Mixed-Size Placement Prototyping Based on Reinforcement Learning with Semi-Concurrent Optimization

Cheng-Yu Chiang, Yi-Hsien Chiang, Chao-Chi Lan, Yang Hsu, Che-Ming Chang, Shao-Chi Huang, Sheng-Hua Wang, Yao-Wen Chang, and Hung-Ming Chen

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

National Taiwan University

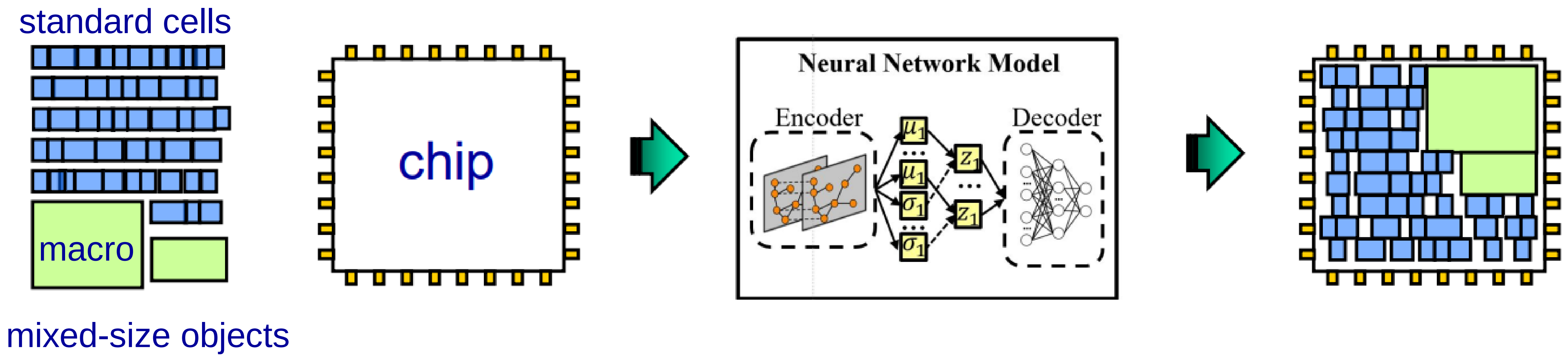
Outline

Outline

- Introduction
- Problem Formulation
- Proposed Approach
- Experimental Results
- Concluding Remarks

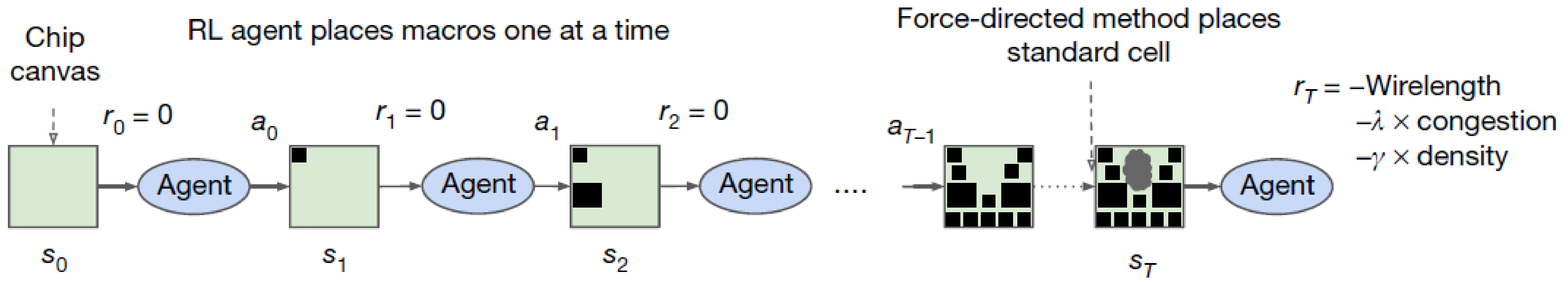
Circuit Placement Based on Machine Learning

- Place objects (macros and standard cells) into a die s.t. no objects overlap with each other & some cost metric (e.g., wirelength & power) is optimized
  - Critical and time-consuming stage that greatly affects overall layout quality
  - Should consider various constraints, multiple objectives, and new technologies
- Explore effective & efficient machine learning (ML) techniques for placement
  - Aim to achieve better and faster solutions



1st RL-Based Placement [Mirhoseini *et al.*, Nature, 2021]

- Formulate as a sequential Markov decision process (MDP)
  - (S, A, p, r): State S and action A spaces, transition dynamics p, and reward signal r
- Reinforcement Learning (RL)-based method
  - Applies an edge-based graph neural network (GNN) to generate a low-dimensional vector representation
  - Trains a neural network to predict rewards on placements of new netlists
  - Groups millions of standard cells into few thousand clusters by hMetis min-cut partitioning
  - Discretizes a few thousand grid cells and place macros and standard cell clusters onto the centers of the grid cells



Chip canvas
RL agent places macros one at a time
Force-directed method places standard cell
r_T = −Wirelength −λ × congestion −γ × density
$r_0 = 0$
a_0
$r_1 = 0$
a_1
$r_2 = 0$
a_{T-1}
Agent
Agent
Agent
Agent
s_0
s_1
s_2
s_T

A. Mirhoseini *et al.*, “A graph placement methodology for fast chip design,” *Nature,* 2021

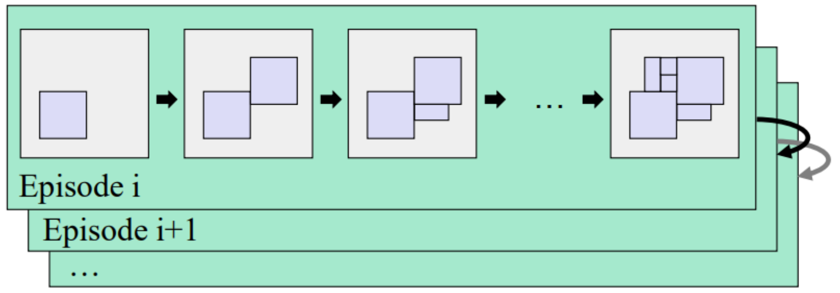
State-of-the-Art ML-Based Placement Works

