Track-Assignment Detailed Routing Using Attention-based Policy Model With Supervision

Haiguang Liao
haiguanl@andrew.cmu.edu
Carnegie Mellon Unversity
Pittsburgh, PA

Qingyi Dong
qingyid@andrew.cmu.edu
Carnegie Mellon Unversity
Pittsburgh, PA

Weiyi Qi
weiyi@cadence.com
Cadence Design Systems
San Jose, CA

Elias Fallon
fallon@cadence.com
Cadence Design Systems
San Jose, CA

Levent Burak Kara
lkara@cmu.edu
Carnegie Mellon Unversity
Pittsburgh, PA

ABSTRACT

Detailed routing is one of the most critical steps in analog circuit design. Complete routing has become increasingly more challenging in advanced node analog circuits, making advances in efficient automatic routers ever more necessary. In this work, we propose a machine learning driven method for solving the track-assignment detailed routing problem for advanced node analog circuits. Our approach adopts an attention-based reinforcement learning (RL) policy model. Our main insight and advancement over this RL model is the use of *supervision* as a way to leverage solutions generated by a conventional genetic algorithm (GA). For this, our approach minimizes the Kullback-Leibler divergence loss between the output from the RL policy model and a solution distribution obtained from the genetic solver. The key advantage of this approach is that the router can learn a policy in an offline setting with supervision, while improving the run-time performance nearly 100× over the genetic solver. Moreover, the quality of the solutions our approach produces matches well with those generated by GA. We show that especially for complex problems, our supervised RL method provides good quality solution similar to conventional attention-based RL without comprising run time performance. The ability to learn from example designs and train the router to get similar solutions with orders of magnitude run-time improvement can impact the design flow dramatically, potentially enabling increased design exploration and routability-driven placement.

KEYWORDS

reinforcement learning, supervised learning, policy model, detailed routing

ACM Reference Format:
Haiguang Liao, Qingyi Dong, Weiyi Qi, Elias Fallon, and Levent Burak Kara. 2020. Track-Assignment Detailed Routing Using Attention-based Policy Model With Supervision. In *2020 ACM/IEEE Workshop on Machine Learning for CAD (MLCAD '20), November 16–20, 2020, Virtual Event, Iceland.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3380446.3430629

MLCAD '20, November 16–20, 2020, Virtual Event, Iceland

ACM ISBN 978-1-4503-7519-1/20/11.
https://doi.org/10.1145/3380446.3430629

1 INTRODUCTION

Detailed routing is a critical step in the physical design of VLSI analog circuits, where actual routes are constructed based on placement and global routing steps. In advanced node technologies, the increased complexity of the problems renders conventional routers difficult to scale to and performs efficiently on the types of problems shown in Fig. 1. While prior analog routing works [6, 9, 17] have proposed template-based, simulation-based, and heuristic-based methods, the key issue of a generalizable net sequencing and pin-pair connection remains largely unsolved. Moreover, a lack of suitable mechanisms that can learn from past routing solutions prevents the use of such valuable data toward an efficient solution of new routing problems. Based on these observations, new data-driven methods have been recently proposed for analog routing and associated classes of problems, [8, 10, 11]. Most closely related to our work, Liao *et al.* propose an attention-based reinforcement learning (ARL) method for detailed routing [8]. However, the approach also suffers from at-times unstable training and fails to take advantage of the already computed genetic routing solutions, as no labeled data can be used to supplement the ARL model.

1.1 Our Contributions

In this work, we propose a new analog router model that builds on ARL, whose training can now be additionally supervised with past solutions. We refer to our routing method as supervised reinforcement learning (SRL). Once trained, both ARL and the new SRL provide nearly 100× acceleration in runtime performance over the GA while generating solutions that are comparable in quality to those generated by GA. Moreover, we show that as the problem complexity increases, SRL solution quality remains more similar to that of GA on previously unseen problems.

