# Reinforcement Learning-based Auto-router considering Signal Integrity

Minsu Kim, Hyunwook Park, Seongguk Kim, Keeyoung Son, Subin Kim, Kyunjune Son, Seonguk Choi, Gapyeol Park and Joungho Kim
*School of Electrical Engineering*
*Korea Advanced Institute of Science and Technology*
Daejeon, South Korea
E-mail : min-su@kaist.ac.kr

***Abstract*—In this paper, we propose artificial intelligent (AI)-router, a reinforcement learning (RL)-based auto-router considering signal integrity (SI), for the first time. Our algorithm has two main stages. At first, we design the transformer-based novel neural architecture considering the keep-out region, crosstalk region, and the number of vias for SI optimization. Then, the designed neural network is optimized by the policy gradient, one of the RL algorithms. Compared with the conventional maze routers, the A* algorithm, and the lee algorithm, it is verified that our AI-router outperforms the algorithms in terms of wire-length and crosstalk in a specific test case. Furthermore, it is shown that AI-router successfully performs multi-layer routing which is not feasible with conventional maze routers.**

***Index Terms*—AI-router, reinforcement learning, signal integrity, transformer**

## I. Introduction

Recent technology trends require terabytes per second bandwidth. To meet these demands, not only the data rate but also channel density has been significantly increased. Moreover, high integration and miniaturization of modern electronic devices critically increase design complexity in channel routing. Therefore, automatic routing considering wire-length, crosstalk, via, and layer selection became a more challenging problem.

Previous researches on auto-routing mainly consist of two stages that global routing strategies and detailed routing [1]. Global routing strategies provide a feasible routing plan including routing order. After that, each pin-to-pin routing is completed by avoiding obstacles using a maze router. However, existing maze routers have two major limitations. Firstly, existing maze routers are not smart enough to perform detailed routing considering SI [2]-[3]. Because the maze-routers cannot consider crosstalk and vias, several hand-crafted heuristics have to be designed to control the maze router for ensuring SI which leads to increases entire algorithm's complexity. Secondly, existing maze-routers have the disadvantage of modifying the entire algorithm for solving the problem which contains additional constraints and optimization variables.

On the other hands, it has been proven several times that learning-based methods can be used universally in a variety of situations without major modifications to the algorithm. In particular, recent studies have suggested that reinforcement learning shows effective results in combinatorial optimization problems including routing and hardware design [4]-[5].

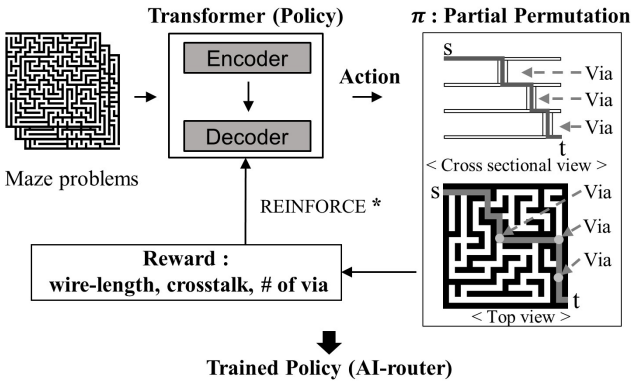


Fig. 1. Overview of training AI-router by RL. Randomly generated maze problems are training data for transformer-based encoder-decoder, which is a policy network. Then the policy network decides action which makes the partial permutation of input coordinate. The value of action is evaluated by the reward function which contains wire-length, crosstalk, and number of vias.

In this paper, we propose RL-based auto-router considering SI to overcome the limitation of the previous maze routers in terms of SI and versatility. We construct the encoder-decoder model based on the transformer, a state-of-the-art neural network in natural language processing and learning-based combinatorial optimization [5]-[6]. The policy gradient is used for training the encoder-decoder model [7]. For verification, the routing performance of the lee algorithm, the A* algorithm, and the proposed AI-router is compared in terms of wire-length and crosstalk [2]-[3]. Furthermore, we demonstrate that the proposed AI-router performs multi-layer routing which is infeasible to conventional maze routers.

## II. Proposal of Reinforcement Learning-Based AI-router

As shown in Fig. 1, a randomly generated maze problem is used as training data for the transformer-based encoder-decoder model. Through via, it is possible to move to another layer, then another maze problem of the changed layer is provided. RL is used for training policy networks which determines an optimal routing path. The trained policy network eventually becomes AI-router. Table I contains variables for the equations described in section II.A and II.B.

