A Appendix

We provide more discussion on the classic solvers for placement and routing in Sec. A.1. We further design and implement an RL-based router (regarded as our implementation of the router [1]) for additional comparison with our generative router in Sec. A.2, which shows its efficacy.

A.1 Related Work on Classic Solvers for Placement and Routing

Classic Solvers for Placement. Global placement solvers have been studied for a long history [2, 3], as basically in three categories: partitioning-based methods, stochastic/hill-climbing methods, and analytic solvers. Partition-based approaches utilize the divide-and-conquer strategy in the early years, after which multi-level partitioning algorithms [4] are devised, with many stochastic methods using annealing [5]. When regarding to analytic solvers, force-directed algorithms [6] and non-linear optimizers [7, 8] are extensively adopted. Modern analytical placers, e.g. ePlace [9] and RePlAce [10], have recently introduced an electrostatics-based system that contains global-smooth density cost function and nonlinear optimizers, which are accelerated by DREAMPlace [11] later based on deep neural network. Mixed-size placement attracts attention for its practical value. Hierarchical [12] and constraint graph-based [13] are proposed to place large scale mixed size designs.

Classic Solvers for Routing. According to the way for solving global routing, there are two types of classic methods: concurrent and sequential solvers. Concurrent approaches attempt to handle numerous nets simultaneously. BoxRouter [14] develops progressive integer linear programming (ILP) and adaptive maze routing to diffuse the congestion. BoxRouter 2.0 [15] further provides a more systematic way of eliminating congestion and assigning layers to wires. GRIP [16, 17], built on a partitioning strategy in a 3D manner, obtains the ideal wirelength but leads to prohibitive runtime. While sequential approaches typically employ net decomposition [18], maze routing [19], pattern routing [20], or negotiation-based rip-up and rerouting (NR&R), and solely route a 2-pin net every time (e.g. [21, 22, 23, 24]). The sequential methods have been often shown faster than the concurrent approaches, but they may rely on the ordering of the nets and thus leading to sub-optimal solutions.

A.2 Implementation of an RL-based Router

We further devise an additional RL-based router following [1] which only gives limited result on small-scale dataset that is generated randomly additional routing methods for comparison.

The RL-based router in our experiment is based on recent work [1] that decides actions of the routing direction, i.e. going north, south, etc in each step. In our implementation, we enlarge the grid size from 8×8 to 16×16 while ignoring different layers in routing, as we concentrate on calculating the final wirelength in more realistic cases. The structure of our Q-network consists of three fully-connected layers with 128 hidden units in each layer. Reward is set as -1 for all valid actions, except that the reward of finding terminal pin is 50 and a penalty of -50 is supposed to pay for invalid action which makes the capacity of routing path smaller than zero.

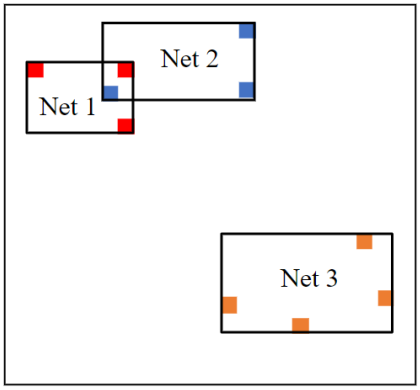


Figure 1: Pins of $net1$, $net2$ and $net3$ are dyed red, blue and orange, respectively. The block that exactly surrounds each net is the bounding box of the net. The bounding box of $net1$ overlaps with the bounding box of $net2$, while the bounding box of $net3$ is independent of the above two.

Table 1: Loss function w.r.t Correctness Rate (CrrtR) and Wirelength Ratio (WLR) on Route-small dataset (80K samples). We use our conditional generative network as the base model. "L2" is short for L2 loss, "FL" is short for focal loss, and "EL" denotes the enhanced loss in Eq. 3.

