DPRoute: Deep Learning Framework for Package Routing

Yeu-Haw Yeh*, Simon Yi-Hung Chen**, Hung-Ming Chen*, Deng-Yao Tu**, Guan-Qi Fang**, Yun-Chih Kuo**, Po-Yang Chen**
*Institute of Electronics, National Yang Ming Chiao Tung University, Hsinchu, Taiwan
**Mediatek Inc., Hsinchu, Taiwan
leafleaf.ee09@nycu.edu.tw,SimonYH.Chen@mediatek.com,hmchen@mail.nctu.edu.tw,devin-dy.tu@mediatek.com,guan-qi.fang@mediatek.com,eric-yc.kuo@mediatek.com,po-yang.chen@mediatek.com

ABSTRACT

For routing closures in package designs, net order is critical due to complex design rules and severe wire congestion. However, existing solutions are deliberatively designed using heuristics and are difficult to adapt to different design requirements unless updating the algorithm. This work presents a novel deep learning-based routing framework that can keep improving by accumulating data to accommodate increasingly complex design requirements. Based on the initial routing results, we apply deep learning to concurrent detailed routing to deal with the problem of net ordering decisions. We use multi-agent deep reinforcement learning to learn routing schedules between nets. We regard each net as an agent, which needs to consider the actions of other agents while making pathing decisions to avoid routing conflict. Experimental results on industrial package design show that the proposed framework can improve the number of design rule violations by 99.5% and the wirelength by 2.9% for initial routing.

CCS CONCEPTS

• **Hardware** → **PCB design and layout**; • **Computing methodologies** → *Machine learning*.

KEYWORDS

substrate routing, deep learning, multi-agent reinforcement learning

ACM Reference Format:
Yeu-Haw Yeh*, Simon Yi-Hung Chen**, Hung-Ming Chen*, Deng-Yao Tu**, Guan-Qi Fang**, Yun-Chih Kuo**, Po-Yang Chen**. 2023. DPRoute: Deep Learning Framework for Package Routing. In *28th Asia and South Pacific Design Automation Conference (ASPDAC '23), January 16–19, 2023, Tokyo, Japan.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3566097.3567902

1 INTRODUCTION

Routing is one of the most time-consuming stages in the package design flow due to the irregular placement of bumps and a large number of nets. With the increasing design rules and varied package design, a general routing technique is urgently needed. So far, most existing routers follow a sequential strategy to resolve routing conflict between nets [7, 9]. [7] adopted the CDT-based routing model to solve the routing problem considering irregular bump assignment and improve the congestion issue by the rip-up-and-reroute process. [9] used the A*-based routing algorithm and handled the net order problem by pushing the routed nets. However, the performance of such sequential routing is sensitive to the orders of nets to be routed. Therefore, [5] proposed an integer linear programming model to complete the routing task. Besides, [2] divided the package design into concentric circles and concurrently routed it from the inner ring to the outer ring. Although these methods address the net ordering problem, these hand-crafted heuristics algorithms may not be possible to be applied in practical package designs. At the same time, a lot of bending is produced, which causes high resistance [1].

In this work, a deep learning routing framework is proposed to find a general routing strategy without heuristic bias. Based on the ring model-based initial routing results, we formulate concurrent detailed routing as a multi-agent deep reinforcement learning (MARL) task [15] for asynchronous routing planning between nets. We regard each net as an agent, which needs to consider the actions of other agents while making pathing decisions to avoid routing conflict. Our contribution can be summarized as follows:

- We redefine the routing area and shrink the routing problem by dividing the entire design into non-overlapping boxes.
- We apply deep learning to concurrent detailed routing algorithms to decide net ordering rather than a heuristic decision.
- The proposed routing framework for industrial package designs demonstrates that our solution can significantly improve the number of design rule violations (DRVs), wirelength and layout pattern.