- GraphPlanner [Liu et al., TODAES'22]
  - An efficient GCN-based floorplanning algorithm
  - Learns the optimized mapping between **circuit connectivity** and **physical wirelength**
- NTU/MTK/Maxeda on RL-based Placement [Chang et al., DAC'22]
  - A flexible **multiple-objective** RL model (MORL)
- MaskPlace [Lai et al., NeurIPS'22]
  - A CNN-based RL placer
  - Recasts chip placement as a problem of **visual representation** learning
- ChiPFormer [Lai et al., ICML'23]
  - A transformer-based RL placer
  - Proposes an offline RL formulation which enables learning a **transferable** placement policy

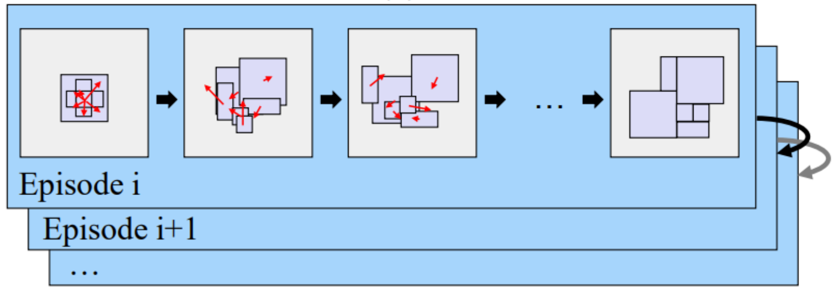
F.-C. Chang et al., "Flexible chip placement via reinforcement learning: late breaking results," in DAC, 2022
Y. Liu et al., "GraphPlanner: Floorplanning with graph neural network," TODAES, 2022
Y. Lai et al., "MaskPlace: Fast chip placement via reinforced visual representation learning," NeurIPS, 2022
Y. Lai et al., "ChiPFormer: Transferable chip placement via offline decision transformer," in ICML, 2023

Our Motivation

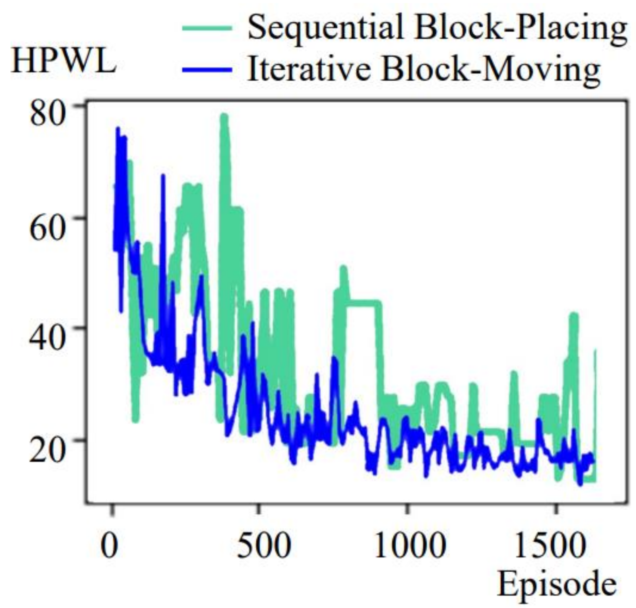
- **Iterative block-moving strategy** learns better than sequential block-placing strategy
  - Place blocks at the chip center and iteratively move blocks to obtain better placement
  - The concurrent movement enables the RL agent to learn the cooperative relationships between multiple actions simultaneously
  - Utilize dense rewards and comprehensive layout information to achieve better convergence



Sequential block-placing



Iterative block-moving



HPWL convergence

Main Contributions

- Propose the first RL-based placer to learn a placement policy for **iteratively moving blocks**, which can characterize dense rewards and comprehensive layout information in each step
- Propose a **semi-concurrent moving mechanism** to learn the collaborative dynamics among actions on a subset of blocks at each step
- Develop a deep Q-learning-based model with continuous action spaces to learn a semi-concurrent moving policy for obtaining desired actions
- Develop a three-stage learning framework considering the result after downstream post-placement
- Experimental results show that our methodology can achieve **10.9%, 7.4%, and 34.9% better HPWL** than the analytical placer DREAMPlace 4.0 (w. NTUplace3) [Liao *et al*., TCAD'23], ML-based floorplanner GraphPlanner [Liu *et al*., TODAES'22], and RL-based placer [Mirhoseini *et al*., Nature'21], respectively

P. Liao *et al*., "DREAMPlace 4.0: Timing-driven placement with momentum-based net weighting and Lagrangian-based refinement," *TCAD*, 2023
Y. Liu *et al*., "GraphPlanner: Floorplanning with graph neural network," *TODAES*, 2022
A. Mirhoseini *et al*., "A graph placement methodology for fast chip design," *Nature,* 2021

Outline

- Introduction
- Problem Formulation
- Proposed Approach
- Experimental Results
- Concluding Remarks