Loss	Route-small	
	CrrtR↑	WLR↓
L2	0.648	1.020
FL	0.756	1.091
L2+FL	0.674	1.035
EL	0.735	1.018

A.3 Additional Experimental Results

A.3.1 Ablation Study for Loss Function of the Routing Model

In this experiment, we utilize our conditional generative routing network as the base model and verify the effectiveness of the enhanced loss function. Our adversarial loss is

$$\mathcal{L}_{adv}(G,D) = \sum_{i=1,2} \lambda_i \left(\mathbb{E}_{x,y} \left[\log D_i(x,y) \right] + \mathbb{E}_x \left[\log(1 - D_i(x, G(x))) \right] \right), \tag{1}$$

where we set $\lambda_1 = \lambda_2 = 0.5$ by default. While our focal loss is

$$\mathcal{L}_{FL}(G) = -\mathbb{E}_{x,y} \left[\frac{1}{N} \sum_{i=1}^{N} \alpha \left[y_i (1 - g_i)^{\gamma} \log g_i + (1 - y_i) g_i^{\gamma} \log(1 - g_i) \right] \right], \tag{2}$$

where we set $\alpha = 0.5$ and $\gamma = 2$ by default. For the enhanced loss, it can be expressed as

$$\min_G \left(\left(\max_D \mathcal{L}_{adv}(G,D) \right) + \mu_{FL} \mathcal{L}_{FL}(G) + \mu_{L2} \mathcal{L}_{L2}(G) + \mu_r \mathcal{L}_r(G) \right), \tag{3}$$

where we set $\mu_{FL} = \mu_{L2} = 5 \times 10^3$ and $\mu_r = 0.1$ in all our experiments.

In Table 1, focal loss delivers the best correctness rate while introducing higher wirelength. In contrast, the L2 loss is opposite of the focal loss. The enhanced loss combines the advantages of these two loss functions and obtains a competitive correctness rate and the best wirelength ratio.

A.3.2 Input-size-Adapting vs. Non-Input-size-Adapting

Table 2: Correctness Rate (CrrtR) and Wirelength Ratio (WLR) on Route-large dataset (100K 128×128 samples). "Non-ISA" is the non-input-size-adapting network. "ISA" is the input-size-adapting network, and "0/50/100" refer to the amount of pre-training epochs for the filling network.

Variants	Params	Route-large	
		CrrtR↑	WLR↓
Non-ISA	15M	0.657	1.023
ISA-0	12M	0.777	1.011
ISA-50	12M	0.784	1.010
ISA-100	12M	0.780	1.013

We explore the performance difference between our input-size-adapting network (represented by ISA) and a non-input-size-adapting variant (represented by Non-ISA). The Non-ISA follows the structure of our basic generator. Its convolutional front-end contains one more layer, and it has three additional residual blocks. The ISA utilizes the well-trained basic generator in part as the guiding network to obtain a feature map, and it further uses another convolutional front-end as the filling network to acquire another feature map. Then the element-wise sum of the feature maps are fed into a series of residual blocks and a decoder to generate the output. We compare the Non-ISA with 3 variants of our ISA (without pre-training the filling network, pre-training the filling network for 50 epochs and pre-training the filling network for 100 epochs). The ISA outperforms the Non-ISA as illustrated in Table 2, while the variants pre-trained for different number of epochs behave similarly.

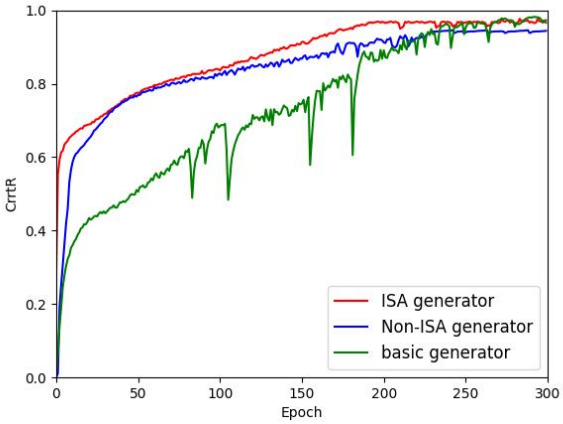


Figure 2: Training curves of the Non-ISA generator, the ISA generator and the basic generator. The variants of the ISA pre-trained for different amounts of epochs hold similar training curves.

Table 3: Evaluation of different backbones w.r.t. correctness rate (CrrtR) and wirelength ratio (WLR) for the routing on: Route-small-extension. cGAN: the vanilla cGAN model with a single realness discriminator; bcGAN: the bi-discriminator version. EL: enhanced loss in Eq. 3.

our router w/ different generative models	Route-small–extension	
	CrrtR↑	WLR↓
CVAE*(CNN) [25]	0.597	1.073
CVAE*-cGAN(CNN)	0.535	1.081
CVAE*-bcGAN(CNN)	0.647	1.023
U-Net* [26]	0.728	1.014
cGAN(U-Net*) [27]	0.572	1.077
bcGAN(U-Net*)	0.731	1.022
ResNet [28]	0.743	1.273
cGAN(ResNet)	0.632	1.017
bcGAN(ResNet)	0.757	1.010
bcGAN(ResNet)+EL (full version of our router)	**0.795**	**1.007**

Fig. 2 shows the training plot for the Non-ISA, the ISA and the basic generator. The variants of the ISA pre-trained for diverse amounts of epochs share similar training curves. The training of all the networks is stable, and the ISA converges faster than the Non-ISA and achieves a better result since it partly inherits the well-trained basic generator and the filling network has also been pre-trained.