Our work thus demonstrates through the use of supervised RL, routing solutions can be generated significantly faster over conventional methods without compromising solution quality. This capability enables significant future potential flow innovations. With a learn-by-example capability in routing, the user could provide examples of manufacturing proven designs which were routed by experts, and have the router learn the key decisions necessary to

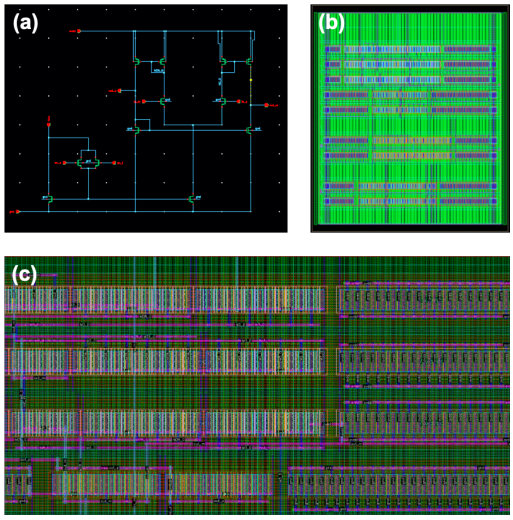

Figure 1: Advanced node technologies analog circuits *biasamp* (random generated) (a) schematic and (b) design solutions given by Cadence Virtuoso, (c) magnified version of design solutions given by Cadence Virtuoso.

match those results. The router then has the potential to implicitly learn the user's underlying quality factors. Subsequently being able to create new routing based on the learned policy, at significantly faster run-time than traditional routers, enables integration of those results into other steps of the design creation flow. For instance, placement algorithms can utilize fast routers in their inner loop as objective functions in their optimization.

2 PRELIMINARIES

2.1 Track-assignment detailed routing

Routing is typically addressed in two sequential steps: *global routing* and *detailed routing*. Global routing approximately allocates routing resources into sub-regions while detailed routing constructs the actual routes in the individual sub-regions while satisfying various design rule constraints. In this work, we focus on routing completion success and overall wirelength as the two primary considerations of a detailed router's solution quality. We thus abstract any applicable design rules into Width Spacing Patterns (WSP), where track patterns consisting of various width and spacing parameters for metal wires are prescribed. By restricting the routes on the WSP rows and tracks, many design rules associated with full custom designs can be imposed a priori using a modified weighted bipartite matching formulation [8].

The use of WSP for design rule abstraction, net decomposition into pin-pairs, and the use of a pattern router for eventual routing allows all three methods (GA, ARL, and the new SRL) discussed in this work to be compared on a common basis. Each of these methods aims to produce a sequencing of the pin-pairs such that when the pairs are routed in that sequence using the pattern router, the overall quality of the solution (measured in terms of the number of completed routes and total wirelength) can be maximized.

2.2 Attention-based reinforcement learning policy model

Recent studies have demonstrated the success of reinforcement learning for combinatorial optimization problems with better generalization capabilities and runtime performance over heuristic based or exact methods [1, 2, 7]. Among these works, Kool *et al.* [4] uses an attention-based reinforcement learning policy model to solve the Traveling Salesman Problem problem (TSP). They achieve state-of-the-art performance with an ability to generalize to unseen problems over previous approaches [12, 15]. In solving the TSP with the policy model, a solution can be defined as a tour π, which consists of an ordered sequence of nodes (cities). At each step, the policy model generates a probability distribution $p_\theta(\pi_t|s, \pi_{1:t-1})$ over the possible nodes to visit next. A solution for a TSP problem can be obtained in n steps, where n represents the number of cities in the TSP problem. Based on this mechanism, the work defines a problem policy by multiplying the probability distributions at each step, as shown in Eqn. 1.

$$p_\theta(\pi|s) = \prod_{i=1}^{n} p_\theta(\pi_t|s, \pi_{1:t-1}) \tag{1}$$

In order to obtain a robust policy model, the policy model needs to be optimized with respect to its parameters. Based on the problem policy and the policy gradient theorem used in REINFORCE, the gradient of the loss function for the policy model can be stated as: $\nabla_\theta E_{\pi\sim p_\theta(\pi|s)}[L(\pi)]$. This can be calculated in Eqn. 2 based on policy gradient theorem [14], which provides a feasible way to calculate the otherwise intractable gradient information to optimize the model parameters:

$$\nabla_\theta E_{\pi\sim p_\theta(\pi|s)}[L(\pi)] = E_{\pi\sim p_\theta(\pi|s)}[L(\pi)\nabla_\theta log p_\theta(\pi|s)] \tag{2}$$

The structure of the applied policy model is an attention-based encoder-decoder (Fig. 2). This model is as a variation of Graph Attention Networks whose advantage is the ability to solve a sequential problem in a sequence-independent way.

