Miracle: Multi-Action Reinforcement Learning-Based Chip Floorplanning Reasoner

Bo Yang[1], Qi Xu*[1], Hao Geng[2], Song Chen[1], Yi Kang[1]
[1]University of Science and Technology of China [2]ShanghaiTech University

***Abstract*—Floorplanning is one of the most critical but time-consuming tasks in the chip design process. Machine learning techniques, especially reinforcement learning, have provided a promising direction for floorplanning design. In this paper, an end-to-end reinforcement learning (RL) framework is proposed to learn a policy for floorplanning automatically, in the combination of edge-augmented graph attention network (EGAT), position-wise multi-layer perceptron, and gated self-attention mechanism. We formulate floorplanning as a Markov Decision Process (MDP) model, where a multi-action mechanism and a dense reward function are developed to adapt the floorplanning problem. In addition, in order to make full use of prior knowledge, we further propose a supervised learning approach on the generated synthetic netlist-floorplan dataset. Experimental results demonstrate that, compared with state-of-the-art floorplanners, the proposed end-to-end framework significantly reduces wirelength with a smaller area.**

I. Introduction

In the back-end physical design of integrated circuits (ICs), chip floorplanning aims to determine the geometric relationship among circuit macros with taking multiple design metrics into account, i.e., area and wirelength. The quality of floorplanning directly affects the subsequent placement and routing. As a well-known combinatorial optimization problem, floorplanning has been proven to be an NP-hard problem [1]. To produce high-quality floorplan solutions, human experts design many heuristic-based floorplanning algorithms, which usually require specialized knowledge and massive efforts [2].

In the past decade, artificial intelligence (AI) technologies have achieved great success in various applications, even in the floorplanning field. Google has developed a reinforcement learning (RL)-based approach to address chip floorplanning [3]. Similar to AlphaGo's strategy [4], the floorplanning task is simulated as playing chess on a chip canvas. Although Google's work demonstrated that the RL-based macro floorplanning surpasses the design generated by a human physical designer, three main issues still exist in the face of real-world chip floorplanning. One is that it cannot provide high-quality solutions for netlists with various macro sizes. Moreover, the designed agent consumes a colossal time and computing resources for determining the macro positions. Additionally, reinforcement learning (RL) algorithm often encounters sparse reward challenges, resulting in the difficulty of exploration in the experiment since the agent rarely receives a feedback reward signal.

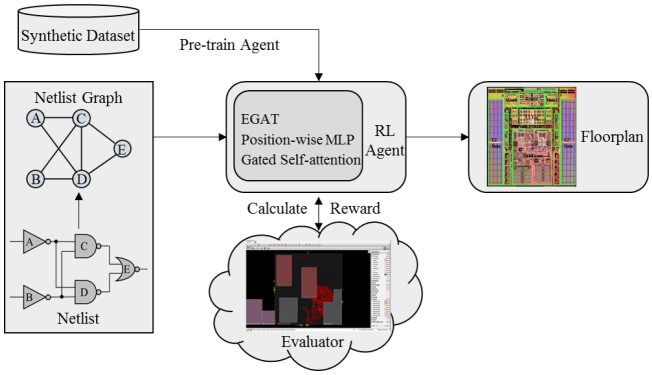


Fig. 1 Reinforcement learning paradigm floorplanning.

To address these issues, we propose an end-to-end RL-based floorplanning framework in this work, as shown in Fig. 1. Given a circuit netlist, we represent it as an undirected graph, which is fed to the RL agent. Then the RL agent sequentially places circuit macros on feasible positions of the layout. In order to train the RL agent, a two-stage training strategy is developed, first using supervised learning to pre-train the network, followed by a reinforcement learning procedure. In the proposed floorplanning framework, the evaluator serves to assist the reward calculation, which includes the best solution the agent has explored. Since the edges in netlist graph have different connection degrees (weights), an edge-augmented graph attention network (EGAT) is proposed in the encoder part to extract the netlist information effectively. Based on the information passing through the EGAT, the decoder components adopt the bit-wise deep cross policy network and gated self-attention value network to achieve higher-order feature interactions and capture the receptive field of the whole graph, respectively [5]. In addition, to tackle the sparse reward challenge, a dense reward function is devised to use the gap in cost differences between two consecutive time steps as the immediate reward. Note that the cost differences are calculated from the partial floorplan produced by the agent and the evaluator solution. Our main contributions are summarized as follows:

- An end-to-end floorplanning framework is proposed,

This work is partially supported by the National Key R&D Program of China under grant No. 2019YFB2204800, the National Natural Science Foundation of China (NSFC) under grant No. 62141415, 61931008, the CAS Project for Young Scientists in Basic Research under grant No. YSBR-029k, and the Strategic Priority Research Program of Chinese Academy of Sciences under grant No. XDB44000000.