A.3.3 Additional Experiments of Generative Backbones

We extend the dataset Route-small to a larger dataset named Route-small-extension that contains 650k instances from 3 circuits as the training set and 200k samples from another circuit as the test set. We conduct additional experiments of generative routing models on Route-small-extension with results shown in Table 3, which is an extension to Table 2 in the main text.

A.3.4 Additional Experiments of Classical Router

We secure source code of Labyrinth 1.1 [20] and run it on our machine, then we obtain the results in Table 4. We set $Num\_Reroute$ as 500 and two-terminal as $True$. All experiments are run on a AMD Ryzen 5 4600H CPU with 2G RAM. Circuit $ibm05$ is a trivial case due to the sufficient routing resources, so that it requires fewer iterations in the rip-up and reroute phase and runtime is relatively shorter.

[1] https://kastner.ucsd.edu/ryan/labyrinth-a-global-router-and-routing-development-tool

Table 4: Evaluation of wirelength (WL), overflow (OF) and runtime (time) for Labyrinth 1.1 on ISPD-98 routing benchmarks. The experiments are performed on our machine with a AMD Ryzen 5 4600H CPU and 2G RAM. Source code is obtained from [1].

Circuits	WL↓	OF ↓	Time(s)↓
ibm01	74413	292	12.3
ibm02	199687	384	49.5
ibm03	183923	122	39.0
ibm04	197405	1124	47.6
ibm05	427303	0	19.0
ibm06	345583	502	96.7

Table 5: Comparison w.r.t wirelength (WL), overflow (OF) and runtime (time) for our conditional generative routing model and RL-based router on circuit $adaptec1$ from ISPD-2005 benchmark. Note the maximal grid size that RL router can deal with is 16×16, so the positions of macros for both methods are scaled to $[0, 16)$ for a fair comparison.

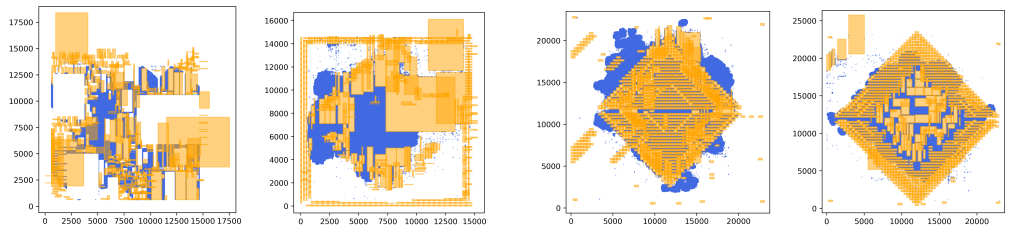
Method	WL↓	OF ↓	Time(s)↓
Generative Router (ours)	5240	0	**1.805**
RL Router	4951	157	11.528

A.3.5 Comparison with RL-based Router

Table 5 compares the results between our conditional generative routing model and RL-based router on circuit $adaptec1$ from ISPD-2005 benchmark. Our conditional generative router is much more efficient than the RL-based router, achieving approximately $10\times$ speedup. In addition, the overflow of RL-based router is greater than zero, which degrades the performance of wirelength in the long run. Note the maximal grid size that RL router can deal with is 16×16 while our generative routing model is still applicable for grids larger than 64×64. Such limitation on size and extremely low efficiency make RL-based router not an option in neural macro placement and routing pipeline.

A.3.6 Additional Visualization of Mixed-size Placement

Despite circuit $bigblue1$, two additional visualization of our mixed-size placer and DeepPlace on circuits $adaptec2$ and $adaptec4$ are elaborated in Fig. 3.



(a) DeepPlace on $adaptec2$. (b) Our mixed-size placer on $adaptec2$. (c) DeepPlace on $adaptec4$. (d) Our mixed-size placer on $adaptec4$.

Figure 3: Visualization of macro (in orange) /standard cell (in blue) placement by DeepPlace [29] and our mixed-size placer on circuits $adaptec2$ and $adaptec4$. On circuit $adaptec4$, the density of macros makes it difficult to fill standard cells into the center of canvas for DeepPlace, while our mixed-size placer eliminates the problem via reducing overlap.