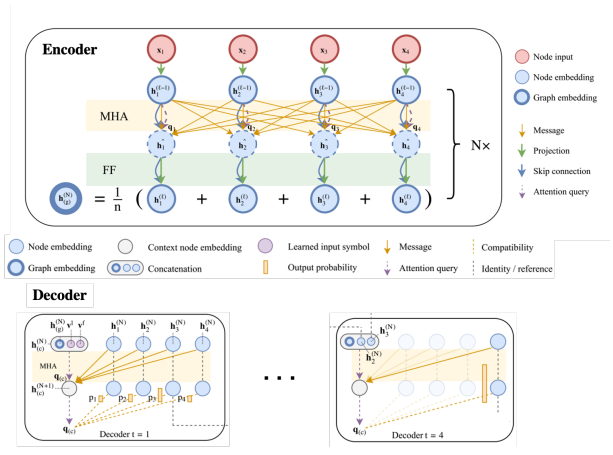

Figure 2: Schematic plot of the attention-based model. Adopted from [4]

Driven by the similarity between the analog detailed routing problem and the TSP, previous work [8] has successfully applied an attention-based reinforcement learning (ARL) policy model to solve the track-assignment detailed routing problems. In that work, ARL is able to solve detailed routing problems in a generalizable way. The most significant contribution of ARL is a new, fast method for sequencing the pin pairs extracted from the netlists. Without an appropriate sequence, detailed routing often needs to be solved using computationally demanding and sub-optimum rip-up and re-route strategies [3]. The ARL is trained without any supervision. This works builds upon ARL and introduces supervision provided by the GA solutions. We call the resulting approach supervised RL (SRL) for routing and discuss it next. This work is inspired by recent works in complementing RL with supervision [13, 16]

2.3 Policy model trained with supervision

In order to enable supervision for the policy model, a target (label) similar to the output of the problem policy must be generated. We use the GA solution as labeled data. We transform these solutions obtained in the populations to probability distributions for use in the policy SRL model. Specifically, consider a given problem instance s which consists of n two-pin pairs to be ordered and routed using a pattern router. After GA solver is applied, a set of optimized (but not necessarily optimal) sequences $\{\pi_1, \pi_2, ..., \pi_m\}$ of the n two-pin pairs can be obtained. At each location $q_t, t = 1, ..., n$ of the sequence, an empirical distribution among two-pin pairs can be obtained. Based on this distribution, by multiplying the $q_t, t = 1, ..., n$, a label distribution for the policy of problem instance s can be obtained. For ease of algebraic manipulation, we utilize the log-label in the form:

$$log(q(s)) = log(\prod_{t=1}^{n} q_t) = \sum_{t=1}^{n} log(q_t) \tag{3}$$

Both q_t and $q(s)$ are vectors and the log operations applied to the individual components in the vector, thus $log(q(s))$ is also a vector. With this label and problem policy given by the attention-based policy model, a loss is formulated based on Kullback-Leibler (KL) divergence:

$$L(s) = D_{KL}(p_{\theta(s)}|q(s)) = p^T_{\theta(s)}(log(p_{\theta(s)}) - log(q(s))) \tag{4}$$

This loss encodes the cross-entropy loss between two distributions; one from the policy model and the other one from the GA solutions. We use a discretized version:

$$L(s) = \sum_{i=1}^{n} p_{\theta(s)}(i)(log(p_{\theta(s)}(i)) - log(q(s))(i)) \tag{5}$$