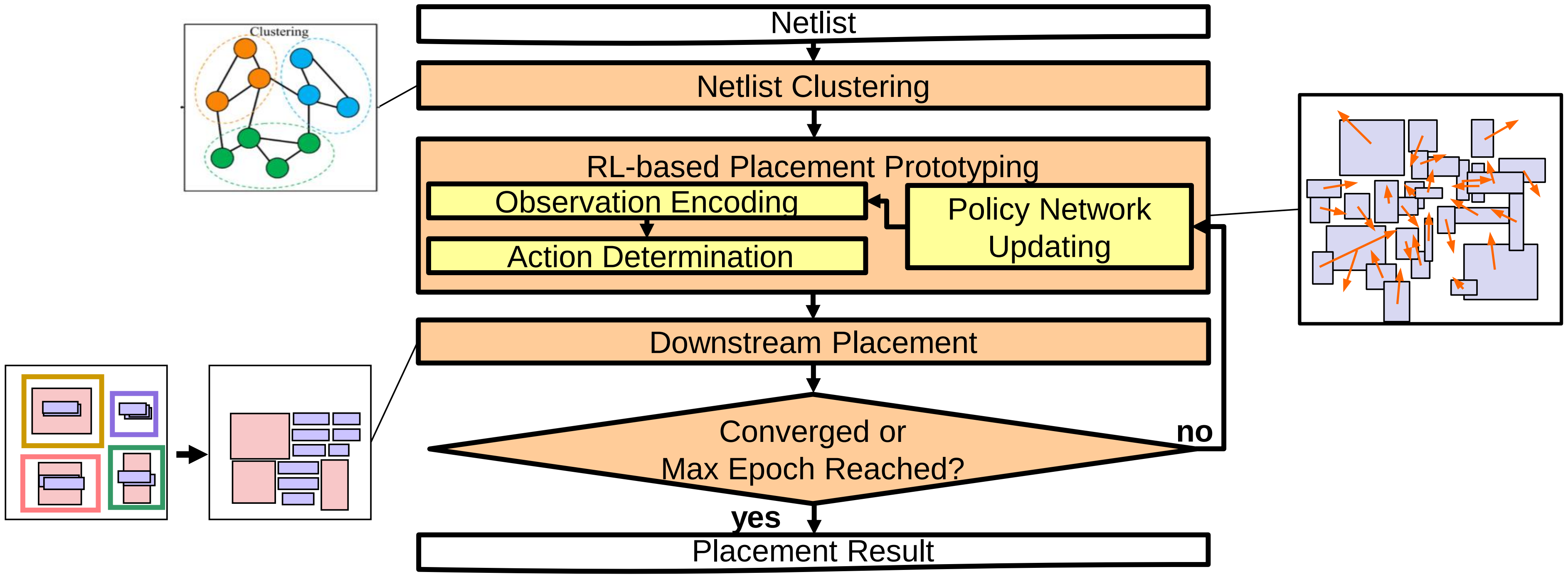
Problem Formulation

- Inputs:
  - A netlist
  - A set of movable blocks (macros and standard cells)
- Outputs:
  - **A trained policy model** to find a desired placement prototype that guides a downstream analytical placer to minimize HPWL
  - **A placement result** with minimized wirelength
- Objective:
  - HPWL minimization
- Constraints:
  - Non-overlapping constraint
  - Boundary constraint

Outline

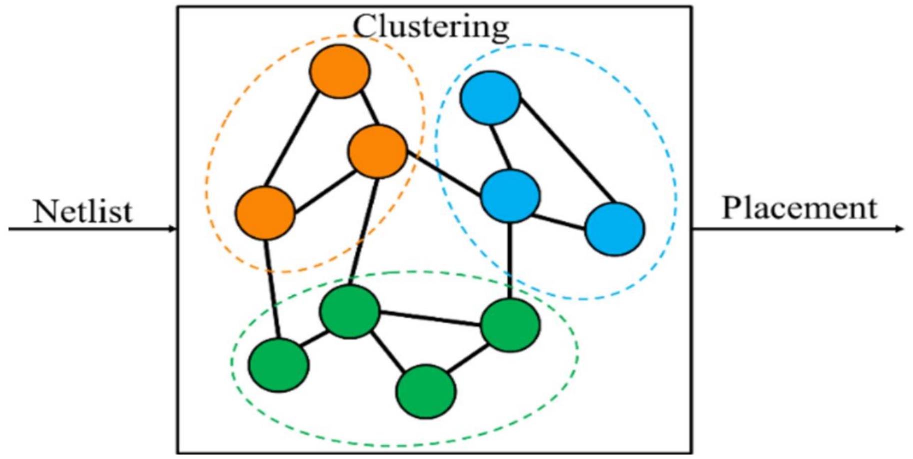
Overview Flow

- Three stages in our RL-based mixed-size chip placement framework
  - **Netlist clustering:** Reduce the problem size to better suit the RL-based model
  - **RL-based placement prototyping**: Train a model to find a desired placement prototype considering prototyping wirelength, density, and post-placement wirelength
  - **Downstream placement:** Decluster the prototyping result and apply an analytical-based placer to complete the placement



Netlist Clustering

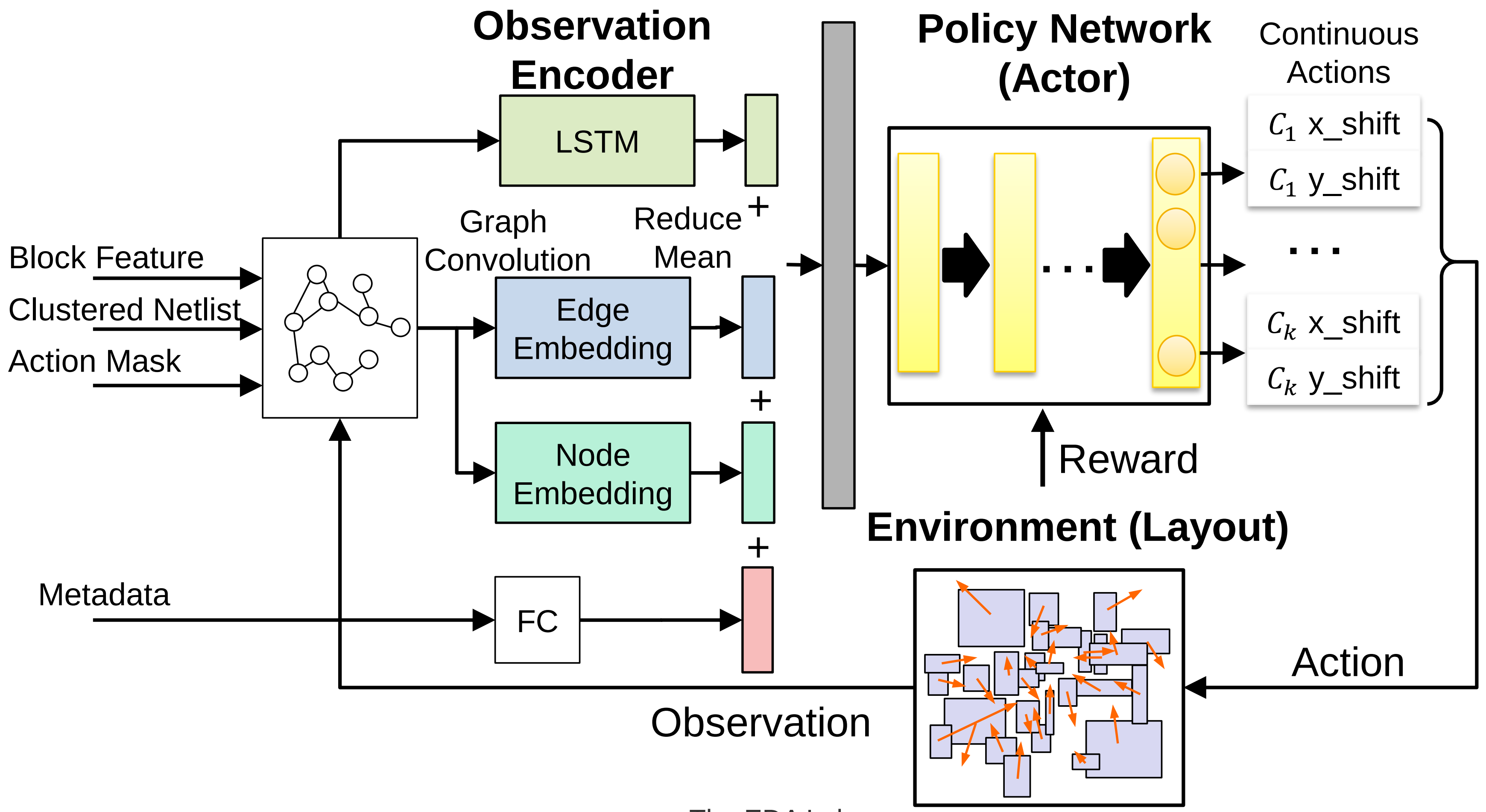
- Apply hMetis min-cut partitioning for netlist clustering to reduce problem sizes for the RL model which often cannot handle high-dimensional inputs
  - Minimize cut size with balanced area
  - Normalize blocks area (node weights) by a sigmoid function to reduce the size differences
  - Cluster macros and standard cells separately, smoothing the optimization for subsequent mixed-size analytical placement