In the rest of the paper, we use the terms agent and net interchangeably. We thus organize this work as follows. Section 2 explains the background of routing design rules and problem formulation. Section 3 details overall proposed routing framework. Section 4 and 5 present the detailed routing algorithms. Section 6 describes the method to refine routing layout. Section 7 reports the experimental results. Finally, section 8 concludes the paper.

2 PRELIMINARIES

2.1 Design Rules in Package Design

In this section, we describe the design rules in our package cases. Package nets can be routed horizontally, vertically, and diagonally on the same layer, but a sharp angle is not allowed [4]. We build the grid-based routing graph, and the surrounding five tiles of the node

ASPDAC '23, January 16–19, 2023, Tokyo, Japan

ACM ISBN 978-1-4503-9783-4/23/01…$15.00
https://doi.org/10.1145/3566097.3567902

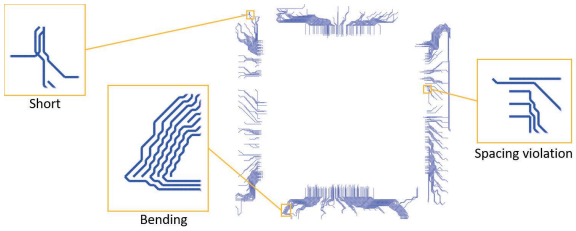


Figure 1: Suboptimal routing layout using the heuristic algorithm on industrial package design.

are next candidate steps. Figure 2 (a) shows the possible directions of routing path. Compared with digital designs, package designs have fewer layers; thus, we need to prevent net crashing cautiously. Besides, package designs have the minimum spacing constraint. We set the tile width as the minimum spacing. The spacing constraint will never be violated when only routing horizontally and vertically. The violation may happen if we allow nets to route diagonally. Any segment near the diagonal segment would cause its spacing to be less than one tile. Figure 2 (b) shows the design rule violation because the spacing between two segments is less than one tile. Figure 2 (c) is acceptable because of the large spacing. We handle this by disallowing nets to be routed on both sides of the diagonal segment.

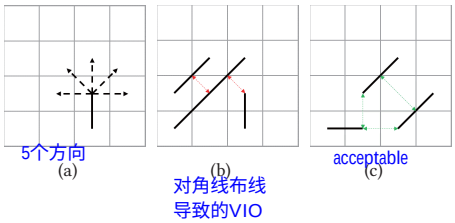


Figure 2: Examples of design rules. (a) Possible five directions from the segment. (b) Spacing rule violation. (c) Violation free.

2.2 Problem Formulation

In this paper, the positions of bumps and balls are determined by the initial setting. Bumps are in the top layer, and balls are in the bottom layer. Each signal needs to be connected from the specified bump to the ball through vias connecting different layers.

Some notations are defined in this paper. After initial routing, we divide the initial result into a set of boxes $S_b = \{b_i | 1 \le i \le M\}$, where M is the number of all boxes. Each box contains a set of nets $S_n = \{n_j | 1 \le j \le N_i\}$, where N_i is the number of nets in box b_i. Information of net n_j includes routing path π_j, source s_j, and target t_j. The problem formulation can be described as follows:

- Inputs are initial settings, including design rules such as spacing constraints and width constraints, die size, a set of signal nets, and the position of bumps and balls.
- Output is a refined routing result.
- The objective is to complete the routing task and minimize wirelength, the number of wire bends, and DRVs.

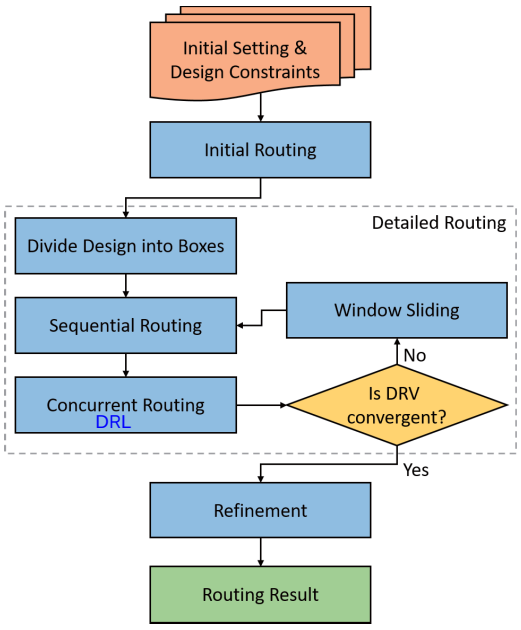


Figure 3: The routing framework.