2.4 Problem Formulation

Fig. 3 shows the proposed learning-based solution flow for track-assignment detailed routing. The input of the flow is a set of placement solutions $\mathbf{S} = \{s_1, s_2, ..., s_m\}$ for a sub-region of a given analog circuit design to be routed. The placement solutions (of a sub-region of a chip) specify the locations of the devices and netlists containing the grouping information of the devices that need to be connected in the detailed routing stage. It is assumed that the placement solution sets **S** for a specific region of a given analog design follows a certain distribution (which is the probability of different devices appearing at the different locations in the design space), instead of a random distribution. This is based on the observation that placement solutions are typically generated by considering similar objectives such as device density or potential drops. After the track assignment and pin decomposition, a detailed routing for each placement s_i is formulated as the connection of a set of two-pin pairs $N = \{t_1, t_2, ..., t_n\}$ with predetermined spatial locations and following a sequence $\pi = \{\pi_1, \pi_2, ..., \pi_n\}$. The number of possible sequences is governed by all the permutations within the two-pin pairs set N. The sequential detailed routing problem is formally formulated as follows:

Sequential Detailed Routing: Consider a set of two-pin pairs $N = \{t_1, t_2, ..., t_n\}$ on a detailed routing region represented as a graph $G(V, E)$, in which pins are distributed on different vertices and edges. Only horizontal and vertical directions are the feasible routing directions, with a routing capacity of 1. The sequential detailed routing engine solves the detailed routing problem in two steps. First, an optimized sequence π of two-pin pairs are generated by a solver. Then, following the sequence π, a pattern router routes individual two-pin pairs sequentially while sharing the same graph $G(V, E)$.

Objective Function: The metrics to be optimized include: (1) the total wirelength of all two-pin pair routes, (2) the total number of openings described as the unrouted two-pin pairs. When a two-pin pair can not be routed, the total number of openings increased by one. For ease of implementation, a weighted sum of the total number of openings and total wirelength is used as the objective function, hence cost, to minimize:

$$Cost(s) = w_1 * Wirelength + w2 * \#Open \tag{6}$$

The lower the cost, the better the solution quality is. The weights of the two terms can be adjusted by the end user to bias the two components of the objective differently.

3 ALGORITHMS

As shown in Fig. 3, we analyze the GA, ARL, and SRL methods. All three methods use the same the track assignment method and pin decomposition algorithm (Kruskal's algorithm [5]). All three methods use the same pattern router consisting of "L" and "Z" which provides a common basis for comparing the algorithms. The GA-based sequencing optimizes the sequence π for a given problem through multiple generations of iterations. It uses permutation-based crossover and mutation operations for creating off-springs. The ARL is used as a way to learn to generate optimized sequences through self-training on a set of problems in previous work, which is built upon the attention-based reinforcement learning policy model.

In this work, we propose a supervised version of ARL, we call SRL, based on the KL-divergence loss and the GA solutions as supervision, as shown in Algorithm 1. During training, 60% of placement solutions of a given design is used as training data, 20% is used for validation and 20% is used as testing. Given the training problems, GA first generates results in the form of a distribution q based on

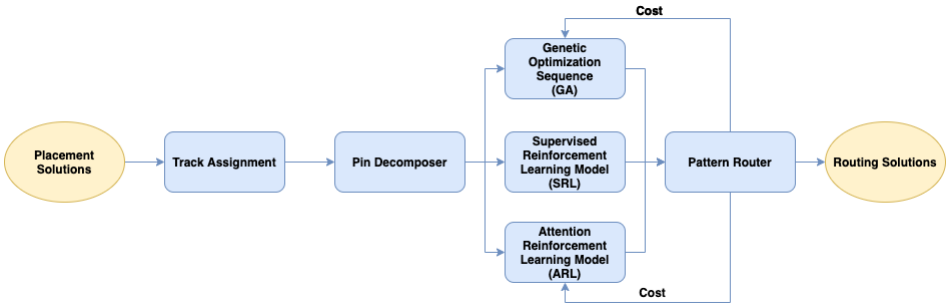

Figure 3: The flow of proposed learning-based solutions for track-assignment detailed routing.

formulation Eqn. 3 after 10 generations. After q is obtained from the GA, SRL training is initiated by calculating the KL-divergence loss between q distribution and distribution output p_θ from the policy model. The loss is then used for updating the policy model parameters by backpropagation, which we perform for 100 epochs.

