Ranking Cost: Building An Efficient and Scalable Circuit Routing Planner with Evolution-Based Optimization

Shiyu Huang[1] , **Bin Wang**[2] , **Dong Li**[2] , **Jianye Hao**[2] , **Ting Chen**[1] and **Jun Zhu**[1]

[1]Tsinghua University
[2]Huawei Noah's Ark Lab
hsy17@mails.tsinghua.edu.cn, {wangbin158,lidong106,haojianye}@huawei.com,
{tingchen,dcszj}@tsinghua.edu.cn

Abstract

Circuit routing has been a historically challenging problem in designing electronic systems such as very large-scale integration (VLSI) and printed circuit boards (PCBs). The main challenge is that connecting a large number of electronic components under specific design rules involves a very large search space. Early solutions are typically designed with hard-coded heuristics, which suffer from problems of non-optimal solutions and lack of flexibility for new design needs. Although a few learning-based methods have been proposed recently, they are typically cumbersome and hard to extend to large-scale applications. In this work, we propose a new algorithm for circuit routing, named as Ranking Cost, which innovatively combines search-based methods (i.e., A* algorithm) and learning-based methods (i.e., Evolution Strategies) to form an efficient and trainable router. In our method, we introduce a new set of variables called cost maps, which can help the A* router to find out proper paths to achieve the global objective. We also train a ranking parameter, which can produce the ranking order and further improve the performance of our method. Our algorithm is trained in an end-to-end manner and does not use any artificial data or human demonstration. In the experiments, we compare with the sequential A* algorithm and a canonical reinforcement learning approach, and results show that our method outperforms these baselines with higher connectivity rates and better scalability.

1 Introduction

As described in Moore's Law [Schaller, 1997], the number of transistors in a dense integrated circuit (IC) increases exponentially over time and the complexity of chips becomes higher and higher. Such high complexity makes the IC design a time-consuming and error-prone work. Thus more capable automatic design systems, such as electronic design automation (EDA) tools, are needed to improve the performance. In the flow of IC designs, we need to find proper paths to place wires which connect electronic components on ICs, and

these wires need to achieve expected connectivity under certain constraints. One of the most important constraints is that wires on the same layout should not intersect. Besides, to reduce the signal propagation delay, the wire-length should be minimized. This is a critical and challenging stage in the IC design flow [Hu and Sapatnekar, 2001], known as circuit routing, which has been studied by lots of researchers [Kramer, 1984; He and Bao, 2020].

Circuit routing involves a large number of nets (a net is a set of vertices or pins with the same electrical property) to be routed, which is computationally expensive and makes manual design extremely time-consuming [Coombs and Holden, 2001]. Even under the simplest setting, where only two pairs of pins need to be routed, it is an NP-complete problem [Kramer, 1984]. Although lots of circuit routing algorithms have been proposed [Zhang, 2016], there remain two major challenges: (1) Early solutions [Hu and Sapatnekar, 2001] are typically designed with hard-coded heuristics, which suffer from problems of non-optimal solutions [Zhang, 2016] and lack of flexibility over new design needs. Therefore, a more powerful routing method that does not depend on domain knowledge is highly desired. (2) Although a few learning-based methods have been proposed [Liao *et al.*, 2020; He and Bao, 2020] recently, their methods are hard to extend to large-scale applications. In real settings, there are lots of components and nets on a single chip, which shows greater demand for the scalability of routing algorithms.

To relieve these problems, we propose a novel algorithm, denoted as Ranking Cost (RC), for circuit routing. We innovatively combine search-based methods (i.e., A* algorithm, the most commonly used algorithm in circuit routing tasks) and learning-based methods (i.e., Evolution Strategies [Salimans *et al.*, 2017], a powerful black-box optimization method) to form a trainable router with proper parametrization. Our method is flexible for integrating new constraints and rules so long as they can be merged into a global objective function. Moreover, our method is an one-stage algorithm, which optimizes the global objective function directly. In our method, we introduce a new set of variables called cost maps, which can help the A* routers to find out proper paths to achieve the global objective. We also train a ranking parameter, which can produce the ranking order and further improve the performance of our method.
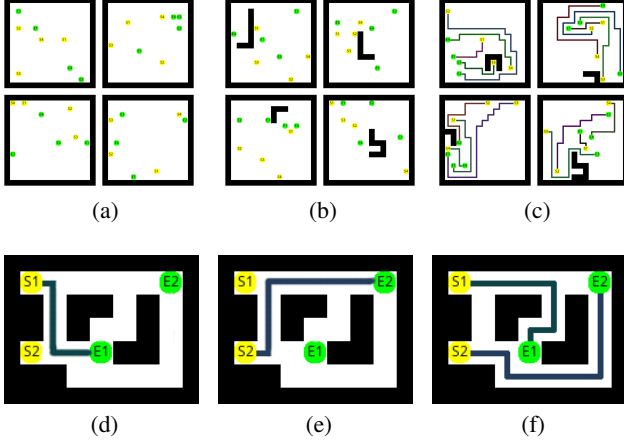
Figure 1: Yellow vertices are the start vertices, green vertices are the end vertices and black blocks are obstacles where paths can not pass through. (a) Examples of maps without obstacle. (b) Examples of maps with obstacles. (c) Circuit routing results derived by our algorithm. In (d)-(f), $(S_i, E_i), i \in \{1, 2\}$ are two pairs of vertices which need to be connected. (d) and (e) show that sequential A* algorithm will greedily find the shortest path of one pair, and it fails to obtain a global solution. (f) shows the solution found by our algorithm, which can connect the two pairs successfully.