Netlist clustering

RL Model Architecture for Iterative Block Moving

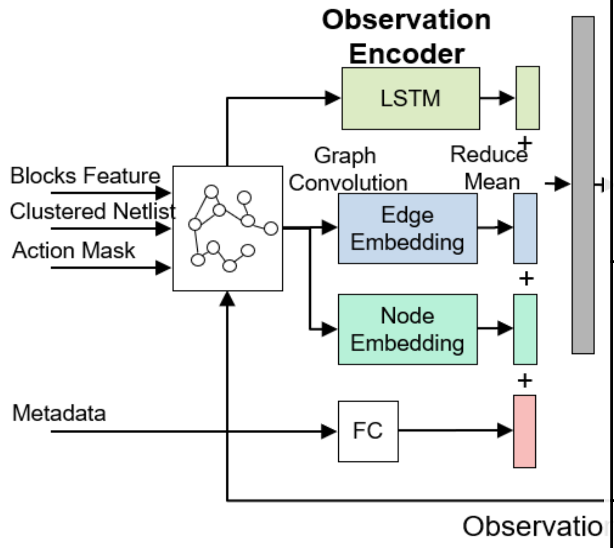
- Contains an observation encoder and a policy network
  - Extracts features and encodes observations from the layout environment
  - Obtains desired movement with encoded observations

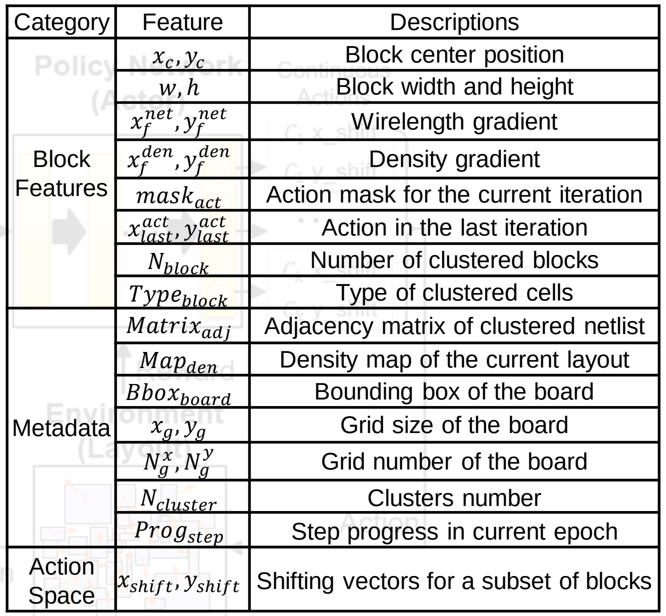


Observation Encoding

- Encode and reduce the dimension of the obtained observations
  - Three main components for encoding different information

1. Long short-term memory (LSTM) layer
   - Captures temporal dependencies for understanding the placement process
   - Learns the relationship between actions in different time steps
2. GCN-based edge and node embedding models
   - Embed and propagate features to related nodes and edges
   - Capture spatial relationships and the connectivity within the clustered netlist
3. Fully connected (FC) encoder
   - Encodes other layout features





Category	Feature	Descriptions
Block Features	x_c, y_c	Block center position
	w, h	Block width and height
	x_f^{net}, y_f^{net}	Wirelength gradient
	x_f^{den}, y_f^{den}	Density gradient
	$mask_{act}$	Action mask for the current iteration
	$x_{last}^{act}, y_{last}^{act}$	Action in the last iteration
	N_{block}	Number of clustered blocks
	$Type_{block}$	Type of clustered cells
Metadata	$Matrix_{adj}$	Adjacency matrix of clustered netlist
	Map_{den}	Density map of the current layout
	$Bbox_{board}$	Bounding box of the board
	x_g, y_g	Grid size of the board
	N_g^x, N_g^y	Grid number of the board
	$N_{cluster}$	Clusters number
	$Prog_{step}$	Step progress in current epoch
Action Space	x_{shift}, y_{shift}	Shifting vectors for a subset of blocks

Feature table

Force-Directed Features

- Iterative block-moving strategy provides dense rewards and comprehensive layout information in each time step
- So block features from the force-directed method are obtainable
  - Wirelength: Pulling force from wirelength gradient
  - Density: Pushing force from density gradient