* The corresponding author.

979-8-3503-4859-0/DATE24/© 2024 EDAA

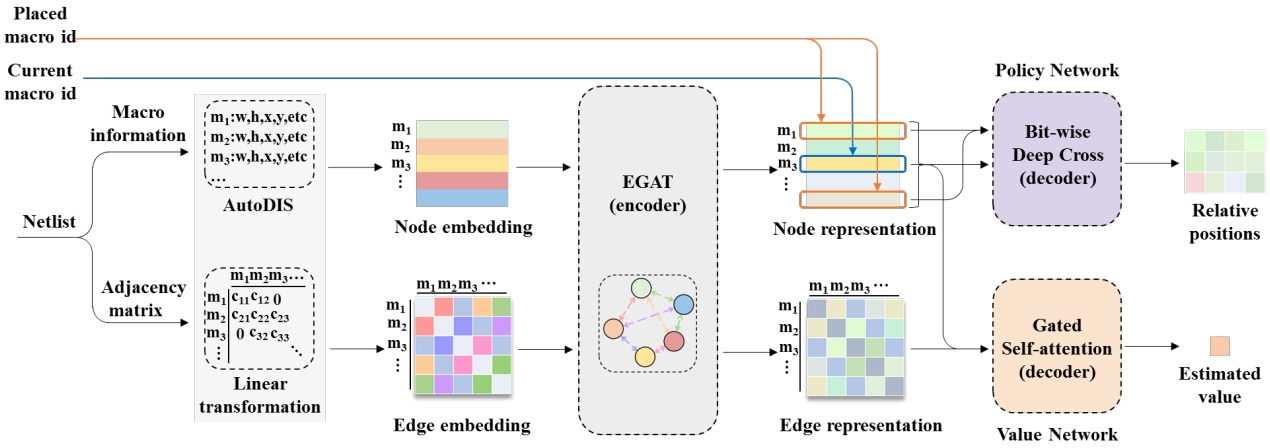


Fig. 2 The architecture of the end-to-end floorplanning network. The arrows mark the information flow of the network.

combining edge-augmented graph attention network (EGAT), bit-wise deep cross policy network, and gated self-attention value network.

- A large number of netlist datasets with the corresponding optimal floorplan solutions are generated to effectively pre-train neural networks.
- A multi-action reinforcement learning model is built, where the actions are set as the relative positions between macros to greatly simplify the search process.

The rest of the paper is organized as follows. Section II presents a brief overview of related works. Section III gives the preliminaries and formulates the floorplanning problem. Section IV describes the details of the end-to-end floorplanning techniques. Section V lists the experimental results, followed by the conclusions in Section VI.

II. Related Works

The prior arts on floorplanning can be divided into three categories. In the first class, researchers develop the simulated annealing-based heuristic [6]. The second class of works present a series of non-linear analytical solvers to handle the mixed-size placement, i.e., RePlAce [7]. However, these solvers cannot expand to address floorplanning, which has a wide range of macro sizes and a high degree of attention to the area metric. Recently, RL has shown great potential in floorplanning field. In Google's work [3], the macro placement is formulated as a sequential decision-making problem and an RL-based approach is developed to solve it. He *et al*. [8] utilizes RL schemes to explore the floorplan local search heuristics without introducing any prior knowledge. In addition, Graph Neural Network (GNN) is adopted to explore the physical meaning of netlist [9], [10], and the extracted embeddings guide the RL agent to tackle floorplanning.

III. Preliminaries

A. Reinforcement Learning

Reinforcement learning learns how to map states to actions so as to maximize a numerical reward signal [11]. Typically, RL is defined as a Markov Decision Process (MDP) defined by (S, A, P, R, γ). At each time step t, the agent receives a state $s_t \in S$, and selects an action $a_t \in A$. After the environment receives a_t, it produces a reward $r_t \in R$ and transfers to a next state $s_{t+1} \in S$ according to the transition distribution $p(s_{t+1}|s_t, a_t) \in P$. Through repeated steps, the agent's trajectory is described. The goal of RL is to find an optimal policy π_θ^* that maximizes the expected sum of discounted rewards $E_{\pi_\theta}[\sum_{k=0}^{\infty} \gamma^k r_{t+k}|s_t]$, where θ refers to the network parameters.

