, boosting embedding prediction efficiency and accuracy.

- GPTA outperforms traditional methods with 9.0x improvement under PPTA, 3.1x under DPTA, 1.6x under CEPTA, and 2.4x under RPTA, while also enhancing convergence and solving previously non-converging circuits.

Future Work

Limitations in our current work:

- Simplistic initial node feature vector cannot fully represent the complexity of the graph structure.
- GNN focuses on local neighborhood features but struggles to capture global information.

How can we better extract circuit topology features?

Future Directions:

- Using more complex, domain-specific node features, such as learnable node embeddings, could enrich the model's input and improve performance.
- Combining the strengths of GNN and Transformer could enhance the representation of complex graph structures and improve classification performance.

超级科学软件实验室
Super Scientific Software Laboratory

Welcome to collaborate with us!

Thanks!

Email: jinzhou@cup.edu.cn