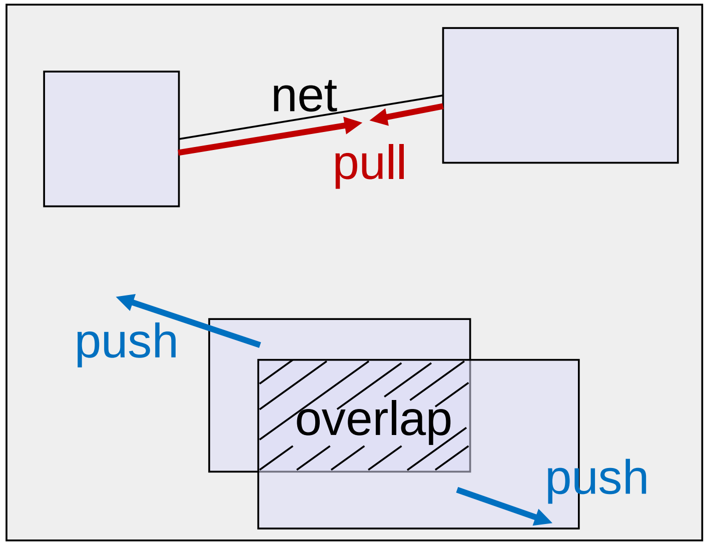


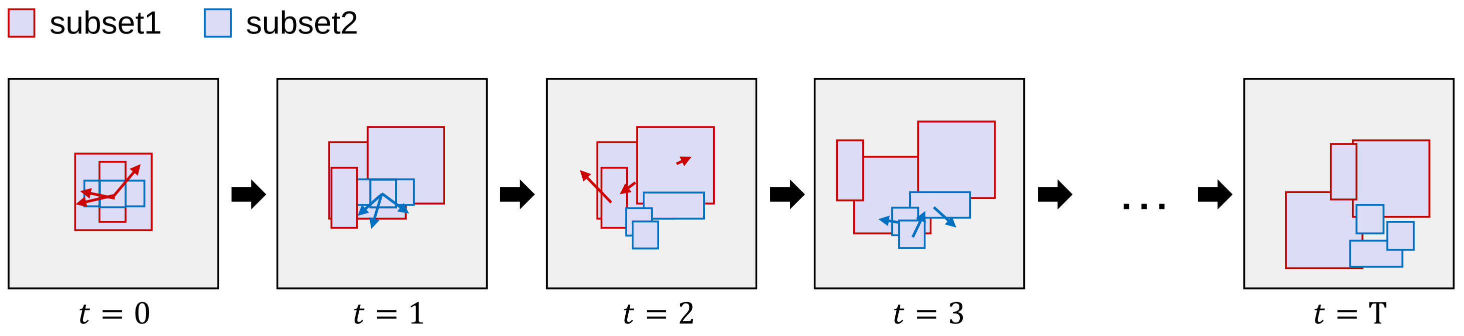
Illustration of force-directed features

Category	Feature	Descriptions
Block Features	x_c, y_c	Block center position
	w, h	Block width and height
	x_f^{net}, y_f^{net}	Wirelength gradient
	x_f^{den}, y_f^{den}	Density gradient
	$mask_{act}$	Action mask for the current iteration
	$x_{last}^{act}, y_{last}^{act}$	Action in the last iteration
	N_{block}	Number of clustered blocks
	$Type_{block}$	Type of clustered cells
Metadata	$Matrix_{adj}$	Adjacency matrix of clustered netlist
	Map_{den}	Density map of the current layout
	$Bbox_{board}$	Bounding box of the board
	x_g, y_g	Grid size of the board
	N_g^x, N_g^y	Grid num of the board
	$N_{cluster}$	Clusters number
	$Prog_{step}$	Step progress in current epoch
Action Space	x_{shift}, y_{shift}	Shifting vectors for a subset of blocks

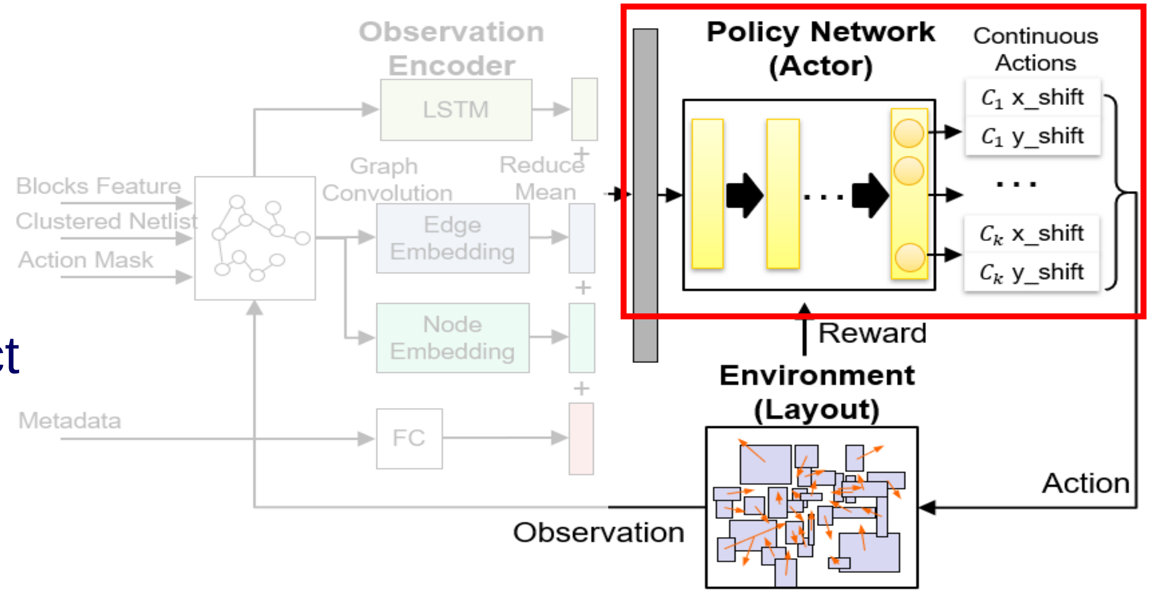
Feature table

Action Determination

- Our RL policy network simultaneously decides a shift vector for blocks
  - Apply semi-concurrent movement to move a subset of blocks with feasible action dimension
  - Learn the collaborative dynamics among these blocks and improve the overall efficiency