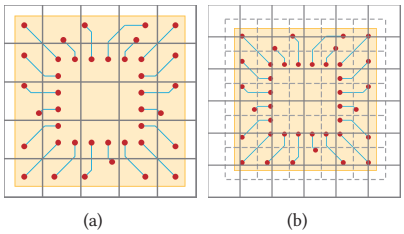


Figure 4: (a) The example of the whole design division. (b) The dotted grid is the division before window sliding, and the solid-line grid is the division after window sliding.

3 OVERALL ROUTING FRAMEWORK

Figure 3 is the flow of the proposed substrate routing framework, composed of three main stages: initial routing, detailed routing, and refinement. Initial routing first generates the coarse result. Then, detailed routing and refinement are responsible for optimizing wirelength, the number of bends, and the number of DRVs. In the initial routing stage, we can use any routing result as our starting point. Here we choose the global routing result generated by [2] as the initial result. This routing result considers detouring and the routibility but ignores the number of bends and allows design rule violations.

We cannot be sure that the initial routing result has no design rule violation. Thus detailed routing would make an effort to reduce the number of DRVs. This stage can be divided into three small steps. First, we divide the design into routing boxes and the splitting way demonstrated in Figure 4 (a). These boxes would be routed initially by decided net order. If any box fails to be routed, the concurrent router will handle the remaining task. The concurrent router based

on deep learning can ignore the net order issue and provide other routing solutions. Because routing box-by-box is weak to process the junctions of boxes, we slide the window like Figure 4 (b) and execute these steps several times. Window sliding redefines routing regions, which helps routers focus on previous boundaries of boxes. In addition, window sliding has a higher chance of making original congested areas routable. We route and slide the window repeatedly until all optimization objectives converge. Finally, in the refinement stage, we apply an A*-based algorithm to adjust wirelength and bending and output the routing result.

4 SEQUENTIAL ROUTING

In this section, we present our sequential routing algorithm for the detailed routing stage. Section 4.1 introduces the buildup of the routing graph and the method of routing path finding in congested areas. Section 4.2 details the algorithm and explains the net order selection.

4.1 Repel Routing

As mentioned earlier, a whole design would be divided into boxes. The advantage is that we can process every box independently and merge them directly in the future. We build the grid-based routing graph on these boxes separately and extract the source and target of nets. These sources and targets are the terminals of the signals or the nodes split by boxes. After building up the graph, the initial routing result becomes a lot of routing problems for 2-pin nets.

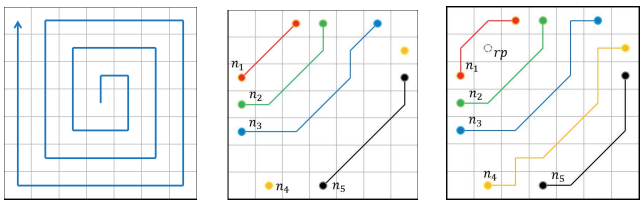


Figure 5: Illustrations of the routing strategy. (a) is the sequence of repulsion point assignments, starting from the inner ring and ending in the outer ring. (b) Net order is $\{n_1, n_2, n_3, n_5, n_4\}$ and paths are found by A* search. n_4 has no path to connect at the end. (c) Net order is $\{n_5, n_4, n_3, n_2, n_1\}$ and paths are found by proposed cost function. Routing paths are prone to apart from repulsion point rp, and all nets are actually routed.