In the experiments, we compare our method with the commonly used A* method and a canonical reinforcement learning approach, and results show that our method outperforms these baselines with higher connectivity rates. Our ablation study also shows that the trained cost maps can capture the global information and lead to a reasonable routing solution. We also show that our method is scalable to larger maps. However, recent learning-based methods [Liao *et al.*, 2020; He and Bao, 2020] only conduct experiments on small maps.

2 Related Work

We summarize the related work on circuit routing.

2.1 Heuristic Algorithms for Circuit Rouging

The routing problem can be heuristically separated into two stages, the first being the global routing step [Mo *et al.*, 2001; Cho *et al.*, 2007], followed by detailed routing [Ho *et al.*, 1991; Mandal *et al.*, 2020; Chen and Chang, 2009]. On the one hand, there are multiple heuristic-based approaches for global routing including regionwise routing [Hu and Sapatnekar, 2001], force-directed routing [Mo *et al.*, 2001], and rip-up and reroute [Cho *et al.*, 2007]. Besides, [Shi *et al.*, 2016] proposes a method to improve the distribution of congestion in global routing, and [Shi and Davoodi, 2017] propose a framework to analyze local congestion for global routing. On the other hand, the most commonly used detailed routing algorithms are channel routing and its variants [Ho *et al.*, 1991; Mandal *et al.*, 2020], which decompose the routing region into routing channels and generate wires in these channels [Chen and Chang, 2009]. One main issue of the two-stage methods is that these two stages do not always co-

ordinate well [Shi and Davoodi, 2017], which results in enormous difficulty in joint optimization. Instead, our method is a one-stage algorithm and new design constraints can be simply involved in the objective function without changing the algorithm itself.

A more straightforward strategy for circuit routing is to select a specific order and then route nets sequentially, e.g., sequential A* algorithm and Lees algorithm [Huang *et al.*, 2014; Malavasi and Sangiovanni-Vincentelli, 1993]. The major advantage of this type of approaches is that the congestion information for previously routed nets can be taken into consideration while routing the current one. However, the drawback of these sequential approaches is that the quality of the solution is very sensitive to the orders [Zhang, 2016]. Moreover, earlier routed paths only focus on finding their own best solutions and are impossible to take into account the situation of subsequent paths. Such greedy strategies may make a solvable circuit routing problem insolvable. Figure 1 shows an example that the sequential A* algorithm will fail to handle. In this example, there are two pairs of points to be routed, i.e., we should connect start vertices S_i and end vertices $E_i, i \in \{1, 2\}$, respectively. If we connect the pair (S_1, E_1) first in its shortest path, it will make (S_2, E_2) disconnected as shown in Figure 1(d), and vice versa (Figure 1(e)). But this case can be solved easily by our algorithm as shown in Figure 1(f), which means that our algorithm can take into account the global information when generating each path.

3 Background and Notations

In this section, we formalize the circuit routing problem first and then give a brief introduction to the OpenAI-ES algorithm [Salimans *et al.*, 2017].