Once training is complete, SRL provides pin pair sequences for previously unseen problems (placements) from the same data set in a forward fashion. The runtime computational cost of the trained SRL (as well as ARL) is negligible compared to GA, as only matrix multiplications are involved during run time.

Algorithm 1: SRL model.

Input : Number of epochs E, batch size B, training set T (with GA-based label, q)
Output: Sequence based on best policy

Init $\theta \leftarrow \theta$;
for *epoch=1,...,E* **do**
  for *batch=1,...,B* **do**
    $t_i \leftarrow$ SampleInstance() $\forall i \in 1, ..., T$;
    $\pi_i \leftarrow$ SampleRollout(t_i, p_θ) $\forall i \in 1, ..., T$;
    $L \leftarrow \sum_{i=1}^{B} p_\theta^T(i)(log(p_\theta(i)) - log(q(i)))$;
    $\theta \leftarrow$ Adam(θ, ΔL);
  end
end

4 EXPERIMENTAL RESULTS

Data: To evaluate the three methods and compare their performance, we use two kinds of analog design problems as follows:

(1) *biasamp*: This consists of comparators and OpAmps, and less than 100 devices in each potential design. The *biasamp* problems are randomly generated placement solutions to explore the model properties and therefore does not represent feasible designs for actual analog circuits.
(2) *sarfsm*: This is a subcircuit of an Analog-to-Digital Converter, with hundreds of devices in each potential design. This problem is more complex compared to *biasamp*.

Both problems are representative of advanced node technologies (sub-16 nm technology) analog circuits design problems. In order to analyze the sample efficiency for training the SRL models, we generate three data sets consisting of a different number of device configurations (designs):

(1) *biasamp*500: biasamp with 500 device configurations
(2) *biasamp*5000: biasamp with 5000 device configurations
(3) *sarfsm*500: sarfsm with 500 device configurations

Implementation: All three routing algorithms and the whole flow is implemented in **Python3.6** with the machine learning framework **Pytorch**. Experiments are run on a workstation with an Intel Core i7-6850 CPU without GPU acceleration. All models are trained using the following parameters: training epochs: 100, batch size: 5, optimizer: Adam.

Training Time: Training time varies with problem size and the size of the data set. SRL takes 6 minutes for training in *biasamp*500, and 30 minutes for training in *sarfsm*500. ARL's training time is virtually identical to that of SRL's. While currently not used, the MHA mechanism of both the SRL and ARL can be readily parallelized, which is the subject of our future work.

4.1 SRL performance

Fig. 4 compares the results of SRL against the GA solutions with respect to the solution cost, with correlation analysis between these two algorithms. Each red point corresponds to a unique device configuration problem that is routed both by SRL and GA. In the three data sets studied (rows), both the training and test sets (columns) reveal a strong linear correlation between the SRL and GA solutions. The resulting slopes (<1.0) and insignificant intercepts (compared to the range of cost) indicate that SRL solution quality is slightly less than that of GA's. This is further quantified in the next few paragraphs.

For the same data set, the correlation strength between SRL and GA remain consistent across the training and test sets. This indicates that the supervised training on can effectively generalize to the previously unseen test sets. The second observation is that increasing the size of data set can effectively increase the SRL's strength of correlation with GA in both the training and test set, as seen from *biasamp500* (Fig. 4a and Fig. 4b) and *biasamp5000* (Fig. 4c and Fig. 4d). Specifically, after significantly increasing the number of placement solutions in the data set, the R^2 score between SRL and GA increase from 0.954 to 0.995.

The cost vs. run time comparisons of the SRL and GA models on test sets *bias500* and *sarfsm500* are shown in Fig. 5. In the plots, blue dots correspond to the SRL cost and run time, and red dots