978-1-7281-6161-7/20/$31.00 ©2020 IEEE

TABLE I
EXPLANATION OF VARIABLES FOR EQUATIONS.

| Variables | Representation | Description |
|---|---|---|
| $X$ | $\{\mathbf{x} = (x, y, \gamma z, m_o, p_c, p_l)\}$ | Input coordinates |
| $m_0$ | $m_0 = \begin{cases} 1 & (x,y,z) \in O \\ 0 & otherwise \end{cases}$ | Obstacle mark |
| $p_c$ | $p_c = \begin{cases} 2 & (x,y,z) \in C \\ 0 & otherwise \end{cases}$ | Penalty of Xtalk |
| $p_l$ | $p_l = \begin{cases} 0.5 & (x,y,z) \in L \\ 0 & otherwise \end{cases}$ | Penalty of loosely Xtalk |
| $\gamma$ | $\gamma = 30$ | Layer changing penalty |
| $\pi$ | $\{\pi_1, \pi_2, ..., \pi_n\}$ , $x_{\pi_i} \in X$ | Output Permutation |
| $O$ | $\{(x_i, y_i, z_i)\}$ | Obstacle region |
| $C$ | $\{(x_i, y_i, z_i)\}$ | Xtalk region |
| $L$ | $\{(x_i, y_i, z_i)\}$ | Loosely Xtalk region |

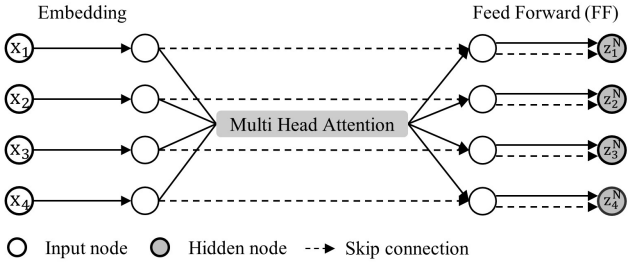


Fig. 2. Conceptual view of encoder architecture. Input nodes **x** are embedded to high dimensional vectors $z$ by multi-head attention, feed forward, and skip connection.

### A. Neural Architecture of AI-router

As illustrated in Fig. 2, the encoder architecture mainly consists of multi-head attention (MHA), feed forward (FF), and skip connection (Skip) which embed input node to high dimensional hidden nodes $z^N$. A detailed equation of the Skip, MHA, and FF is explained in [5]-[6].

First, the input node is embedded through the linear projection as the following equation:

$$z_i^0 = W\mathbf{x}_i + b \tag{1}$$

$W$ and $b$ are learnable parameters. Then, the hidden nodes $z^N$ are made from the embedded node with MHA as the following equation:

$$z_j^{i+1} = Skip_j(FF(Skip_j(MHA_j(z_1^i, z_2^i, ..., z_n^i)))) \tag{2}$$

Then the hidden nodes are propagated forward the decoder as illustrated in Fig.3. The main purpose of the decoder is to output the stochastic policy. To be specific, *context vector* $c$ creates a policy for the next selection, referring to the previous node, the average node of all, the average node of obstacles(keep-out region and pre-routed region), the average node of coupled crosstalk region, the average node of loosely-coupled crosstalk region and the destination(target pin) node:

$$c_k = F_{\theta_1}([z_{k-1}^N, z_n^N]) + F_{\theta_2}([z_{mean}^N, z_o^N, z_c^N, z_l^N]) \tag{3}$$

$F_{\theta_1}$ and $F_{\theta_2}$ is fully-connected layer with learnable parameter $\theta_1$,$\theta_2$. As shown in (5), the context vector passes MHA and becomes query $c'$.

$$c'_k = MHA(z_1^N, ..., z_n^N, c_k) \tag{4}$$

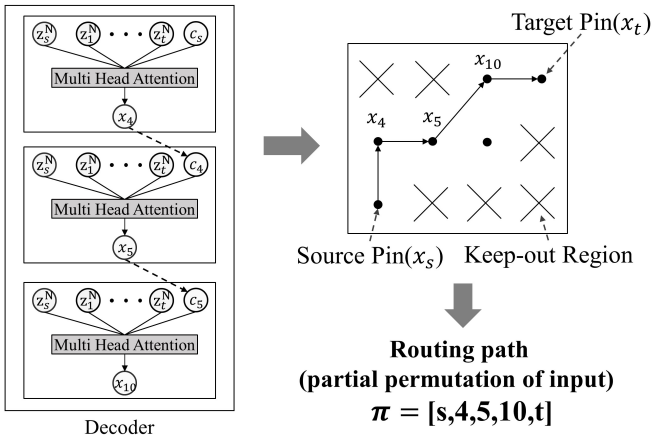


Fig. 3. Conceptual view of proposed decoder processing. The decoder uses the context vector $c$ and MHA to determine the next node. The determined node affects the selection of a next node by changing the context vector $c$. The decoder eventually creates a routing path of $\pi$.