3.1 Circuit Routing

Circuit routing is a path search problem, where the goal is to find non-intersecting paths that connect an arbitrary number of pairs of vertices. It can be formalized as a grid graph $G = (V, E)$, where each vertex $v_i \in V$ represents an intersection on the grid, and each edge $e_{ij} \in E$ represents the path between v_i and its 1-hop neighbor $v_j \in V$. And the non-routable obstructive vertices form an obstacle set, denoted as $O \subset V$. In circuit routing, a net $N = \{v_{n1}, v_{n2}, ...\} \subset V$ is a set of vertices that need to be connected and a multi-vertex net can be decomposed into multiple two-vertex nets via a minimum spanning tree (MST) or a rectilinear Steiner tree (RST) [de Vincente *et al.*, 1998; Hu and Sapatnekar, 2001]. Following [He and Bao, 2020], we simplify this problem by letting each net only contain two vertices, i.e, a two-vertex net is defined as $N = \{v_s, v_e\}$ with a start vertex v_s and an end vertex v_e. The net should be connected by a path $P = [v_1, v_2, ..., v_n]$ with $v_s = v_1$, $v_e = v_n$ and $P \cap O = \varnothing$. We use $|P|$ to represent the length of the path P. Given a set $\mathcal{N}$ of nets $\{N_i\}$, we need to find a set $\mathcal{P}$ of paths $\{P_i\}$ that connect these nets. A reasonable routing plan for a given set of nets is that all the paths do not share any vertex. In some cases, the routing problem does not have a solution because such a non-intersecting set $\mathcal{P}$ does not exist. In circuit routing, the commonly used objective is minimizing

the total length of paths. Given a solution $\mathcal{P}$, the total length of paths is defined as $L = \sum_{P \in \mathcal{P}} |P|$.

3.2 Evolution Strategies

To optimize the parameters in our algorithm, we need a powerful black-box optimization method. Actually, evolution Strategies (ES) is a class of black-box optimization methods [Rechenberg, 1994; Hansen and Ostermeier, 2001]. Recently, an evolution strategy variant, referred to as the OpenAI-ES [Salimans *et al.*, 2017], has attracted attention because it could rival the performance of modern deep reinforcement learning methods on multiple control tasks. In the OpenAI-ES, there is a reward function, denoted as $F(\psi)$, where ψ is a solution vector, sampled from a probability distribution function $\psi \sim p_\theta(\psi)$. The goal is to maximize the expected value $J(\theta)$:

$$\max_\theta J(\theta) = \mathbb{E}_{\psi \sim p_\theta}[F(\psi)]. \tag{1}$$

We can use the gradient ascent to optimize θ:

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta), \tag{2}$$

where α is the learning rate. In the OpenAI-ES, a score function estimator is used to calculate the gradient, which is similar to REINFORCE [Williams, 1992]:

$$\nabla_\theta J(\theta) = \nabla_\theta \mathbb{E}_{\psi \sim p_\theta}[F(\psi)] = \mathbb{E}_{\psi \sim p_\theta}[F(\psi) \nabla_\theta \log p_\theta(\psi)]. \tag{3}$$

Usually, p_θ is an isotropic multivariate Gaussian with mean θ and fixed covariance $\sigma^2 I$. And the expected value can be rewritten as $\mathbb{E}_{\psi \sim p_\theta}[F(\psi)] = \mathbb{E}_{\epsilon \sim \mathcal{N}(0,I)}[F(\theta + \sigma\epsilon)]$. Thus the gradient estimator changes to:

$$\begin{aligned} \nabla_\theta J(\theta) &= \nabla_\theta \mathbb{E}_{\epsilon \sim \mathcal{N}(0,I)}[F(\theta + \sigma\epsilon)] \\ &= \frac{1}{\sigma} \mathbb{E}_{\epsilon \sim \mathcal{N}(0,I)}[F(\theta + \sigma\epsilon)\epsilon], \end{aligned} \tag{4}$$

where ϵ is sampled from a standard normal distribution. Once we form an objective function $J(\theta)$, the gradient $\nabla_\theta J(\theta)$ can be approximated via Eq. (4) and parameters θ can be updated via Eq. (2).

4 Methodology

In this section, we first introduce the Ranking Learning algorithm and then the Cost-Map Learning algorithm. At last, we give out our final algorithm (Ranking Cost) and its training strategy.

4.1 Sequential A* with Ranking Learning

The A* algorithm can be used to find the shortest path in a two-vertex net $N = \{v_s, v_e\}$, which is considered more efficient than breadth-first search. A* finds a path between a start vertex v_s and an end vertex v_e, with a priority function defined as:

$$s(v) = g(v) + h(v), \tag{5}$$

where v is a vertex in the graph, $g(v)$ represents the total length from v_s to the current vertex v, and $h(x)$ represents the future heuristic cost from v to the end vertex v_e. The $h(x)$ is often set to the Euclidean distance or Manhattan distance. In a circuit routing problem, there are k nets to be routed.

Given a specific ranking order R, we can apply the A* algorithm to each net sequentially. For example, if there are 3 nets $\{N_1, N_2, N_3\}$ to be routed and a ranking order $R = (2, 1, 3)$ is given. We can apply A* to these 3 nets following the order $N_2 \rightarrow N_1 \rightarrow N_3$. After the routing procedure, we can get a total path length from this ranking order, denoted as $L(R)$. Different ranking orders will result in different total lengths. We can go through all the ranking orders and get their corresponding total lengths. The ranking order with minimum total length can be used as the final solution. However, this is unacceptable in time complexity because a routing problem with k nets has $O(k!)$ different ranking orders.