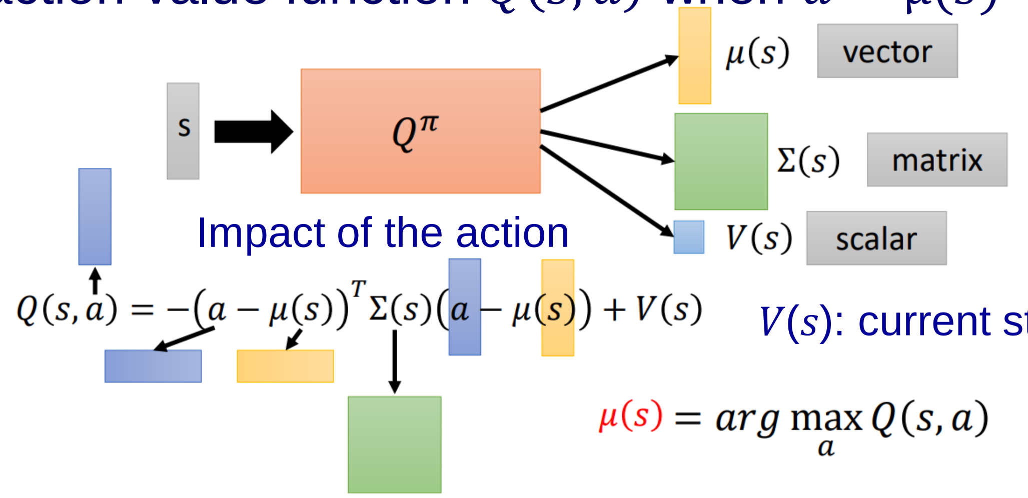


- DQN-based policy network
  - Policy-based approaches failed to learn effectively for this challenging problem
  - Use the value-based approach to predict potential rewards for states and actions



Normalized Advantage Functions (NAF) [Gu *et al*., ICML'16]

- Allows the DQN-based RL agent to decide multiple actions simultaneously
- Reduces the difficulty of the $arg\ max$ problem for DQN with continuous action space (CAS)
  - a: action for state s (Gaussian distribution)
  - $\mu(s)$: target action (Gaussian mean)
  - $\Sigma(s)$: positive definite matrix (Gaussian various)
  - $V(s)$: value function as dueling networks (current state value)
  - Have the maximum action-value function $Q(s,a)$ when $a = \mu(s)$



$$Q(s,a) = -\left(a-\mu(s)\right)^T \Sigma(s)\left(a-\mu(s)\right) + V(s)$$

$V(s)$: current state value

$$\mu(s) = arg\max_{a} Q(s,a)$$

S. Gu *et al*., "Continuous deep Q-learning with model-based acceleration," in *ICML*, 2016 (DeepMind)

Policy Network Update

- Aims to learn a prototyping policy to obtain a desired prototyping result
  - Guide downstream analytical placement to generate a placement with minimized HPWL

- Two-stage learning process
  - Stage 1 (95% epochs): Learn a good moving policy that places the clustered blocks to minimize costs
  - The moving reward is the cost variation for internal steps or the final cost for the final step

$$r(s_t, a_t) = \begin{cases} -\left(\alpha \Delta \widetilde{W_t} + (1-\alpha)\Delta \widetilde{D_t}\right), & if\ t < T \\ -\left(\beta \widetilde{W_t} + (1-\beta)\widetilde{D_t}\right), & if\ t = T, \end{cases}$$

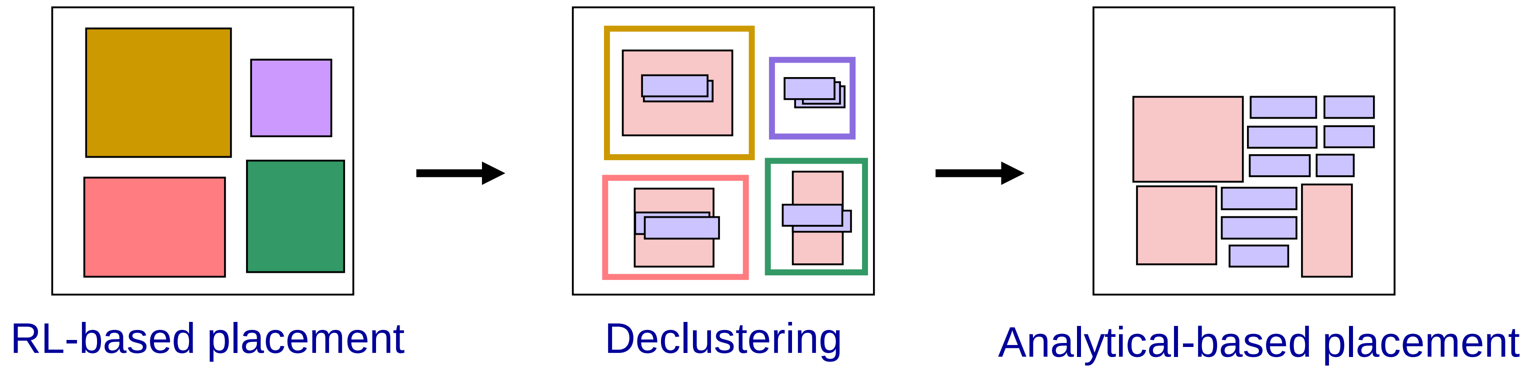
where $\widetilde{W_t}$ and $\widetilde{D_t}$ are normalized costs of HPWL and density

  - Stage 2 (last 5% epochs): Learn a good prototyping policy that guides downstream analytical placement to minimize HPWL
  - The reward of the final step is normalized by placement wirelength

$$r'(s_{\mathrm{T}}, a_T) = r(s_T, a_T)\, W^P / W^P_{norm}$$

Analytical Mixed-Size Placement

- Evaluation of the prototyping result with an analytical-based placer
  - Use declustered prototyping result as an initial placement
  - Return the final HPWL of the placement result as a reward
- Complete downstream mixed-size placement
  - Decluster the prototyping result
  - Apply DREAMPlace (w. NTUplace3) [Liao *et al.*, TCAD'23] for global placement, legalization, and detailed placement (NTUplace3)



P. Liao *et al.*, "DREAMPlace 4.0: Timing-driven placement with momentum-based net weighting and Lagrangian-based refinement," in *TCAD,* 2023

Outline

Experimental Settings

- Platform
  - Python programming language
  - PyTorch for RL-based model implementation
  - AMD EPYC 7313 @ 3.0 GHz Linux workstation with 192 GB memory
  - NVIDIA RTX A6000 GPU *1