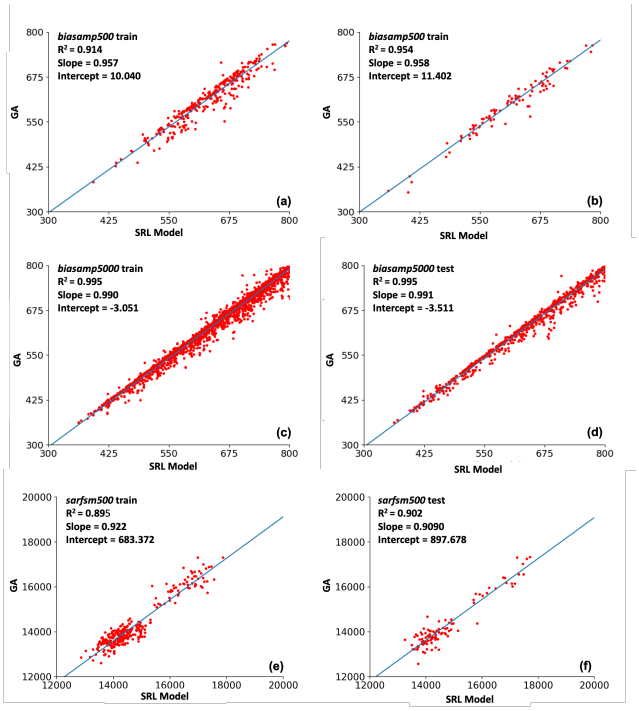

Figure 4: Correlation analysis between SRL solutions' cost and GA solutions' cost on (a) *biasamp500* training set, (b) *biasamp500* test set, (c) *biasamp5000* training set, (d) *biasamp5000* test set, (e) *sarfsm500* training set, (f) *sarfsm500* test set.

correspond to GA's cost and run time. The red-blue point pairs connected with gray lines correspond to the same problem. As seen, the run time performance or SRL is two orders of magnitude better than that of the GA's, while only encountering a slight degradation in the solution quality. To show the difference in the quality of the solutions between SRL and GA, Fig. 6 shows the histograms of signed deviations of SRL model's cost from the corresponding GA-based results on different data sets. In these plots, the random variable is $\frac{Cost^{SRL}-Cost^{GA}}{Cost^{GA}}$. By comparing the results on different data sets, it is seen that larger data sets and more complex designs result in a wider variation in the SRL solution relative to GA. In all the data sets, SRL's cost are within $[-8\%, 10\%]$ of the GA cost, while in most cases SRL's results are within $\pm 5\%$ of the GA results. This demonstrates that for the data sets studied in this work, the SRL policy would be at most 10% worse than GS, while generating solutions around 1% of the time it takes for GA.

4.2 SRL Comparison with ARL

Correlation analysis between SRL and ARL models on the training and test sets of *biasamp5000* and *sarfsm500* is also conducted. On both data sets, the two models solutions' cost show relatively strong correlations on the training and test, and the R^2 scores on the training and test for the same design is similar. However, the R^2 scores on the training ($R^2 = 0.940$) and test set ($R^2 = 0.935$) of

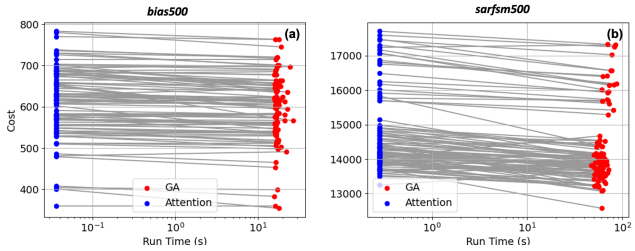

Figure 5: Cost vs. Runtime comparison between SRL and GA for test set of (a) *biasamp500* and (b) *sarfsm500*.