B. Floorplanning Formulation

Given a macro set $M = \{m_1, \ldots, m_n\}$ including n rectangular macros and a netlist N specifying connections among macros, a legal floorplan f assigns a position (x_i, y_i) for each macro m_i to minimize chip area and wirelength metrics with no overlapping constraint between any two macros. Thus, the objective function of floorplanning is defined as follows.

$$\min_{f} \quad A(f) + \eta \cdot W(f), \tag{1}$$

where η is a coefficient, and $A(\cdot)$ and $W(\cdot)$ represent the area and wirelength function, respectively. Besides, we adopt Sequence Pair (SP) as the floorplan representation [1].

IV. Methodology

In this section, we first present the proposed end-to-end floorplanning network. Then, a reinforcement learning model is built for the floorplanning. In the end, the essential techniques of the supervised learning are fully described.

A. End-to-end Floorplanning Network

In chip floorplanning, the information interaction between nodes is necessary. We propose an end-to-end floorplanning network as illustrated in Fig. 2, which consists of four salient high-level components.

Embeddings. The input to the end-to-end floorplanning network contains a circuit netlist, which can be represented by an un-directed graph $G(V, E)$. V denotes the set of

nodes (macros), while E is an $n \times n$ adjacency matrix with $E_{ij} = c_{ij}$ if nodes i and j are connected by an edge (c_{ij} refers to the connection degree), otherwise $E_{ij} = 0$. To handle macro information and adjacency matrix of the netlist graph, embeddings are needed to map the data to a dense representation in an latent space. AutoDis embedding framework [12] has a high model capacity and can generate unique representations in an end-to-end manner. Therefore, for a netlist graph, its macro information (shape, position, etc.) is embedded as $\boldsymbol{h} \in \mathbb{R}^{n \times d_n}$ by performing AutoDis. In addition, since the linearity of the values in the adjacency matrix directly reflects the connection degree among macros, the adjacency matrix is transformed into $\boldsymbol{e} \in \mathbb{R}^{n^2 \times d_e}$ through a multi-layer perceptron (MLP).

EGAT Encoder. Based on the generated node and edge embeddings, the developed EGAT encoder further extracts the features of nodes and edges. In order to increase the expressive capability of the model, an edge-augmented attention mechanism is performed for each node. Each layer l in the EGAT encoder consists of three calculation processes: attention score calculation, node embedding update, and edge embedding update. Firstly, we compute the attention score of a node with its neighborhoods as follows.

$$\begin{aligned} \boldsymbol{a}_{ij}^{l} &= \tanh(\boldsymbol{W}_i^l \boldsymbol{h}_i^l + \boldsymbol{W}_j^l \boldsymbol{h}_j^l + \boldsymbol{W}_e^l \boldsymbol{e}_{ij}^l + \boldsymbol{b}^l), \\ s_{ij}^{l} &= \frac{\exp({\boldsymbol{z}^l}^\top \boldsymbol{a}_{ij}^l)}{\sum_{j \in \mathcal{N}_i \cup i} \exp({\boldsymbol{z}^l}^\top \boldsymbol{a}_{ij}^l)}, \end{aligned} \tag{2}$$

where $\boldsymbol{h}_i^l \in \mathbb{R}^{d_n}$, $\boldsymbol{h}_j^l \in \mathbb{R}^{d_n}$ and $\boldsymbol{e}_{ij}^l \in \mathbb{R}^{d_e}$ are the embeddings of node i, node j and the corresponding edge in the l-th layer, respectively. $\boldsymbol{W}_i^l \in \mathbb{R}^{d_e \times d_n}$, $\boldsymbol{W}_j^l \in \mathbb{R}^{d_e \times d_n}$, $\boldsymbol{W}_e^l \in \mathbb{R}^{d_e \times d_e}$, $\boldsymbol{b}^l \in \mathbb{R}^{d_e}$ and $\boldsymbol{z}^l \in \mathbb{R}^{d_e}$ denote learned parameters. Besides, $\mathcal{N}_i$ represents the neighborhoods of node i in the graph. Note that for each node, we add an edge pointing to itself. The calculated attention score s_{ij}^l indicates the importance of node j's embedding to node i.