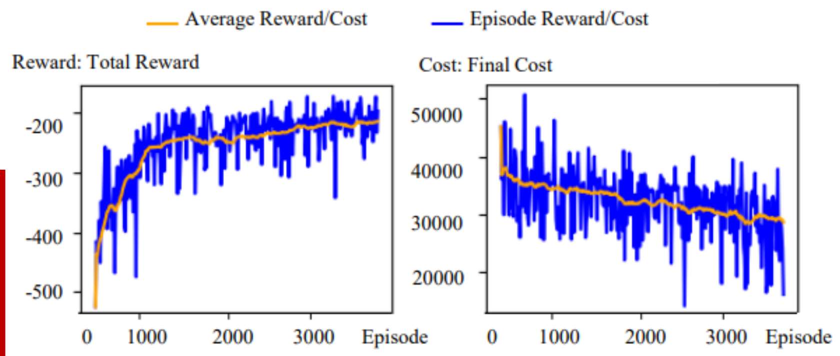
- Comparison of HPWL using the ISPD'05 benchmark suite
  - Analytical-based mixed-size placer (DREAMPlace, with NTUplace3 as its detailed placer) [Liao *et al*., TCAD'23]
  - ML-based floorplanner (GraphPlanner) [Liu *et al*., TODAES'22]
  - RL-based placer (CT) [Mirhoseini *et al*., Nature'21]

P. Liao *et al*., "DREAMPlace 4.0: Timing-driven placement with momentum-based net weighting and Lagrangian-based refinement," *TCAD*, 2023
Y. Liu *et al*., "GraphPlanner: Floorplanning with graph neural network," *TODAES*, 2022
A. Mirhoseini *et al*., "A graph placement methodology for fast chip design," *Nature,* 2021

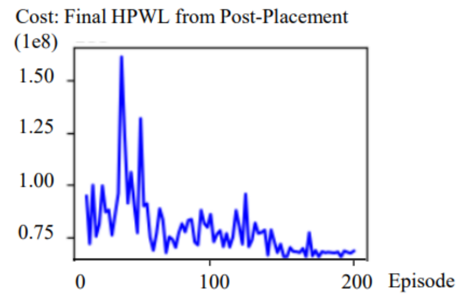
Experimental Results

- Achieves respective 10.9%, 7.4%, and 34.9% HPWL reduction compared with *DREAMPlace* (w. *NTUplace3*), *GraphPlanner*, and *CT* (Google's work)
- Our policy model can obtain a better prototype for guiding downstream analytical placer (*DREAMPlace w. NTUplace3*) than GraphPlanner
- Our model learned better policies with the iterative moving strategy
  - Provides comprehensive layout information in each step
  - Is faster with more stable convergence

	DREAMPlace		***GraphPlanner***		***CT***		**Ours**	
	WL	*R.*	*WL*	*R.*	*WL*	*R.*	*WL*	*R.*
adaptec1	6.560	1.014	6.550	1.013	8.670	1.341	6.467	1.000
adaptec2	10.110	1.363	7.750	1.045	12.410	1.673	7.419	1.000
adaptec3	15.630	1.098	15.080	1.059	25.800	1.812	14.238	1.000
adaptec4	14.410	1.021	14.270	1.011	25.580	1.813	14.113	1.000
bigblue1	8.520	0.998	8.590	1.006	16.850	1.973	8.541	1.000
bigblue2	12.570	1.027	12.720	1.039	14.200	1.160	12.237	1.000
bigblue3	46.060	1.380	**--**	--	36.480	1.093	33.365	1.000
bigblue4	79.500	1.081	**--**	--	104.000	1.414	73.556	1.000
Avg.		1.123		1.079		1.535		1.000



Learning trend of rewards and final costs



Learning trend of final HPWL

Outline

- Introduction
- Problem Formulation
- Proposed Approach
- Experimental Results
- Concluding Remarks

Conclusions

- Proposed the first RL-based placer to learn a placement policy for iteratively moving blocks
- Proposed a semi-concurrent moving mechanism to learn the collaborative dynamics among actions on a subset of blocks at each step
- Developed a DQN-based model with CAS to obtain the desired actions concurrently for a subset of blocks
- Developed a learning framework to train our model considering the result after the downstream analytical placement
- Our model improves the HPWL compared with the state-of-the-art analytical placer, ML-based floorplanner, and RL-based placer

Thank You!

References

- S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, “Continuous deep Q-learning with model-based acceleration,” in *Proc. of ICML*, New York City, NY, pp. 2829–2838, June 2016.
- C.-C. Huang, H.-Y. Lee, B.-Q. Lin, S.-W. Yang, C.-H. Chang, S.-T. Chen, Y.-W. Chang, T.-C. Chen, and I. Bustany, “NTUplace4dr: A detailed-routing-driven placer for mixed-size circuit designs with technology and region constraints,” *IEEE Tran. on CAD*, vol. 37, no. 3, pp. 669–681, 2017.
- G. Huang, J. Hu, Y. He, J. Liu, M. Ma, Z. Shen, J. Wu, Y. Xu, H. Zhang, K. Zhong et al., “Machine learning for electronic design automation: A survey,” ACM Tran. on DAES, vol. 26, no. 5, pp. 1–46, 2021.
- G. Karypis and V. Kumar, “Multilevel k-way hypergraph partitioning,” in *Proc. of DAC*, New Orleans, Louisiana, pp. 343–348, June 1999.
- M.-C. Kim, N. Viswanathan, C. J. Alpert, I. L. Markov, and S. Ramji, “MAPLE: Multilevel adaptive placement for mixed-size designs,” in *Proc. of ISPD*, San Francisco, California, June 2012
- T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” arXiv preprint arXiv:1609.02907, 2016.
- T. N. Kipf and M. Welling, “Variational graph auto-encoders,” arXiv preprint arXiv:1611.07308, 2016.
- J. M. Kleinhans, G. Sigl, F. M. Johannes, and K. J. Antreich, “GORDIAN: VLSI placement by quadratic programming and slicing optimization,” *IEEE Tran. on CAD*, vol. 10, no. 3, pp. 356–365, 1991.
- A. Jaiswal, A. R. Babu, M. Z. Zadeh, D. Banerjee, and F. Makedon, “A survey on contrastive self-supervised learning,” arXiv preprint arXiv:2011.00362, 2021.
- Y. Lai, Y. Mu, and P. Luo, “MaskPlace: Fast chip placement via reinforced visual representation learning,” *Adv. In NeurIPS*, vol. 35, no. 1812, pp. 24 019–24 030, 2022.