If the box includes a few nets, it can be solved easily by the A* algorithm. However, when the number of nets increases, the box would become congested, A* algorithm is hard to handle, just like Figure 5 (b). To keep the balance between wirelength and the routibility, we apply the cost function defined as follows:

$$COST(n_i, c, p, f) = \alpha \times L(n_i, \pi_c) + \beta \times B(\pi_c) + \gamma \times R(\pi_c, p, f), \tag{1}$$

where n_i is information of net i, including positions of source and target, π_c is the path from source to point c, $L(n_i, \pi_c)$ is the wirelength estimation function which is based on A* search, $B(\pi_c)$ is the number of bends of path π_c, α, β, and γ are user-defined parameters, $R(\pi_c, p, f)$ is the repulsion function, which is described as here:

$$R(\pi_c, p, f) = \frac{f \times \sum_{e \in \pi_c} DIST^{-1}(p, e)}{LEN(\pi_c)}, \tag{2}$$

where edge e belongs to path π_c, $DIST(p, e)$ is the distance between point p and edge e, f is the routing failure number, $LEN(\pi_c)$ is the length of path π_c. The repulsion function can calculate the average repulsion between path π_c and point p. For (2), moving segments of the paths away from the repulsion point reduces the cost, while (1) balances path length and fights against the long paths by $L(n_i, \pi_c)$. f is set to zero initially. As the number of failures increases, the repulsion function will dominate the cost function. If point p is placed in the center of the box, paths are prone to route near the periphery of the box. The repulsion function benefits routing space utilization. Figure 5 (c) shows the effect of the repulsion function, the repulsion point is assigned to the top-left corner. According to (2), keeping segments of nets away from the repulsion point lets the cost in (1) decrease. As a result, all routing paths are nearby the boundary or other paths.

4.2 Implementation Details

High-density boxes are sensitive to parameters and net order because of the narrow solution space. Repulsion point position and net order would directly impact the routing space utilization; thus, both determinations are essential concerns. The proposed routing algorithm is summarized in Algorithm 1.

We divide the box into concentric circles and initially place repulsion point rp in the inner circle. After several rounds of routing, if the failure number exceeds upper bound U_f, the repulsion point will be moved from the inner ring to the outer one until the box is successfully routed. The sequence of point assignments is illustrated in Figure 5 (a), and all candidate positions in the same ring would be put into repulsion point queue Q_{rp}.

If the A* algorithm can not handle the box, the net sequence S_n will be rearranged by the distance between the nets and the repulsion point. The net farthest from the repulsion point is closest to the box boundary, and same-side nets tend to route toward its direction. We route the farthest one first to prevent other nets from blocking it.

5 CONCURRENT ROUTING

In this section, we first detail the background of multi-agent reinforcement learning. Then, we define the state space, action space, reward function, and the basic setup. In the end, we explain the dedicated MARL techniques for our routing problem.

5.1 Multi-Agent Reinforcement Learning (MARL)

Multi-agent reinforcement learning enables N agents to learn a joint policy by simultaneously interacting with the environment to maximize the collaborative cumulative reward. We model the multi-net routing RL problem as decentralized partially observable Markov decision progress (Dec-POMDP) [11]. Given the true state $s \in S$ of environment, each agent $a \in A \equiv \{1, \ldots, N\}$ takes an action $u^a \in U$ at every timestep concurrently, forming a joint

Algorithm 1 Sequential Routing

Input: Grid G, Set of sources S_s, Set of targets S_t
Output: Set of paths S_π

Sort S_n by net length
Reset G
for each s_i, t_i in S_n **do**
  $\pi_i = REPEL\_ROUTING(G, s_i, t_i, 0, 0)$
  if $t_{\pi_i} \neq t_i$ **then**
    break
  end if
end for
if all $t_{\pi_i} = t_i$ **then return** S_π 全都接上直接就结束了
end if
for r = inner ring to outer ring **do** r从内到外
  Put all r positions into Q_{rp}
  while $|Q_{rp}| \neq 0$ **do**
    $rp = POP\_FRONT(Q_{rp})$
    Sort S_n by distance between net and rp
    for $f = 1$ to U_f **do** Uf: upper bound of failure num
      Reset G
      for each s_i, t_i in S_n **do**
        $\pi_i = REPEL\_ROUTING(G, s_i, t_i, rp, f)$
        if $t_{\pi_i} \neq t_i$ **then**
          break
        end if
      end for
    end for
  end while
end for
return S_π