On the basis of the attention scores, the node embeddings are updated by the weighted average pooling operation. To further enhance the representation ability of the model, we perform an MLP and residual connection on node representation. The final node representation is updated as:

$$\boldsymbol{h}_i^{l+1} = \mathrm{MLP_n}(\sum_{j \in \mathcal{N}_i \cup i} s_{ij}^l \boldsymbol{h}_j^l) + \boldsymbol{h}_i^l, \tag{3}$$

where $\mathrm{MLP_n}(\cdot)$ represents a non-linear transformation with input dimension d_n and output dimension d_n. Similarly, the edge representation also ends with a residual connection:

$$\boldsymbol{e}_{ij}^{l+1} = \mathrm{MLP_e}(\widehat{\boldsymbol{W}}_e^l \boldsymbol{a}_{ij}^l) + \boldsymbol{e}_{ij}^l, \tag{4}$$

where $\mathrm{MLP_e}(\cdot)$ denotes another non-linear transformation with input dimension d_e and output dimension d_e, $\widehat{\boldsymbol{W}}_e^l \in \mathbb{R}^{d_e \times d_e}$ refers to a learning weight vector, and $\boldsymbol{a}_{ij}^l$ is the intermediate feature obtained according to Equation (2).

As demonstrated above, the proposed EGAT network introduces edge features to calculate attention scores, which are further adopted to update both node and edge features. Since the edge features reflect the connectivity information between nodes, its value is directly related to the wirelength metric in the objective function. Thus by incorporating the edge feature into the calculation of attention score, the information interaction between nodes can be better guided following the optimization goal.

Bit-wise Deep Cross Policy Network. To learn more predictive feature crosses, a bit-wise deep cross network [13] is plugged into the policy network. As depicted in Fig. 2, the bit-wise deep cross policy network receives the current and placed node representations and outputs a probability distribution over actions. The overall architecture consists of a cross network and a deep network, which are stacked sequentially. We denote $\boldsymbol{h}_c$ as the representation of the current macro to be placed onto the chip, while $\boldsymbol{h}_p$ means the representation of the p-th placed macro. At each step, two vectors are concatenated, and then fed to the cross network followed by the deep network. The core of the policy network lies in the cross layers that produce explicit feature interactions. The resulting output of the $(l+1)$-th cross layer is depicted in Equation (5).

$$\begin{aligned} \boldsymbol{x}^0 &= \mathrm{concat}(\boldsymbol{h}_c, \boldsymbol{h}_p), \\ \boldsymbol{x}^{l+1} &= \boldsymbol{x}^0 \odot (\boldsymbol{W}^l \boldsymbol{x}^l + \hat{\boldsymbol{b}}^l) + \boldsymbol{x}^l, \end{aligned} \tag{5}$$

where $\boldsymbol{x}^0 \in \mathbb{R}^{2d_n}$ is the original embedding of order 1. $\boldsymbol{W}^l \in \mathbb{R}^{2d_n \times 2d_n}$ and $\boldsymbol{b}^l \in \mathbb{R}^{2d_n}$ refer to the learned parameters in the l-th layer. $\boldsymbol{x}^l$ ($\boldsymbol{x}^{l+1}$) $\in \mathbb{R}^{2d_n}$ represents the input (output) of the $(l+1)$-th cross layer. For an L-layered cross network, the highest polynomial order is $L+1$ and the network contains all the feature interactions up to the highest order. Deep network is an MLP, which outputs the probability of the action. Among them, the cross network achieves explicit feature interactions and the deep network learns implicit feature crosses.

Gated Self-attention Value Network. In order to effectively capture the global information and enlarge the receptive filed, we adopt a transformer-like gated self-attention mechanism [14] in the value network. The decoder performs the attention over the learned representations of all nodes generated by the encoder, which are denoted as $\boldsymbol{h}$ with the last dimension d_n. We first project $\boldsymbol{h}$ into key $\boldsymbol{K}$ and value $\boldsymbol{V}$ through fully connected networks separately, and $\boldsymbol{h}_c$ is mapped into query $\boldsymbol{Q}$. Then we compute the scaled dot product of the query with all keys, divide each by $\sqrt{d_n}$, and apply a softmax function to obtain the attention score $\boldsymbol{A}$ on values. The attention mechanism ensures that information is propagated between different nodes only through the global self-attention, which is agnostic to the internal arrangement of the data [15]. The calculation procedures of the attention function are described as:

$$\begin{aligned} \boldsymbol{K}, \boldsymbol{V} &= fc_{k,v}(\boldsymbol{h}), \ \boldsymbol{Q} = fc_q(\boldsymbol{h}_c), \\ \boldsymbol{A} &= \mathrm{softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^\mathrm{T}}{\sqrt{d_n}})\boldsymbol{V}. \end{aligned} \tag{6}$$