To relieve the problem of combinatorial explosion of ranking orders, we propose a method to learn the ranking order. We define a k-dimension ranking parameter $\theta_r = (\beta_1, \beta_2, ..., \beta_k)$. And the ranking order is determined completely by the ranking parameter. More concretely, the order of values in θ_r is exactly the routing order, and the net with the largest value will be routed first and the rest may be deduced by analogy. For example, given a ranking parameter $\theta_r = (0.5, 0.2, 0.4)$, we have $\beta_1 > \beta_3 > \beta_2$ and the corresponding ranking order will be $R_{\theta_r} = (1, 3, 2)$. And the ranking parameter will determine the final routing result and also the total path length. We define a reward function over the ranking parameter:

$$F(\theta_r) = -L(R_{\theta_r}), \tag{6}$$

where $L(R_{\theta_r})$ is the total length when taking the order R_{θ_r}, and less length leads to larger reward. As described in Section 3.2, we define the expected value over $F(\cdot)$ as:

$$J(\theta_r) = \mathbb{E}_{\psi \sim p_{\theta_r}} F(\psi) = \mathbb{E}_{\epsilon \sim \mathcal{N}(0,I)} F(\theta_r + \sigma\epsilon). \tag{7}$$

The gradient $\nabla_{\theta_r} J(\theta_r)$ can be estimated via Eq. (4), and θ_r can be updated via Eq. (2). At every training step, we sample a fixed number of noise vectors from Gaussian, add them to the ranking parameter and then evaluate all the new ranking parameters. The reward function $F(\cdot)$ will return a group of scalar rewards and the ranking parameter will be updated with these rewards. In this way, we can train a ranking parameter θ_r to approach the minimal total length L.

As mentioned in Section 2.1, a sequential routing algorithm may make a solvable circuit routing problem insolvable. And even a method with a learned ranking order will still suffer from this problem. In the following section, we will introduce how our method could overcome this problem by coupling the A* algorithm with cost maps.

4.2 Circuit Routing with Cost Maps

As introduced in the previous section, $g(v)$ in Eq. (5) represents the total length from start vertex v_s to current vertex v, thus the A* algorithm can only search for the shortest path of the current net and it cannot take into account the information of following paths. Our goal is to add global information to the A* routers so that these routers can cooperate to achieve their common objective. First, for each net $N_i, i \in \{1, ..., k\}$, we define its cost map as $C_i = (c_1, c_2, ..., c_m) \in \mathbb{R}^m$, where $m = |V|$ is the number of vertices in graph G. Therefore, there is a total of k cost maps, and all the cost maps can be learned with our algorithm. We apply a cost-map function

$C(i, v) = c, c \in C_i$ to simplify the notation. In this way, we can reformulate the A* algorithm by adding the cost maps:

$$s(v) = g(v) + \sum_{i=1}^{k} C(i, v) + h(v), \quad (8)$$

where $g(v)$ and $h(v)$ are with the same definitions as in Eq. (5). The cost-map enhanced A* algorithm reveals two key points: (1) When removing $g(v)$ and $h(v)$ from Eq. (8), it will degenerate into a pure learning-based algorithm and the path will be determined completely by the cost function. (2) When removing the cost functions or set them to zero (i.e., $C(i, v) = 0$), it is a pure search-based algorithm. Hence, it is a method which combines the searched-based algorithm and the learning-based algorithm and could take advantage of both sides. Firstly, the A* search makes our method easier to find a connected path in a complex environment, while it is hard for learning-based methods to find a connected path [Tamar *et al.*, 2016]. Second, the global information can be merged into cost maps and the global objective can be approached by tuning the cost maps.

To learn the cost maps, we first define a cost-map parameter $\theta_c \in \mathbb{R}^{k \times m}$ and the values in cost maps are defined as $c_j = \max(0, \theta_c[i, j]), c_j \in C_i$. Given a ranking order R and cost maps, we can route the nets sequentially using the cost-map enhanced A* and a total path length L will be obtained from the routing result. Similar to Eq. (6), we define a reward function over the cost-map parameter and also its the expected value:

$$\begin{aligned} F(\theta_c) &= -L(R, C_{\theta_c}), \\ J(\theta_c) &= \mathbb{E}_{\psi \sim p_{\theta_c}} F(\psi) = \mathbb{E}_{\epsilon \sim N(0,I)} F(\theta_c + \sigma\epsilon), \end{aligned} \quad (9)$$

where C_{θ_c} presents the cost maps derived from θ_c, and θ_c can be updated via Eq. (2). By iteratively executing the A* search and cost-map learning step, our method could solve some hard problems that a sequential routing algorithm can not solve as shown in Figure 1. In the next section, we will introduce how cost-map learning can couple with ranking learning and how to train the algorithm efficiently in practice.