action $\mathbf{u} \in \mathbf{U} \equiv U^a, \forall a \in \{1, \ldots, N\}$. The state s stochastically transits to the next state s', determined by transition probability function $P(s'|s, \mathbf{u}) : S \times \mathbf{U} \times S \mapsto [0, 1]$. Each agent has independent observation $z \in Z$, determined by observation function $f(s, \mathbf{u}) : S \times \mathbf{U} \mapsto Z$. The agent performs a policy $\pi(u^a, \tau^a) : T \times U \mapsto [0, 1]$ based on its observation history $\tau^a \in Z \times U$. All agents share same reward function $r(s, \mathbf{u}) : S \times \mathbf{U} \mapsto \mathbb{R}$ and $\gamma \in [0, 1)$ is the discount factor. As shown in (3), the objective is to learn a joint policy $\pi = \langle \pi^1, \ldots, \pi^N \rangle$ that maximizes the joint expected cumulative reward.

$$J(\pi) = \mathbb{E}_{u^1 \sim \pi^1, \ldots, u^N \sim \pi^N} \left[\sum_{t=0}^{\infty} \gamma_t r_t \left(s_t, u_t^1, \ldots, u_t^N \right) \right] \tag{3}$$

In this work, centralized training and decentralized execution (CTDE) [8] is applied to the learning process. We define each agent as a path planning actor for an independent net. Actors need to perform actions jointly while avoiding routing conflicts and maximizing the shared reward from the environment. As shown in Figure 6 (a), actors are trained online in a centralized way, in which the learning algorithm can access all local action observations, global states, and sharing gradients and parameters. Actors and critics are training simultaneously to learn the policy and value functions. The policy function tells the agent how to make decisions, and the value function helps actors to identify important decisions in routing. However, the critic is only used in the training process to output the advantage A to tell actors the difference between the actual reward an agent sees and the expected reward at some point in the episode. In the execution stage, each individual agent can only access its local action observation history from the environment.

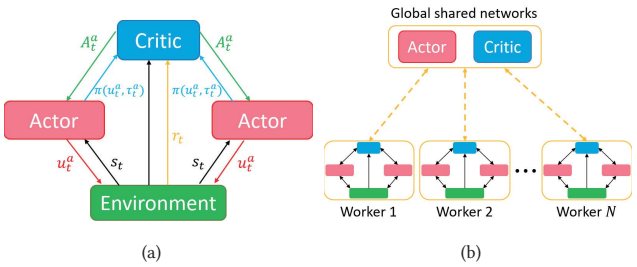


Figure 6: (a) The multi-agent reinforcement learning framework. Actors (agents) and critics are trained simultaneously to learn the policy function and value function. (b) The asynchronous advantage actor-critic framework.

5.2 Basic MARL Setup

In this section, we define state space, action space, and reward function as follows:

State space: A state s is the collective representation of features for all agents, and each agent can observe five features. The first feature is the vector of available actions formatted by one-hot encoding. The second feature is the information of the observed agent, including wirelength, current position, and target position. The third feature is other agents' information, including information on the second feature, relative distance, and last action. The fourth feature is the space occupancy information in the environment. The final feature is the number of times routed so far.

Action space: Given an environment state, a set of agents jointly execute actions from ten options, including "Stop," "North," "East," "South," "West," "Northeast," "Southeast," "Southwest," and "Northwest." 9个？

Reward: Given the joint actions of agents, the environment will transit from the current state to the next state and provide feedback. Agents receive common rewards according to the following reward function:

$$r = \sum_{i}^{N} C_i + S, \tag{4}$$

where C_i is the ratio of the Euclidean distance between two pins of the net and the routing length, S is defined as a cooperative reward term when all nets are routed successfully.