The gated attention mechanism extends the above attention mechanism by imposing a real vector gate to control the

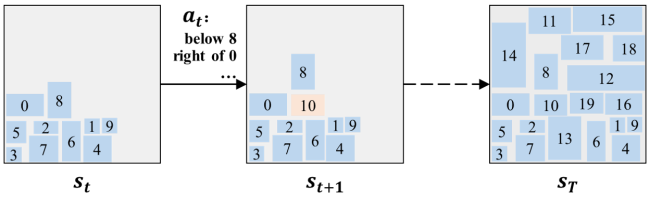


Fig. 3 The diagram of the proposed MDP model, where the RL agent sequentially places circuit macros onto chip.

flow of information, which can be considered as an activation function. As shown in Equation (7), whether or not each value in a vector is activated depends on the input, i.e. it is dynamic.

$$\begin{aligned} \boldsymbol{G} &= \sigma(fc_g(\boldsymbol{h}_c)), \\ \boldsymbol{h}_c &= fc_c(\text{concat}(\boldsymbol{G} \odot \boldsymbol{A}, \boldsymbol{h}_c)), \end{aligned} \tag{7}$$

where $\sigma(\cdot)$ denotes the sigmoid function, $\text{concat}(\cdot)$ refers to the concatenation operation, $fc_g(\cdot)$ and $fc_c(\cdot)$ represent fully connected networks. Besides, $\boldsymbol{G}$ stands for the gated activation vector, controlling which elements of $\boldsymbol{A}$ are activated. Based on the node feature $\boldsymbol{h}_c$ and the edge representation $\boldsymbol{e}$ generated by the EGAT encoder, the output of the value network is the estimated value as follows.

$$o_v = \text{MLP}(\text{concat}(\boldsymbol{h}_c, \text{mean}(\boldsymbol{e}))), \tag{8}$$

where $\text{mean}(\cdot)$ indicates a mean aggregator over all edges.

B. Reinforcement Learning Formulation

In this work, we target the chip floorplanning problem, and the RL agent sequentially places circuit macros to feasible positions on the layout. Once a macro has been placed, the corresponding reward is computed to update the RL agent. For a better understanding, Fig. 3 presents an overview process of our RL approach. The four key elements of the proposed MDP model are defined as follows:

- State Space. A state contains information about the current partial floorplan. To represent a state, we define the following features, including netlist graph (adjacency matrix) and macro information (size, placed coordinate, whether it has been placed).
- Action Space. Each action is defined as the relative position between two macros, namely left and right, right and left, top and bottom, bottom and top. Thus, at each time step, we need to take multi-actions to determine the position of a macro. That is, for the current macro to be placed, we should establish its position relationship with all placed macros in turn, and mask illegal actions.
- State Transition. Given a state and an action, the environment moves to another state. Thus, the transition model of our MDP is deterministic.
- Reward. At each time step, the evaluator contains the best solution the agent has explored and transforms it into the baseline. Then, the cost difference between the partial floorplan generated by the agent and the baseline is calculated. We designate the gap in cost differences between two consecutive time steps as the immediate reward for the current step.

Dense Reward Calculation. In chip floorplanning, we aim to optimize floorplanning metrics as defined in Equation (1). In this work, a dense reward strategy is proposed to solve the challenge. First, the evaluator records the current best floorplan solution. Then, given the evaluator solution and the placed order of macros, the negative objective cost of partial floorplan at each time step t is derived, which is denoted as a baseline value c_t^b. Meanwhile, during reinforcement learning, the agent places the macros in the same order, and the corresponding cost is recorded as c_t^a. We calculate the cost difference between the generated floorplan and the baseline, and the gap in cost differences between two consecutive time steps is set as a reward r_t, as follows.

$$\begin{aligned} c_t^b &= -A(f_t^b) - \eta \cdot W(f_t^b), \\ c_t^a &= -A(f_t^a) - \eta \cdot W(f_t^a), \\ d_t &= c_t^a - c_t^b, \ r_t = s \cdot (d_t - d_{t-1}), \end{aligned} \tag{9}$$

where f_t^b and f_t^a refer to the partial floorplan at time step t produced by evaluator and RL agent, respectively. s is a scaling factor. To speed up the network convergence, a small positive reward is produced if $d_t > 0$, and vice versa.