4.3 Ranking Cost Algorithm

Our final algorithm, denoted as Ranking Cost (RC), can learn the ranking parameter θ_r and cost-map parameter θ_c jointly. In Eq. (8), all the cost maps are used when calculating the priority function and the ranking order has no impact on the cost maps. To learn the cost-map with ranking order, we change the priority function to:

$$s_j(v) = g_j(v) + \sum_{i=j+1}^{k} C(i, v) + h_j(v), \quad (10)$$

where $s_j(v)$ is the priority function for the net whose ranking order is the j-th. This modification is reasonable because the earlier routed nets will use more cost maps and observe more global information. For example, when routing the first net, its ranking index is $j = 1$ and it will use all the rest $k - 1$ cost maps. As a result, the first routed path focuses more on its impact on other unrouted nets. When routing the last net,

its ranking index is $j = k$ and no cost map will be used. As a result, the last net is routed via a normal A* algorithm and the searched solution is the shortest path at the current situation. As all the previous nets have been routed, the best way to route the last net is just to find out the shortest path. Our RC algorithm achieves this naturally and makes it flexible for arbitrary ranking orders. When the ranking order changes, the order and the usage of cost maps change accordingly. In this way, the reward function can be defined as:

$$F(\theta_r, \theta_c) = -L(R_{\theta_r}, C_{\theta_c}), \quad (11)$$

and both θ_r and θ_c can be updated via the OpenAI-ES algorithm. Actually, the reward function F can be integrated with arbitrary metrics, such as the signal latency. Thus the cost maps are free to adapt to new constraints and design rules.

In the OpenAI-ES algorithm, we will sample a fixed number of Gaussian noises and add them to the original parameters to form new parameters. These new parameters are then fed into n evaluators (or workers). The evaluator has an independent environment and executes the algorithm based on received parameters. Finally, each evaluator will return a scalar reward. To stabilize the training process, we normalize collected rewards $\{r_1, ..., r_n\}$ from all the evaluators:

$$\bar{r}_i = (r_i - r_{mean})/r_{std}, \quad (12)$$

where r_{mean} and r_{std} are the mean and standard deviation of all the rewards.

To further improve the final results, we add a post-processing step to the solution obtained from the Ranking Cost algorithm. In the post-processing step, we will choose a connected path and re-plan it using the canonical A* algorithm while keeping other paths fixed. If we find a shorter path during the re-planning, we will update the current path to the new one. This post-processing step will be iteratively applied on each path until all the paths can not be updated. In practice, all the evaluators can execute in parallel as proposed in [Salimans *et al.*, 2017].

5 Experiments

In this section, we will compare our proposed method with some other baselines. We will also study how ranking learning and cost maps work in our algorithm. Finally, we further show the scalability of our algorithm for larger applications.

5.1 Methods Evaluation

To evaluate our algorithm, we build a grid map simulator as the test environment. We construct 300 maps with three different sizes: 16×16, 32×32 and 64×64. These maps are split into two types, i.e., a simple one without obstacle and a more complex one with obstacles. Figure 1(a) and Figure 1(b) show some of the maps used in our experiments. Algorithms will be evaluated on these maps, and the success rates and the average of total lengths will be reported. Our experiments are performed on a desktop machine with 128 GB RAM and one 64-core CPU. More details about the maps can be found in the supplementary material.

Baselines:
Sequential A*: Sequential A* is a common used search-based algorithm in circuit routing. It routes nets with a

	16×16 (4 pairs)	32×32 (32 pairs)	64×64 (10 pairs)
	no obstacle	no obstacle	no obstacle
Seq A*(5)	0.92±0.02(37.0±0.16)	0.96±0.01(109.9±0.36)	0.76±0.04(434.2±5.06)
Seq A*(200)	0.96±0.0(36.5±0.0)	1.0±0.0(**107.0**±0.0)	0.84±0.0(**400.0**±1.11)
VINs	0.86±0.0(37.1±0.27)	0.52±0.03(124.1±1.3)	0(-)
CML	0.93±0.0(38.1±0.02)	0.98±0.0(111.3±2.0)	0.88±0.0(457.7±0.0)
Ranking Cost I	1.0±0.0(36.5±0.02)	1.0±0.0(108.2±0.32)	0.86±0.0(417.1±0.0)
Ranking Cost II	**1.0**±0.0(**36.5**±0.02)	**1.0**±0.0(108.2±0.34)	**0.86**±0.0(413.8±0.0)
	with obstacles	with obstacle	with obstacle
Seq A*(5)	0.92±0.01(36.3±0.09)	0.53±0.02(105.4±0.73)	0.1±0.03(384.0±13.65)
Seq A*(200)	0.96±0.0(36.1±0.0)	0.78±0.0(103.8±0.0)	0.26±0.01(367.5±0.0)
VINs	0.87±0.01(36.4±0.16)	0.21±0.02(114.7±2.09)	0(-)
CML	0.9±0.0(36.8±0.09)	0.67±0.01(104.4±0.31)	0.16±0.0(405.5±0.0)
Ranking Cost I	0.98±0.0(36.1±0.02)	0.82±0.0(103.8±0.0)	0.32±0.0(360.5±0.0)
Ranking Cost II	**0.98**±0.0(**36.1**±0.02)	**0.82**±0.0(**103.8**±0.0)	**0.32**±0.0(**358.5**±0.0)