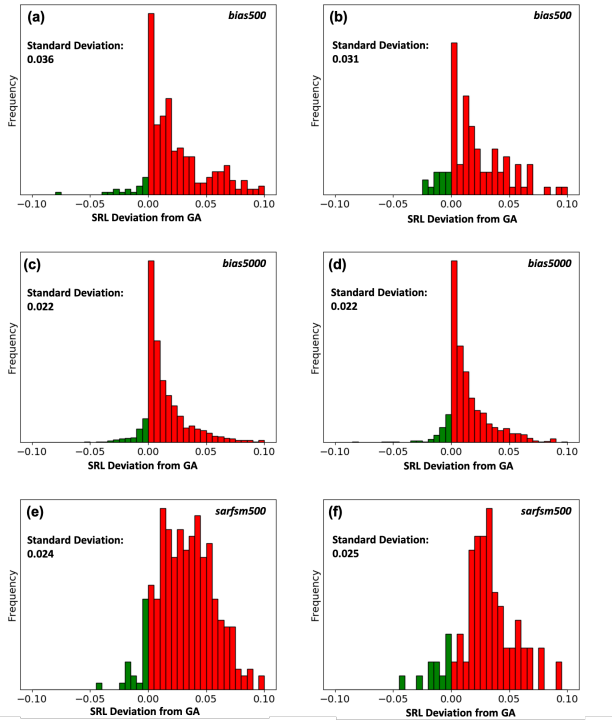

Figure 6: Histograms showing deviation of SRL from GA for training sets of (a) *biasamp500*, (b) *biasamp5000*, (c) *sarfsm500*, and test set of (d) *biasamp500*, (e) *biasamp5000*, (f) *sarfsm500*.

biasamp are higher than those of *sarfsm* (training: $R^2 = 0.821$,test: $R^2 = 0.860$). This indicates that the possibility of higher similarity between solutions of the ARL and SRL models on *biasamp* than that of *sarfsm*, which can be confirmed by comparing the actual routing solutions of three methods (SRL, ARL and GA) on test set of *sarfsm500* test set (Fig. 7).

For *sarfsm*, less similar patterns can be observed between SRL (Fig. 7a) and ARL (Fig. 7b) model's routes; while more similar patterns can also be found between the SRL policy model (Fig. 7a) and GA (Fig. 7c), than the comparison between the ARL policy model (Fig. 7b) and the GA (Fig. 7c). This shows the possibility that the SRL policy model in better imitating the behavior of methods used for

generate supervision. Yet, further statistical analysis on larger data sets is needed to consolidate this observation. Another interesting observation is that, given the divergence behavior of ARL and SRL policy model on *sarfsm* data set, there is still a high correlation between their solutions cost . This shows the possibility that ARL policy model can achieve as good performance of SRL policy model, yet with less similar routing patterns of GA-based routes. The ARL policy model results' similarity to GA is also analyzed: in both training and test sets of *biasamp5000* and *sarfsm500*, the ARL policy model's cost deviation from GA is similar to the pattern displayed by SRL policy model (Fig. 6) with maximum deviation less than 10%. In general, the ARL policy model is as good as the SRL policy model given the data sets used in this research.

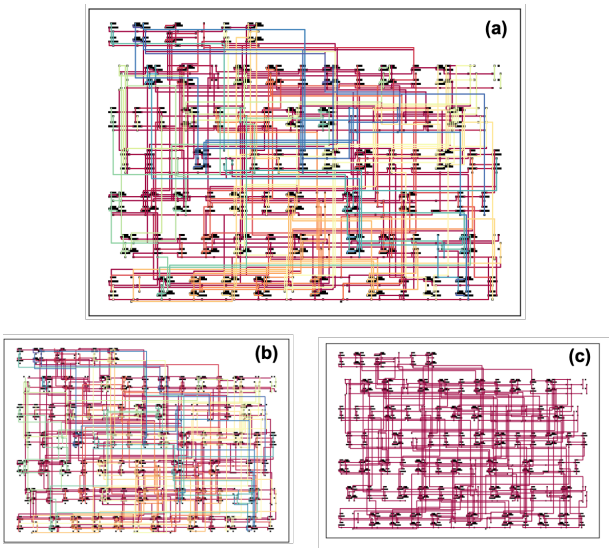

Figure 7: Visualization of detailed routing solutions for a *sarfsm500* problem in test data set given by (a) supervised learning policy model, (b) reinforcement learning policy model and (c) genetic algorithm.