Order Selection. At each step, we not only select the order in which the macros are placed, but also pick the order of calculating the relative positions between the selected macro and other placed macros. In selecting macro order, the macro with the highest connection degree is placed first. In the subsequent process, the placed macros are merged into a set, and the unplaced macro with the greatest connection degree to the macros in the set is then chosen for placement. Continue until all macros are placed onto the chip. Similarly, in the action order decision, we first select the placed macro, which possesses the highest connection degree with the current macro to be placed, for relative position calculation. Then, the chosen macro and the current macro are combined into one group. Next, the unselected macro with the greatest connection degree to all macros in that group is picked to determine the relative position. Note that actions that violate the relative position constraint are masked. This process continues until the relative positions between the current macro to be placed and all placed macros are calculated.

C. Supervised Learning for Floorplanning

In order to make reinforcement learning effective, high-quality trajectories are needed to guide agents. In this work, we propose a supervised learning method for floorplanning to enforce the agent to gain knowledge on high-quality trajectories. To train the supervised model, a large number of training dataset are required, where each netlist-floorplan pair includes the optimal mapping between circuit connectivity and physical floorplan. Thus, we develop a fast data generation algorithm for floorplanning, and create a synthetic training dataset with numerous tuples of the netlist and corresponding optimal floorplan.

TABLE I Comparison with state of the arts on MCNC and GSRC benchmarks (Area unit:$\times 10^6 \mu m^2$, Wire unit:$\times 10^5 \mu m$, RT unit:s). Although we list the runtime, due to the long chip design process (several months or years), this time can be ignored.

Circuit	#Macro	#Net	SA			ICCD'20 [8]			KDD'22 [10]			Ours		
			Area	Wire	RT	Area	Wire	RT	Area	Wire	RT	Area	Wire	RT
ami33	33	123	1.275	0.59	82	**1.240**	0.69	43	–	0.82	–	1.253	**0.46**	88
ami49	49	408	39.053	14.22	165	38.650	17.24	67	–	13.75	–	**38.127**	**9.74**	202
n100	100	576	0.205	1.54	396	0.195	1.55	389	–	3.37	–	**0.192**	**1.25**	1425
n200	200	1274	0.207	3.34	1102	0.215	3.48	785	–	3.52	–	**0.200**	**3.10**	3603
n300	300	1632	0.329	5.44	2062	0.340	5.25	3767	–	**4.77**	–	**0.309**	**4.77**	8031

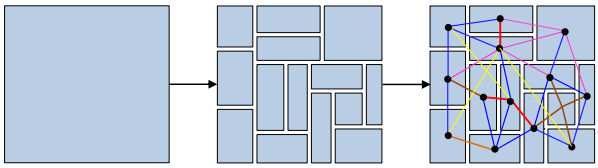


Fig. 4 An illustration of the proposed synthetic training dataset generation.

Since the optimization goal of floorplanning is to minimize the weighted sum of area and wirelength as defined in Equation (1), each netlist-floorplan pair in the training dataset should meet the requirements for both area and wirelength optimality. Note that the solution satisfies area optimality if there is no whitespace in a floorplan. Accordingly, to achieve wirelength optimality, closely placed macros are allocated to more connection degrees. In this case, swapping the position of any two macros will increase the wirelength, which has been proved in [9]. Fig. 4 depicts the generation of a netlist-floorplan pair in the dataset, where different coloured lines indicate different connection degrees.

The details of constructing the floorplanning dataset are summarized as follows. Firstly, given an empty chip region and the macro size constraint, n macros are produced by dividing the region horizontally and vertically without introducing any whitespace. Then, based on the macros' coordinates, the relative position relationships between any two macros are derived, which are represented by an SP. Note that the relative positions between macros are recorded as labels. Finally, the connection degree between any two macros is assigned according to the negative correlation of distances between macros. If the distance between two modules is too far, the connection degree is set to 0.