Table 1: Evaluation results of different algorithms over 6 seeds. Two values and their corresponding standard deviations are reported. The first value is the success rate (the higher the better) and the second value in the bracket is the common average length (the lower the better). Our methods achieve the best performance on most of the maps. The comparison between CML and RC shows that the ranking parameter learning can improve the performance of our algorithm.

Seq A*(5)	Seq A*(200)	VINs	CML	Ranking Cost I	Ranking Cost II
0.8 sec/map	41.5 sec/map	1.4 sec/map	37.5 sec/map	38.3 sec/map	38.3 sec/map

Table 2: The wallclock time of different algorithms. The Ranking Cost algorithm takes more running time, since it involves a learning procedure when solving each task. But the time consumption is acceptable, i.e., our algorithm takes less than one minute to get the solution, and it is worthy to sacrifice more time to earn better solutions.

specific ranking order and different orders will lead to different results. In the experiments, we randomly sample 5 different ranking orders and run the algorithm 5 times. The best score of 5 runs will be reported. To build a stronger baseline, we further randomly sample 200 different ranking orders and the best score of 200 runs will be reported.
Value Iteration Networks (VINs) [Tamar *et al.*, 2016]: The VIN is a learning-based approach for routing on grid maps. VINs use the value iteration from reinforcement learning and can achieve better performance compared with supervised models. We apply VINs to the circuit routing problem by sequentially executing it on each net. Similar to the sequential A* algorithm, we randomly sample five different ranking orders and the best score of five runs will be reported.
Cost-Map Learning (CML): We construct a Cost-Map Learning algorithm from the Ranking Cost. In the CML, the ranking parameter will not be trained (the ranking order is fixed) and only the cost-map will be trained.

We also implement two versions of the Ranking Algorithm, i.e., the version without post-processing step is denoted as Ranking Cost I and the version with post-processing step is denoted as Ranking Cost II. Table 1 shows the evaluation results of different algorithms on all the maps over 6 seeds. Two values and their corresponding standard deviations are reported for each algorithm. The first value is the success rate (the higher the better) and the second value in the bracket is the common average length (the lower the better). Our final algorithm (Ranking Cost) achieves the best performance

on most of the maps. On the most complex maps (with size 64×64, obstacles and 10 pairs of vertices), Ranking Cost outperforms other baselines by a large margin, which implies that our method has greater advantage on handling larger maps. The comparison between Seq A*$_{(200)}$ and Ranking Cost shows that our method can achieve the performance which the fully sampled A* will never achieve. Taking maps with the size of 16x16 (4 pairs) for example, there are only $4! = 24$ orders, and we sampled all the possible orders for the A* since our maximum sampling number for is 200. But the final connecting rate of the A* is 96%. However, our RC algorithm can reach 100% connecting rate. Even given the full sampling, the A* will fail in many cases. Moreover, the comparison between Cost-Map Learning and Ranking Cost shows that the ranking parameter learning can further improve the performance of our algorithm. Table 2 shows the wallclock time of different algorithms. The comparison between Ranking Cost I and Ranking Cost II shows that our post-processing step can help to reduce the total connected length with insignificant time-consuming.

5.2 Ablation Study

In this section, we study how each part of our algorithm works and the scalability of our algorithm.