5.3 Central Actor-Critic

In this section, we describe the MARL algorithm used in this work. As shown in Figure 6 (b), Central Actor-Critic [12] is based on the asynchronous advantage actor-critic (A3C) [10] method with multiple actors running in parallel. The difference is that Central Actor-Critic shares a similar structure with COMA's critic [6]. It takes $(s_t, \mathbf{u_{t-1}})$ as the input and outputs $V(s)$. Besides, to deal with

Algorithm 2 Central Actor-Critic

1: Initialize actor network $\pi^{\vartheta}(u|s)$ and critic network
2: $V_{\pi}^{\phi}(s)$ with weights ϑ, ϕ from global shared networks
3: **for** *episode* $= 1, 2, \ldots, M$ **do**
4: Initialize environment e, and receive initial observation
5: state s_0 from e
6: **for** $t = 1, 2, \ldots, T$ **do**
7: **for** *each agent* a **do**
8: Sample action $u^a \sim \pi(u^a|\tau^a)$
9: **end for**
10: Execute joint action u and observe reward r and next
11: state s_{t+1} from e
12: Set TD target $y_t = r + \gamma V_{\pi}^{\phi}(s_{t+1})$
13: Update critic by minimizing loss: $\delta_t = \left(y_t - V_{\pi}^{\phi}(s)\right)^2$
14: Update actor policy by minimizing loss:
15: $-\sum_a \log \pi(u^a|\tau^a)\left(Q(s,\mathbf{u}) - V_{\pi}^{\phi}(s)\right)$
16: Update $s_t \leftarrow s_{t+1}$
17: **end for**
18: **end for**

COMA's unstable training due to its inability to predict accurate counterfactual action-value for un-taken actions, our actors follow the simple TD advantage policy gradient [13] given by:

$$\nabla_{\theta} J = \mathbb{E}_{\pi}\Big[\sum_{a} \nabla_{\theta} \log \pi(u^a|\tau^a)\big(Q(s,\mathbf{u}) - V(s)\big)\Big], \tag{5}$$

where $Q(s,\mathbf{u}) = r + \gamma V(s')$. We detail how actor and critic are updated in Algorithm 2. Before learning starts, each agent takes a copy of the global shared actor network with parameter θ in lines 1-2 and then conditions a stochastic policy until a terminal state is reached. After the gradients of actor and critic are computed in lines 12-15, the global shared networks are updated asynchronously.

6 REFINEMENT

Detailed routing considers the routibility and aims to reduce the number of DRVs. It is possible to refine the routing result even further. We refine all nets in the box separately by the A*-based algorithm. Besides, we apply the layout discriminator to skip well-routing areas. We collect some package designs from the industrial database and tile them into boxes with no overlapping pixels at native magnification. All boxes are annotated by factors such as routing wirelength, the number of bends, and the layout pattern. We leverage ImageNet-pretrained EfficientNet [3, 14] with data augmentation to find suboptimal routing boxes. During the training and testing, augmentation methods (e.g., rotation and transpose) generate diverse routing data to mitigate overfitting problems and improve the inference results.

7 EXPERIMENTAL RESULTS

7.1 Model Performance

All experiments are obtained by using Nvidia RTX 1080 Ti graphic cards. Figure 7 shows the improvement in the capabilities of the deep learning model as the training progresses. For MARL, the agent network consists of a gate recurrent unit (GRU) with a 64-dimensional hidden layer with a fully connected (FC) layer before and after. The architecture of the critic network is a feedforward fully-connected network composed of 2 FC layers with 128 units and a FC layer with 1 unit. Figure 8 shows a series of MARL-based concurrent routing processes. After training, the agents can jointly allocate the routing space occupied by each other and learn to make good use of stopping motions to avoid potential routing conflicts.