Then the partial policy of selecting the next action can be expressed as follows:

$$u_j = \begin{cases} 10\tanh(\frac{(Qc')^t(Kz_j)}{4}) & nodes \in Action \\ -\infty & otherwise \end{cases} \tag{5}$$

$$P_\theta(\pi_i = j|X, \pi_{i-1}, \pi_n) = \frac{e^{u_j}}{\sum_{k=1}^n e^{u_k}} \tag{6}$$

Q and K in (5) are learnable parameters which make query and key, respectively. A detail of the query and key is introduced in [5]-[6]. The $Action$ in (5) contains left, right, up, down, 45-degree routing and layer changing through via. Also, obstacles are not possible to be selected by the $Action$.

Therefore the policy of routing permutation $\pi$ is

$$P_\theta(\pi|X) = \prod_{k=2}^{n-1} P_\theta(\pi_k|X, \pi_{k-1}, \pi_n) \tag{7}$$

### B. Training the AI-router with Reinforcement Learning

The policy gradient is used to train the AI-router architecture [7]. When we get permutation of nodes $\pi$, we calculate the cost of AI-router as follows:

$$L(\pi) = \sum_{k=1}^{n-1} \left(g(\mathbf{x}_{\pi_{k+1}} - \mathbf{x}_{\pi_k}) + p_c(\mathbf{x}_{\pi_k}) + p_{lc}(\mathbf{x}_{\pi_k})\right) \tag{8}$$

$$g(\mathbf{x}) = \sqrt{x^2 + y^2 + \gamma^2 z^2} \tag{9}$$

To estimate the crosstalk in the cost function $L(\pi)$, we set the crosstalk region and loosely crosstalk region at 1$w$ and 2$w$ distances based on the pre-routed channel, respectively. The $w$ is the width of the routing channel. In the end, the total crosstalk is estimated by counting the number of blocks that are invading crosstalk regions on a grid. A block in the grid is a path which is made by node selection from the $\mathbf{x}_{\pi_k}$ to $\mathbf{x}_{\pi_{k+1}}$ in (8).

Also, we assign $\gamma$ as a penalty for layer change. When changing a layer, it greatly interferes with the routing plan of other layers. Also, the via transition effect adversely affects SI. Therefore, layer change through via is not recommended

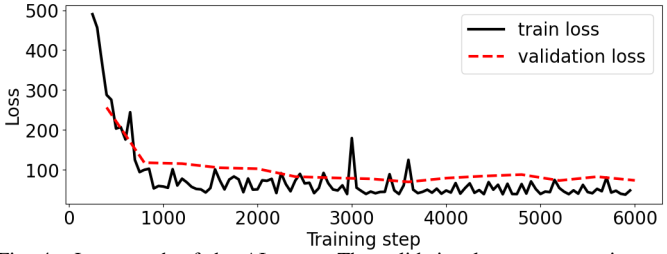


Fig. 4. Loss graph of the AI-router. The validation loss converges just as training loss.

TABLE II
RESULT OF THE TEST CASE IN FIG 5. BOLD MEANS THE BEST PERFORMANCE.

| Method | Wire-length | # of Xtalk region ($1w$) |
|---|---|---|
| Lee Algorithm | 3.8mm | 16 |
| A* Algorithm | **3.16mm** | 5 |
| AI-router (ours) | **3.16mm** | **0** |

except the case that routing is not feasible on a current layer, a high penalty of $\gamma = 30$ is given.

Eventually, the objective function is defined by the expectation of the cost $L(\pi)$ with an output probability of $p_\theta$ in (8). $\theta$ represents the whole parameters from the encoder-decoder architecture in II.A.

$$J(\theta|X) = \mathbb{E}_{p_\theta}[L(\pi)] \tag{10}$$

We optimize $\theta$ using the gradient of $J(\theta|X)$ with Adam optimizer, learning rate 0.00001 [8]. The gradient of the objective function can be derived as follows:

$$\nabla J(\theta|X) = \mathbb{E}_{p_\theta}[(L(\pi) - b(X))\nabla \log p_\theta(\pi|X)] \tag{11}$$

The $b$ in (11) is the rollout baseline for reducing variance [5].