References

- A. Agnesina, P. Rajvanshi, T. Yang, G. Pradipta, A. Jiao, B. Keller, B. Khailany, and H. Ren, “AutoDMP: Automated dreamplace-based macro placement,” in *Proc. of ISPD*, Virtual Event, USA, pp. 149–157, March 2023.
- F.-C. Chang, Y.-W. Tseng, Y.-W. Yu, S.-R. Lee, A. Cioba, I.-L. Tseng, D.-s. Shiu, J.-W. Hsu, C.-Y. Wang, C.-Y. Yang, R.-C. Wang, Y.-W. Chang, T.- C. Chen, and T.-C. Chen, “Flexible chip placement via reinforcement learning: late breaking results,” in *Proc. of DAC*, San Francisco, California, pp. 1–2, July 2022.
- Y.-C. Chang, Y.-W. Chang, G.-M. Wu, and S.-W. Wu, “B*-trees: A new representation for non-slicing floorplans,” in *Proc. of DAC*, New York, NY, pp. 458–463, June 2000.
- T.-C. Chen, P.-H. Yuh, Y.-W. Chang, F.-J. Huang, and T.-Y. Liu, “MP-trees: A packing-based macro placement algorithm for modern mixed-size designs,” *IEEE Tran. on CAD*, vol. 27, no. 9, pp. 1621–1634, 2008.
- T.-C. Chen, Z.-W. Jiang, T.-C. Hsu, H.-C. Chen, and Y.-W. Chang, “NTUplace3: An analytical placer for large-scale mixed-size designs with preplaced blocks and density constraints,” *IEEE Tran. on CAD*, vol. 27, no. 7, pp. 1228–1240, 2008.
- C.-K. Cheng, A. B. Kahng, I. Kang, and L. Wang, “RePlAce: Advancing solution quality and routability validation in global placement,” *IEEE Tran. on CAD*, vol. 38, no. 9, pp. 1717–1730, 2018.
- R. Cheng, X. Lyu, Y. Li, J. Ye, J. Hao, and J. Yan, “The policy-gradient placement and generative routing neural networks for chip design,” Adv. In NeurIPS, vol. 35, no. 1995, pp. 26 350–26 362, 2022.
- R. Cheng and J. Yan, “On joint learning for solving placement and routing in chip design,” *Adv. In NeurIPS*, vol. 34, no. 1305, pp. 16 508–16 519, 2021.
- H. Eisenmann and F. M. Johannes, “Generic global placement and floorplanning,” in *Proc. of DAC*, San Francisco, California, pp. 269–274, June 1998.

References

- Y. Lai, J. Liu, Z. Tang, B. Wang, J. Hao, and P. Luo, “ChiPFormer: Transferable chip placement via offline decision transformer,” in *Proc. of ICML*, San Francisco, California, pp. 1–19, April 2023.
- P. Liao, D. Guo, Z. Guo, S. Liu, Y. Lin, and B. Yu, “DREAMPlace 4.0: Timing-driven placement with momentum-based net weighting and Lagrangian-based refinement,” *IEEE Tran. on CAD*, vol. 42, no. 10, pp. 3374–3387, 2023.
- Y. Liu, Z. Ju, Z. Li, M. Dong, H. Zhou, J. Wang, F. Yang, X. Zeng, and L. Shang, “Floorplanning with graph attention,” in *Proc. of DAC*, San Francisco, California, p. 1303–1308, July 2022.
- Y. Liu, Z. Ju, Z. Li, M. Dong, H. Zhou, J. Wang, F. Yang, X. Zeng, and L. Shang, “GraphPlanner: Floorplanning with graph neural network,” *ACM Tran. on DAES*, vol. 28, no. 2, pp. 1303–1308, 2022
- J. Lu, H. Zhuang, P. Chen, H. Chang, C.-C. Chang, Y.-C. Wong, L. Sha, D. Huang, Y. Luo, C.-C. Teng et al., “ePlace-MS: Electrostatics-based placement for mixed-size circuits,” *IEEE Tran. on CAD*, vol. 34, no. 5, pp. 685–698, 2015.
- Y.-C. Lu, T. Yang, S. K. Lim, and H. Ren, “Placement optimization via PPA-directed graph clustering,” in *Proc. of MLCAD*, Virtual Event, China, pp. 1–6, September 2022.
- A. Mirhoseini, A. Goldie, M. Yazgan, J.W. Jiang, E. Songhori, S.Wang, Y.-J. Lee, E. Johnson, O. Pathak, A. Nazi et al., “A graph placement methodology for fast chip design,” *Nature*, vol. 594, no. 7862, pp. 207–212, 2021
- V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski et al., “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

References

- Y. Mo, L. Peng, J. Xu, X. Shi, and X. Zhu, “Simple unsupervised graph representation learning,” in *Proc. of AAAI*, Palo Alto, California, pp. 7797–7805, June 2022.
- H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, “Rectangle-packing-based module placement,” in *Proc. of ICCAD*, San Jose, California, pp. 472–479, November 1995.
- G.-J. Nam, C. J. Alpert, P. Villarrubia, B. Winter, and M. Yildiz, “The ISPD2005 placement contest and benchmark suite,” in *Proc. of ISP*D, San Francisco, California, pp. 216–220, April 2005.
- N. Quinn and M. Breuer, “A forced directed component placement procedure for printed circuit boards,” *IEEE Tran. on Circuits and systems*, vol. 26, no. 6, pp. 377–388, 1979.
- F. Schroff, D. Kalenichenko, and J. Philbin, “FaceNet: A unified embedding for face recognition and clustering,” in *Proc. of CVPR*, Boston, MA, pp. 815–823, June 2015.
- R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, “Policy gradient methods for reinforcement learning with function approximation,” *Adv. in NeurIPS*, vol. 12, no. 34, pp. 1057–1063, 1999.
- P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, “Graph attention networks,” arXiv preprint arXiv:1710.10903, 2017.
- N. Viswanathan, M. Pan, and C. Chu, “FastPlace 3.0: A fast multilevel quadratic placement algorithm with placement congestion control,” in *Proc. of DAC*, San Diego, California, pp. 135–140, June 2007.
- Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, “Dueling network architectures for deep reinforcement learning,” in *Proc. of ICML*, New York City, NY, pp. 1995–2003, June 2016.