Impact of Ranking Orders
To show the impact of ranking learning, we increase the sample number of orders for sequential A* and use the Ranking Learning algorithm described in Section 4.1 as a comparison. Figure 2(a) shows the curve of the changes in success rates
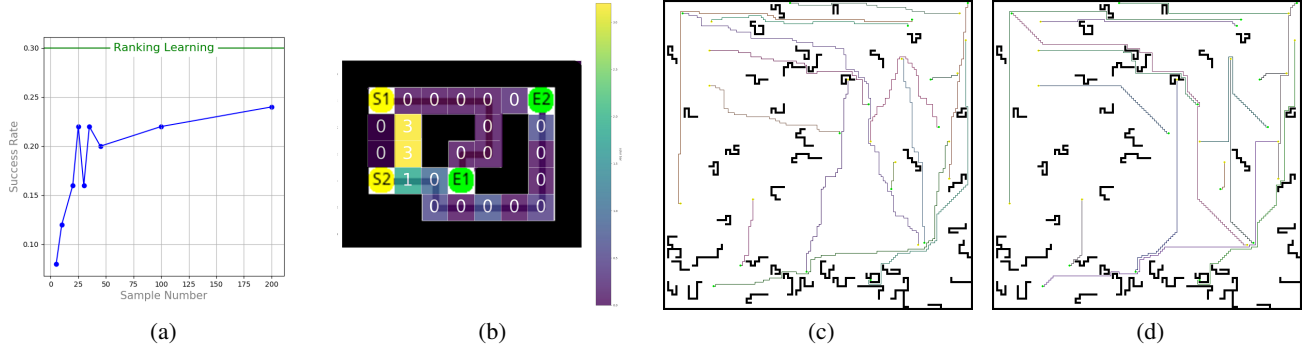
Figure 2: (a) The curve of success rates and sample numbers and the experiment is conducted on maps with size of 64×64 and with obstacles. Our learned ranking order achieves better performance than randomly sampled orders. (b) Trained cost map of the example from Figure 1(f). There are two larger cost values (the yellow grids) in the cost map, which will give out a large cost if the path chooses to go through here. As a result, the first pair $((S_1, E_1))$ will not take its shortest path which will block the second pair$((S_2, E_2))$. It shows that cost maps can capture global information and guide routers to complete the task successfully. (c) An example of routing results of the Ranking Cost algorithm on a large map with the size of 150×150 and with 15 pairs, which implies that our method could be adapted to larger scales of applications. (d) An example shows the routing result of the Ranking Cost algorithm with the post-processing step, which implies that the post-processing step could improve the final planning result.

with different order sample numbers. The result shows that we can improve the success rate by increasing the order sample number, but it is hard to further improve the performance by linearly increasing the sample number as the order complexity is $O(k!)$, where k is the number of pairs (or nets). However, our learned ranking order can achieve much better performance than randomly sampled orders.

Impact of Cost Maps

To study the impact of cost maps, we fix the ranking order and only train the cost-map parameter. We use the example map showed in Figure 1(f) and the ranking order is fixed as $R = (1, 2)$. Because the ranking order is fixed and there are only two cost maps, only the second cost map will be used when routing the first pair (i.e., (S_1, E_1)). Finally, we visualize the trained cost maps as shown in Figure 2(b). There are two larger cost values (the yellow grids) in the cost map, which will give out a large cost if the path chooses to go through here. As a result, the first pair $((S_1, E_1))$ will take the path as shown in Figure 1(f) instead of taking its shortest path which will block the second pair$((S_2, E_2))$. It shows that our cost maps can capture global information and guide routers to approach global optima.

Scalability of Ranking Cost Algorithm

In [Liao *et al.*, 2020], their algorithm is only evaluated on the maps with the size of 8×8. In [He and Bao, 2020], their algorithm is evaluated on the maps with the size of 30×30 and with only 5 pairs. The search space is huge in their methods, which prevents them from applying to larger maps. To test the scalability of our algorithm, we apply our algorithm on larger maps with the size of 150×150 and with 15 pairs. Our algorithm solves such large maps within 3 min and Figure 2(c) shows the routing result of our method without post-processing step, which implies that our method could be adapted to larger scales of applications. Figure 2(d) shows the routing result of the Ranking Cost algorithm with the post-

processing step, which implies that the post-processing step could improve the final planning result.

6 Discussion

In this paper, we propose a novel algorithm, called Ranking Cost, to solve the historically challenging circuit routing problem. In our method, we innovatively combine search-based algorithms and learning-based algorithms to form an efficient and trainable router under a proper parameterization. Our method is a one-stage circuit routing algorithm which can optimize the global objective function directly, and it is easy to implement and flexible for new design rules and constraints. Experimental results show that our method is powerful and scalable to more complex tasks. In the future, we will try to explore faster optimizer for the Ranking Cost parameters and extend our algorithm to broader applications, such as pedestrian path prediction and robot navigation. Lastly, we do not believe our work has broader negative societal impacts, as we focus on developing circuit routing algorithms with evolution-based methods.

References

[Chen and Chang, 2009] Huang-Yu Chen and Yao-Wen Chang. Global and detailed routing. In *Electronic Design Automation*, pages 687–749. Elsevier, 2009.