References

[1] H. Liao, W. Zhang, X. Dong, B. Poczos, K. Shimada, and L. Burak Kara, "A deep reinforcement learning approach for global routing," *Journal of Mechanical Design*, 2020.

[2] M. A. Breuer, "A class of min-cut placement algorithms," in *DAC*, 1977.

[3] C. M. Fiduccia and R. M. Mattheyses, "A linear-time heuristic for improving network partitions," in *DAC*. IEEE, 1982.

[4] A. R. Agnihotri, S. Ono, and P. H. Madden, "Recursive bisection placement: Feng shui 5.0 implementation details," in *ISPD*, 2005.

[5] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, 1983.

[6] P. Spindler, U. Schlichtmann, and F. M. Johannes, "Kraftwerk2—a fast force-directed quadratic placement approach using an accurate net model," *TCAD*, 2008.

[7] 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," *TCAD*, 2008.

[8] A. B. Kahng and Q. Wang, "Implementation and extensibility of an analytic placer," *TCAD*, 2005.

[9] J. Lu, P. Chen, C.-C. Chang, L. Sha, D. J.-H. Huang, C.-C. Teng, and C.-K. Cheng, "eplace: Electrostatics-based placement using fast fourier transform and nesterov's method," *TODAES*, 2015.

[10] C.-K. Cheng, A. B. Kahng, I. Kang, and L. Wang, "Replace: Advancing solution quality and routability validation in global placement," *TCAD*, 2018.

[11] Y. Lin, Z. Jiang, J. Gu, W. Li, S. Dhar, H. Ren, B. Khailany, and D. Z. Pan, "Dreamplace: Deep learning toolkit-enabled gpu acceleration for modern vlsi placement," *TCAD*, 2020.

[12] T. Taghavi, X. Yang, and B. Choi, "Dragon2005: Large-scale mixed-size placement tool," in *ISPD*, 2005.

[13] H. Chen, Y. Chuang, Y.-W. Chang, and Y. Chang, "Constraint graph-based macro placement for modern mixed-size circuit designs," in *ICCAD*, 2008.

[14] M. Cho and D. Z. Pan, "Boxrouter: A new global router based on box expansion and progressive ilp," *TCAD*, 2007.

[15] M. Cho, K. Lu, K. Yuan, and D. Z. Pan, "Boxrouter 2.0: A hybrid and robust global router with layer assignment for routability," *TODAES*, 2009.

[16] T.-H. Wu, A. Davoodi, and J. T. Linderoth, "Grip: Scalable 3d global routing using integer programming," in *DAC*, 2009.

[17] T.-H. Wu, A. Davoodi, and J. T, Linderoth, "A parallel integer programming approach to global routing," in *DAC*. IEEE, 2010.

[18] C. Chu and Y.-C. Wong, "Flute: Fast lookup table based rectilinear steiner minimal tree algorithm for vlsi design," *TCAD*, 2007.

[19] C. Y. Lee, "An algorithm for path connections and its applications," *IRE Transactions on Electronic Computers*, 1961.

[20] R. Kastner, E. Bozorgzadeh, and M. Sarrafzadeh, "Pattern routing: Use and theory for increasing predictability and avoiding coupling," *TCAD*, 2002.

[21] M. D. Moffitt, "Maizerouter: Engineering an effective global router," *TCAD*, 2008.

[22] M. M. Ozdal and M. D. Wong, "Archer: A history-based global routing algorithm," *TCAD*, 2009.

[23] H.-Y. Chen, C.-H. Hsu, and Y.-W. Chang, "High-performance global routing with fast overflow reduction," in *ASP-DAC*. IEEE, 2009.

[24] W.-H. Liu, W.-C. Kao, Y.-L. Li, and K.-Y. Chao, "Nctu-gr 2.0: Multithreaded collision-aware global routing with bounded-length maze routing," *TCAD*, 2013.

[25] D. Utyamishev and I. Partin-Vaisband, "Late breaking results: A neural network that routes ics," in *DAC*. IEEE, 2020.

[26] R. O, F. P, and B. T, "U-net: Convolutional networks for biomedical image segmentation," in *MICAAI*, 2015.

[27] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *CVPR 2017*, pp. 5967–5976, 2017.

[28] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[29] R. Cheng and J. Yan, "On joint learning for solving placement and routing in chip design," *NeurIPS*, 2021.