## III. VERIFICATION OF THE PROPOSED METHOD

For training AI-router, 12800 randomly generated maze data are used. Each maze problem has a total of 800 nodes with two signal layers. Also, 2560 validation data are used to validate AI-router's learning-convergence and flexibility in various environments. As shown in Fig. 4, the validation loss of AI-router is converged just as it's train loss which means AI-router is stably trained.

To verify the trained AI-router's performance, we set two test cases. Firstly, routing performances of the lee algorithm, A* algorithm, and AI-router are compared in a single-layer test case. Secondly, the AI-router is tested on a multi-layer test case to verify AI-router's capability to perform multi-layer routing.

As illustrated in Fig. 5 and Table II, AI-router makes a better decision that makes shorter and less crosstalk routing than the Lee-router. Also, AI-router makes less crosstalk than the A* algorithm while preserving wire-length. Fig. 7 shows the results of the AI-router in a multi-layer routing test case. It can be observed that the AI-router finds the via location that minimizes crosstalk and wire-length by considering the two layers of keep-out regions and pre-routed regions.

## IV. CONCLUSION

In this paper, we proposed an RL-based AI-router considering wire-length, crosstalk, and vias. Our AI-router outperformed conventional maze routers in terms of SI in a

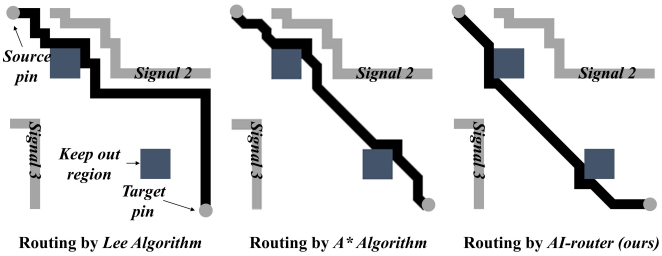


Fig. 5. Test case comparison between lee algorithm, A* algorithm, and AI-router. Because the keep-out region is blocking the middle between the source pin and target pin, the proposed AI-router chooses the routing to bypass downwards considering both crosstalk and wire-length.

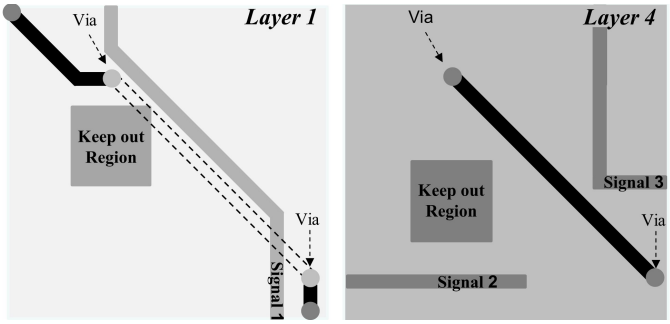


Fig. 6. Result of multi-layer routing by AI-router. Because signal 1 in layer 1 is blocking the path between the source pin and target pin, AI-router decides to change the signal layer through via. The AI-router decides via location considering the routing environment of layer 1 and layer 4.

specific test case. Because of the versatility of the learning-based method, our proposed algorithm can be further applied to various application targets by just fine-tuning the objective function of RL.

## ACKNOWLEDGEMENT

We would like to acknowledge the technical support from ANSYS Korea. This work was supported in part by the National Research Foundation of Korea Grant funded by the Korean Government (NRF-2019M3F3A1A03079612)

## REFERENCES

[1] J. McDaniel, Z. Zimmerman, D. Grissom, and P. Brisk, "Pcb escape routing and layer minimization for digital microfluidic biochips," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 1, pp. 69–82, 2017.

[2] C. Y. Lee, "An algorithm for path connections and its applications," *IRE Transactions on Electronic Computers*, vol. EC-10, no. 3, pp. 346–365, 1961.

[3] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[4] H. Park, J. Park, S. Kim, K. Cho, D. Lho, S. Jeong, S. Park, G. Park, B. Sim, S. Kim, Y. Kim, and J. Kim, "Deep reinforcement learning-based optimal decoupling capacitor design method for silicon interposer-based 2.5-d/3-d ics," *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 10, no. 3, pp. 467–478, 2020.

[5] W. Kool, H. van Hoof, and M. Welling, "Attention, learn to solve routing problems!" in *International Conference on Learning Representations*, 2019.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30*, 2017, pp. 5998–6008.

[7] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, pp. 229–256, 1992.

[8] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations*, 2019.