V. Experimental Results

A. Experimental Settings

The proposed end-to-end floorplanning framework is implemented in `Python` with the `Pytorch` library [16], and we execute it on a Linux server with one Nvidia GeForce RTX 3090 GPU. We pre-train the network model for one week on the synthetic training dataset containing 500000 netlists, and then fine-tune the model with the RL algorithm on the randomly generated netlists. In the end, the model is tested on MCNC [17] and GSRC [18] benchmarks.

B. Comparison with SOTA Works

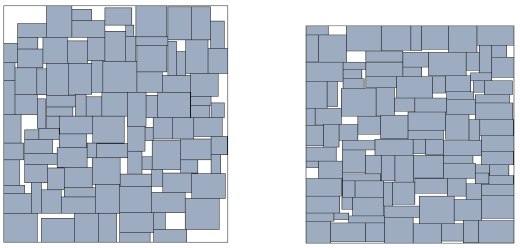
To verify the superiority of the proposed end-to-end floorplanning framework, in addition to comparing the Simulated Annealing (SA) algorithm, we also compare our framework with two state-of-the-art RL-based methods, including ICCD'20 [8] and KDD'22 [10]. For fair comparisons, all methods are conducted on the same computing platform. Besides, since the runtime of both SA and ICCD'20 depend on the termination conditions, we terminate the algorithms when the optimal value is not updated within a time step of ten times the total number of macros.

TABLE I lists the experimental results on MCNC and GSRC benchmarks. Columns "#Macro" and "#Net" represent the number of macros and nets in each benchmark circuit. Columns "Area" and "Wire" denote the area cost and the half-perimeter wirelength of the floorplan, while column "RT" is the average runtime of the algorithms. Because the area results are not reported in KDD'22, we only list the wirelength results. We emphasize the better results in bold. As TABLE I shows, the proposed floorplanning framework exhibits better area results in 4 cases out of 5 benchmark circuits. Moreover, the wirelength results show that our end-to-end floorplanning framework approach significantly outperforms all existing approaches by a large margin. For example, compared with SA, ICCD'20 and KDD'22, our framework reduces the wirelength by 18.4%, 23.3% and 29.6% on average. Although the framework's runtime is longer than non-end-to-end approaches, the runtime is negligible relative to the long-term design process of the entire chip design. And by using more GPUs, this time can be significantly reduced.

Fig. 5 depicts the floorplans of n100 circuit generated by our method and SA method, respectively. It can be seen that our method achieves a compact floorplan with a smaller area.

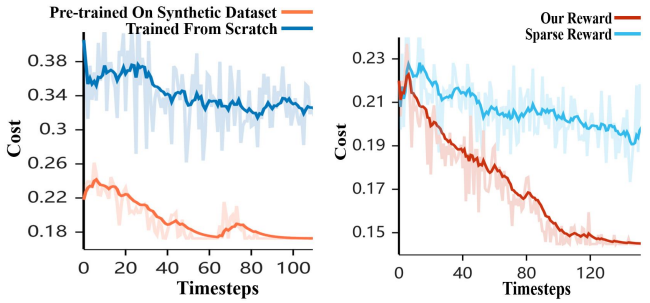
C. Ablation Studies

We first perform an ablation study to investigate how the proposed supervised learning process affects the performance. Fig. 6(a) compares two training strategies (i.e., training from scratch and training from a pre-trained network on the synthetic dataset) regarding the floorplan quality and the convergence speed on ami33 circuit. It can be seen that compared with training from scratch, the pre-trained network starts at a lower floorplan cost and finally produces a high-quality floorplan solution. Besides, the convergence speed increases by 40%. Furthermore, we also explore the impact



(a) area: $0.205 \times 10^6 \mu m^2$ (b) area: $0.192 \times 10^6 \mu m^2$

Fig. 5 Visualization of n100 floorplans from (a) SA algorithm; (b) the proposed end-to-end framework.



(a) (b)

Fig. 6 (a) Comparison of agents trained from scratch and pre-trained on synthetic dataset; (b) comparison between our designed reward and sparse reward.

TABLE II Impact of network components on performance (Area unit:$\times 10^6 \mu m^2$, Wire unit:$\times 10^5 \mu m$).