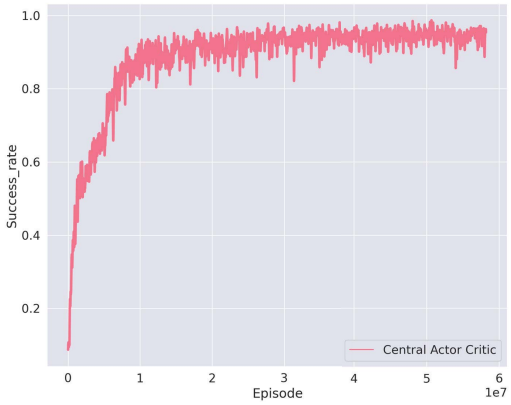
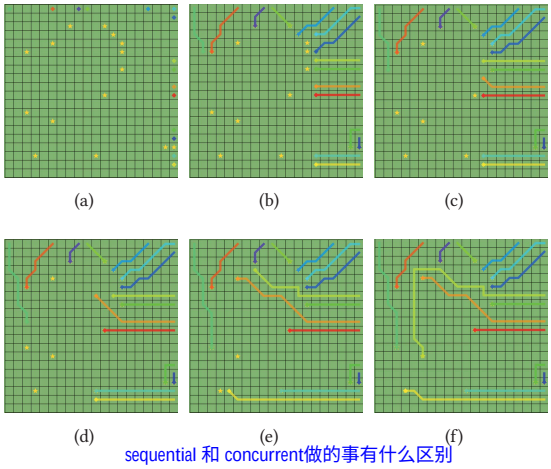


Figure 7: The performance of deep neural networks. The capacity of the central actor-critic is strong enough after about 60,000,000 episodes.



sequential 和 concurrent做的事有什么区别
现在有拆了重新布线？

Figure 8: Visualization of concurrent multi-net routing between different time steps. It presents the situation of cooperation between each net and the significance of the "stop" action. The Square symbolizes starting point, and the star indicates the destination. (a) is the initial routing state. In (a)-(c), every net routes concurrently. In (c)-(d), the yellow net selects the "stop" action to let the orange net routing continue. (f) is the final state where each net arrives the destination successfully.

Table 1: Comparison of routing results

	Die size(μm^2)	# Nets	Initial routing result [2]				Final result			
			# DRVs	WL(μm)	# Bends	Runtime(s)	# DRVs	WL(μm)	# Bends	Runtime(s)
Case1	6399×6339	381	69	561557	3827	3.06	0	557006	1985	539.26
Case2	10820×9960	327	121	654874	4570	6.52	1	624639	1857	642.59
Ratio	-	-	1.000	1.000	1.000	-	0.005	0.971	0.458	-

只有两个case？

7.2 Routing Results

To evaluate our work, we apply the proposed method to two industrial cases. Figure 9 illustrates the routing results of two different stages. Zooming in to observe, we can find some wire bends in Figure 9 (a) become smooth in Figure 9 (b). Table 1 lists the number of DRVs, wirelength, and the number of bends in three different stages. Compared with the initial routing result [2], detailed routing almost cleans all design rule violations and achieves a 2.7% improvement in wirelength and 51.3% improvement in the number of bends. With the refinement method, the overall flow achieves a 2.9% improvement in wirelength and 54.2% improvement in the number of bends even further. Experimental results show that the proposed routing framework has the capability of completing the industrial routing task.

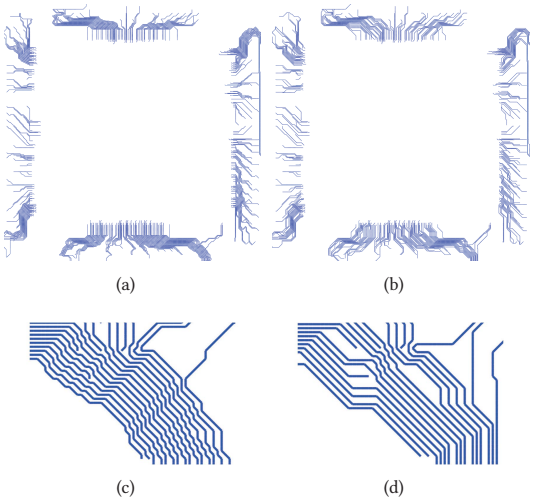


Figure 9: Illustration of Case2 routing results. (a) Initial result. (b) Final result. (c) Enlarged view of nets with many bends in (a). (d) Enlarged view of improved result in (b).