[Cho *et al.*, 2007] Minsik Cho, Katrina Lu, Kun Yuan, and David Z Pan. Boxrouter 2.0: Architecture and implementation of a hybrid and robust global router. In *2007 IEEE/ACM International Conference on Computer-Aided Design*, pages 503–508. IEEE, 2007.

[Coombs and Holden, 2001] Clyde F Coombs and Happy T Holden. *Printed circuits handbook*, volume 1. McGraw-Hill New York, 2001.

[de Vincente *et al.*, 1998] J de Vincente, Juan Lanchares, and Román Hermida. Rsr: A new rectilinear steiner minimum tree approximation for fpga placement and global routing. In *Proceedings. 24th EUROMICRO Conference (Cat. No. 98EX204)*, volume 1, pages 192–195. IEEE, 1998.

[Hansen and Ostermeier, 2001] Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, 9(2):159–195, 2001.

[He and Bao, 2020] Youbiao He and Forrest Sheng Bao. Circuit routing using monte carlo tree search and deep neural networks. *arXiv preprint arXiv:2006.13607*, 2020.

[Ho *et al.*, 1991] T-T Ho, S Sitharama Iyengar, and S-Q Zheng. A general greedy channel routing algorithm. *IEEE transactions on computer-aided design of integrated circuits and systems*, 10(2):204–211, 1991.

[Hu and Sapatnekar, 2001] Jiang Hu and Sachin S Sapatnekar. A survey on multi-net global routing for integrated circuits. *Integration*, 31(1):1–49, 2001.

[Huang *et al.*, 2014] Tsung-Wei Huang, Pei-Ci Wu, and Martin D Wong. Ui-route: An ultra-fast incremental maze routing algorithm. In *2014 ACM/IEEE International Workshop on System Level Interconnect Prediction (SLIP)*, pages 1–8. IEEE, 2014.

[Kramer, 1984] Mark R Kramer. The complexity of wirerouting and finding minimum area layouts for arbitrary vlsi circuits. *Advances in computing research*, 2:129–146, 1984.

[Liao *et al.*, 2020] Haiguang Liao, Wentai Zhang, Xuliang Dong, Barnabas Poczos, Kenji Shimada, and Levent Burak Kara. A deep reinforcement learning approach for global routing. *Journal of Mechanical Design*, 142(6), 2020.

[Malavasi and Sangiovanni-Vincentelli, 1993] Enrico Malavasi and Alberto Sangiovanni-Vincentelli. Area routing for analog layout. *IEEE transactions on computer-aided design of integrated circuits and systems*, 12(8):1186–1197, 1993.

[Mandal *et al.*, 2020] Tarak Nath Mandal, Ankita Dutta Banik, Kaushik Dey, Ranjan Mehera, and Rajat Kumar Pal. Algorithms for minimizing bottleneck crosstalk in two-layer channel routing. In *Computational Advancement in Communication Circuits and Systems*, pages 313–330. Springer, 2020.

[Mo *et al.*, 2001] Fan Mo, Abdallah Tabbara, and Robert K Brayton. A force-directed maze router. In *IEEE/ACM International Conference on Computer Aided Design. ICCAD 2001. IEEE/ACM Digest of Technical Papers (Cat. No. 01CH37281)*, pages 404–407. IEEE, 2001.

[Rechenberg, 1994] Ingo Rechenberg. Evolutionsstrategie: Optimierung technischer systeme nach prinzipien der biologischen evolution. frommann-holzbog, stuttgart, 1973. *Step-Size Adaptation Based on Non-Local Use of Selection Information. In Parallel Problem Solving from Nature (PPSN3)*, 1994.

[Salimans *et al.*, 2017] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

[Schaller, 1997] Robert R Schaller. Moore's law: past, present and future. *IEEE spectrum*, 34(6):52–59, 1997.

[Shi and Davoodi, 2017] Daohang Shi and Azadeh Davoodi. Trapl: Track planning of local congestion for global routing. In *Proceedings of the 54th Annual Design Automation Conference 2017*, pages 1–6, 2017.

[Shi *et al.*, 2016] Daohang Shi, Azadeh Davoodi, and Jeffrey Linderoth. A procedure for improving the distribution of congestion in global routing. In *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 249–252. IEEE, 2016.

[Tamar *et al.*, 2016] Aviv Tamar, Yi Wu, Garrett Thomas, Sergey Levine, and Pieter Abbeel. Value iteration networks. In *Advances in Neural Information Processing Systems*, pages 2154–2162, 2016.

[Williams, 1992] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[Zhang, 2016] Ran Zhang. *A Study of Routing Algorithms for PCB Design*. PhD thesis, 2016.