Circuit	GAT		MLP Policy		MLP Value		Ours	
	Area	Wire	Area	Wire	Area	Wire	Area	Wire
ami33	1.433	0.92	1.353	0.52	1.280	0.66	**1.253**	**0.46**
ami49	40.532	16.35	38.942	14.42	39.314	12.24	**38.127**	**9.74**
n100	0.226	1.82	0.201	1.54	0.197	1.44	**0.192**	**1.25**
n200	0.235	3.94	0.211	3.44	0.205	3.24	**0.200**	**3.10**
n300	0.342	5.92	0.332	5.13	0.317	5.22	**0.309**	**4.77**

of the proposed dense reward function on the performance. Fig. 6(b) shows the cost of floorplan solutions on ami49 circuit generated by the framework with the proposed reward design and the sparse reward method, respectively. Note that in the sparse reward method, the reward is set to 0 for all previous actions, and the final reward is a negative weighted sum of area and wirelength. The curves in Fig. 6(b) demonstrate the dense reward design can accelerate network convergence and achieve better floorplan results.

Another ablation study is performed to investigate how the network components affect the performance. We replace the proposed EGAT, bit-wise deep cross policy network, and gated self-attention value network with the graph attention network (GAT) [19] and two MLP networks, respectively. Note that when observing the impact of a specific network on performance, the other two networks remain unchanged. TABLE II lists the statistical results. It is evident that our proposed end-to-end network structure exhibits the best results, with each component contributing positively.

VI. Conclusion

In this work, we have proposed an innovative end-to-end RL-based floorplanning framework, mainly containing edge-augmented graph attention network and gated self-attention mechanism. An MDP model is built for floorplanning including novel representations of state, multi-action strategy, and dense reward function. To support the large-scale neural network training, we construct a synthetic netlist-floorplan dataset that meets the area and wirelength optimality. Experimental results show that, compared with the SOTA implementations, our approach produces better floorplan solutions.

References

[1] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "VLSI module placement based on rectangle-packing by the sequence-pair," *IEEE TCAD*, vol. 15, no. 12, pp. 1518–1524, 1996.

[2] K. Lei, P. Guo, Y. Wang, X. Wu, and W. Zhao, "Solve routing problems with a residual edge-graph attention neural network," *Neurocomputing*, vol. 508, pp. 79–98, 2022.

[3] 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.

[4] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. NIPS*, 2017, pp. 1–11.

[6] T.-C. Chen and Y.-W. Chang, "Modern floorplanning based on fast simulated annealing," in *Proc. ISPD*, 2005, pp. 104–112.

[7] C.-K. Cheng, A. B. Kahng, I. Kang, and L. Wang, "RePlAce: Advancing solution quality and routability validation in global placement," *IEEE TCAD*, vol. 38, no. 9, pp. 1717–1730, 2018.

[8] Z. He, Y. Ma, L. Zhang, P. Liao, N. Wong, B. Yu, and M. D. Wong, "Learn to floorplan through acquisition of effective local search heuristics," in *Proc. ICCD*, 2020, pp. 324–331.

[9] 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. DAC*, 2022, pp. 1303–1308.

[10] M. Amini, Z. Zhang *et al.*, "Generalizable floorplanner through corner block list representation and hypergraph embedding," in *Proc. KDD*, 2022, pp. 2692–2702.

[11] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[12] H. Guo, B. Chen, R. Tang, W. Zhang, Z. Li, and X. He, "An embedding learning framework for numerical features in CTR prediction," in *Proc. KDD*, 2021, pp. 2910–2918.

[13] R. Wang, R. Shivanna, D. Cheng, S. Jain, D. Lin, L. Hong, and E. Chi, "DCN V2: Improved deep & cross network and practical lessons for web-scale learning to rank systems," in *Proceedings of the Web Conference (WWW)*, 2021, pp. 1785–1797.

[14] J. Jumper, R. Evans, A. Pritzel, T. Green *et al.*, "Highly accurate protein structure prediction with AlphaFold," *Nature*, vol. 596, no. 7873, pp. 583–589, 2021.

[15] M. S. Hussain, M. J. Zaki *et al.*, "Global self-attention as a replacement for graph convolution," in *Proc. KDD*, 2022, pp. 655–665.

[16] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito *et al.*, "Automatic differentiation in PyTorch," in *NIPS Workshop*, 2017.

[17] "MCNC Benchmark," http://vlsicad.eecs.umich.edu/BK/MCNCbench.

[18] "GSRC Benchmark," http://vlsicad.eecs.umich.edu/BK/GSRCbench.

[19] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *Proc. ICLR*, 2018, pp. 1–12.