8 CONCLUSION

In this paper, we propose a novel routing framework leveraging deep learning. Based on the initial result, the package design is tiled into boxes. We then route each box to generate the detailed routing result while decreasing the number of design rule violations and wirelength. Our detailed routing algorithms can be divided into sequential-based and concurrent-based routing models. For the former, we apply the repulsion to the routing algorithm. It is helpful for dealing with congested boxes. For the latter, we transform concurrent routing into a cooperative pathfinding problem between networks. The net observes the state of the environment and chooses action options that maximize future rewards to avoid routing conflicts. Furthermore, our proposed framework can continuously evolve to adapt to different designs by accumulating design data. Experimental results show that the proposed framework can be successfully applied to industrial package design cases with almost no DRV and achieve a 2.9% improvement in wirelength and 54.2% improvement in the number of bends from the initial routing result.

REFERENCES

[1] Mustafa Celik, Lawrence Pileggi, Larry Pileggi, and Altan Odabasioglu. 2002. *IC interconnect analysis.* Springer Science & Business Media.
[2] Hao-Yu Chi, Simon Yi-Hung Chen, Hung-Ming Chen, Chien-Nan Liu, Yun-Chih Kuo, Ya-Hsin Chang, and Kuan-Hsien Ho. 2022. Practical substrate design considering symmetrical and shielding routes. *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)* (2022), 951–956.
[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition.* 248–255. https://doi.org/10.1109/CVPR.2009.5206848
[4] Jia-Wei Fang, Kuan-Hsien Ho, and Yao-Wen Chang. 2008. Routing for chip-package-board co-design considering differential pairs. In *2008 IEEE/ACM International Conference on Computer-Aided Design.* IEEE, 512–517.
[5] Jia-Wei Fang, Chin-Hsiung Hsu, and Yao-Wen Chang. 2008. An integer-linear-programming-based routing algorithm for flip-chip designs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 28, 1 (2008), 98–110.
[6] Jakob N. Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2017. Counterfactual Multi-Agent Policy Gradients. *CoRR* abs/1705.08926 (2017). arXiv:1705.08926 http://arxiv.org/abs/1705.08926
[7] Jyun-Ru Jiang, Yun-Chih Kuo, Simon Yi-Hung Chen, and Hung-Ming Chen. 2020. On pre-assignment route prototyping for irregular bumps on BGA packages. *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)* (2020), 1311–1314.
[8] Landon Kraemer and Bikramjit Banerjee. 2016. Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing* 190 (2016), 82–94. https://doi.org/10.1016/j.neucom.2016.01.031
[9] Shenghua Liu, Guoqiang Chen, Tom Tong Jing, Lei He, Tianpei Zhang, Robi Dutta, and Xian-Long Hong. 2009. Substrate topological routing for high-density packages. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 28, 2 (2009), 207–216.
[10] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous Methods for Deep Reinforcement Learning. *CoRR* abs/1602.01783 (2016). arXiv:1602.01783 http://arxiv.org/abs/1602.01783
[11] Sylvie CW Ong, Shao Wei Png, David Hsu, and Wee Sun Lee. 2009. POMDPs for robotic tasks with mixed observability. In *Robotics: Science and systems*, Vol. 5. 4.
[12] Jianyu Su, Stephen Adams, and Peter A Beling. 2021. Value-decomposition multi-agent actor-critics. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 11352–11360.
[13] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction.* MIT press.
[14] Mingxing Tan and Quoc V. Le. 2019. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *CoRR* abs/1905.11946 (2019). arXiv:1905.11946 http://arxiv.org/abs/1905.11946
[15] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* 575, 7782 (2019), 350–354.