5 CONCLUSION

We present a new supervised learning approach for training the policy model in solving track-assignment detailed routing. The supervision is based on minimizing the KL divergence loss between the output from the policy model and a given distribution from the GA solver. It achieves performance comparable to reinforcement learning trained policy model, indicating the supervised learning as an alternative approach for training the policy model, that can leverage existing expert solutions to obtain user customized routers. Both supervised trained and reinforcement learned policy models have around 100× acceleration compared with the baseline GA solver. To better understand the policy models' behavior in solving the sequencing step in detailed routing, a statistical analysis of the results are applied. The results show high correlations between GA results' cost and policy model results' cost, indicating that policy models can be deployed as a fast and high-quality surrogate model for GA in solving sequential detailed routing. This would enable better routability-driven placement driven optimization: with much less routability evaluation time for a placement solutions, the optimization resources for placement optimization has a two scale increase. By comparing the behavior and working principle of supervised learning and reinforcement learning in training policy models in this work, it is concluded that SRL model is able to behave as well as ARL model, while provides a feasible way for learning from past designs or human experts' designs. Future works including more systematic investigation of sample efficiency of the SRL and ARL policy models. After this, integrating the policy models into placement optimization would also be implemented.

6 ACKNOWLEDGMENTS

This work is partially funded by the DARPA IDEA program (HR0011-18-3-0010).

REFERENCES

[1] Thomas D Barrett, William R Clements, Jakob N Foerster, and Alex I Lvovsky. 2019. Exploratory combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1909.04063* (2019).
[2] Irwan Bello, Hieu Pham, Quoc V Le, Mohammad Norouzi, and Samy Bengio. 2016. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940* (2016).
[3] William A Dees and Patrick G Karger. 1982. Automated rip-up and reroute techniques. In *19th Design Automation Conference*. IEEE, 432–439.
[4] Wouter Kool, Herke Van Hoof, and Max Welling. 2018. Attention, learn to solve routing problems! *arXiv preprint arXiv:1803.08475* (2018).
[5] Joseph B Kruskal. 1956. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society* 7, 1 (1956), 48–50.
[6] Koen Lampaert, Georges Gielen, and Willy Sansen. 1996. Analog routing for manufacturability. In *Proceedings of Custom Integrated Circuits Conference*. IEEE, 175–178.
[7] Zhuwen Li, Qifeng Chen, and Vladlen Koltun. 2018. Combinatorial optimization with graph convolutional networks and guided tree search. In *Advances in Neural Information Processing Systems*. 539–548.
[8] Haiguang Liao, Qingyi Dong, Xuliang Dong, Wentai Zhang, Wangyang Zhang, Weiyi Qi, Elias Fallon, and Levent Burak Kara. 2020. Attention Routing: track-assignment detailed routing using attention-based reinforcement learning. *arXiv preprint arXiv:2004.09473* (2020).
[9] Enrico Malavasi and Alberto Sangiovanni-Vincentelli. 1993. Area routing for analog layout. *IEEE transactions on computer-aided design of integrated circuits and systems* 12, 8 (1993), 1186–1197.
[10] Prasanth Mangalagiri. 2019. Analog Layout Synthesis: Are We There Yet?. In *Proceedings of the 2019 International Symposium on Physical Design*. 127–127.
[11] Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Joe Jiang, Ebrahim Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, Sungmin Bae, et al. 2020. Chip Placement with Deep Reinforcement Learning. *arXiv preprint arXiv:2004.10746* (2020).
[12] Mohammadreza Nazari, Afshin Oroojlooy, Lawrence Snyder, and Martin Takác. 2018. Reinforcement learning for solving the vehicle routing problem. In *Advances in Neural Information Processing Systems*. 9839–9849.
[13] Evan Shelhamer, Parsa Mahmoudieh, Max Argus, and Trevor Darrell. 2016. Loss is its own reward: Self-supervision for reinforcement learning. *arXiv preprint arXiv:1612.07307* (2016).
[14] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.
[15] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in neural information processing systems*. 2692–2700.
[16] Jin Zhang, Jiansheng Chen, Yiqing Huang, Weitao Wan, and Tianpeng Li. 2018. Applying Online Expert Supervision in Deep Actor-Critic Reinforcement Learning. In *Chinese Conference on Pattern Recognition and Computer Vision (PRCV)*. Springer, 469–478.
[17] Keren Zhu, Mingjie Liu, Yibo Lin, Biying Xu, Shaolan Li, Xiyuan Tang, Nan Sun, and David Z Pan. 2019. Geniusroute: A new analog routing paradigm using generative neural network guidance. In *Proc. ICCAD*.