A Reinforcement Learning-Based Framework for Solving Physical Design Routing Problem in the Absence of Large Test Sets

Upma Gandhi
University of Calgary
Calgary, Canada
upma.gandhi@ucalgary.ca

Ismail Bustany
Xilinx
San Jose, United States
ismailb@xilinx.com

William Swartz
TimberWolf Systems, Inc
University of Texas at Dallas
Dallas, United States
billswartz7@gmail.com

Dr. Laleh Behjat
University of Calgary
Calgary, Canada
laleh@ucalgary.ca

***Abstract*—Advances in Electronic Design Automation(EDA) methods have made the designers and programmers to search for new ways to solve the complex problems seen in today's Very Large Scale Integration circuits. Machine learning (ML), especially supervised learning, has been used to predict design rule violations. However, supervised learning requires large amount of labeled data. With the competitive nature of EDA based companies, there is limited access to benchmarks and labeled data. In this work, we propose a data-independent reinforcement learning (RL) based routing model called Alpha-PD-Router, which learns to route a circuit and correct short violations. The Alpha-PD-Router is based on a two-player collaborative game model that has been trained on a small circuit and successfully resolves 75 violations in 99 cases of 2 pins net arrangements in the testing phase. The proposed model has the potential to be used as a framework to develop RL based routing techniques untethered by the scarce availability of large routing data samples or designer expertise.**

***Index Terms*—Physical design, routing, machine learning, reinforcement learning, collaborative min-max game theory**

I. Introduction

Routing is one of the important and challenging stages of the physical design of integrated circuits. During routing, the path of the wires in a circuit are determined. The routing problem has traditionally been solved using heuristics and optimization techniques [1]–[3]. Recently, there has been several attempts to solve the problem using machine learning (ML) [4]–[7]. However, the lack of a large number of test cases has been a significant hindrance to obtaining high-quality results.

In this work, we propose a framework to perform routing using RL and collaborative game theory. The proposed technique can learn by performing routing and design rule violation clean up without supervision. Hence, we do not require a large data set or any designer expertise to solve the challenging problems faced during routing.

There are two advantages to our proposed approach: (1) It can produce better quality results after each training iteration as opposed to general path-finding algorithms that generate identical solutions every time and (2) This proposed formulation transcribes the routing problem into a potentially tractable two-player collaborative game problem rather than a multi-player game problem. The main impact of the work is that it shows how ML can be applied in the CAD context without requiring large training data sets. This can enable researchers to use machine learning as an effective tool to deal the complex features and many design rules in lower technology nodes without requiring hard to obtain design IP data. The main contributions of this work are as follows:

- Development of a reinforcement model for routing.
- Development of a reinforcement model for rip-up and reroute
- Designing a collaborative game-theory model for routing and rip-up & reroute.
- Proof of concept on small test cases.

In section II, we describe the routing problem and give a brief background to the machine learning techniques applied to routing congestion estimation. In section III, we introduce the methodology behind our framework and illustrate how to formulate the routing problem as a collaborative two-player game problem. In section IV, we demonstrate how this framework works on proof of concept test cases. In section V, we conclude this manuscript with observations and extensions for future work.

II. Background

A. Physical Design Flow

During the physical design stage, a physical layout with placed circuit elements and routed wires is generated [8]. After some post-processing, corrections, and checks for manufacture-ability, this layout is ready to be printed using lithographic techniques to generate a physical chip. The physical design process is normally divided into 6 steps-Partitioning, Chip Planning, Placement, Clock Tree Synthesis, Signal Routing, and Timing Closure [1]. Routing is one of the last phases in Physical Design. The main purpose of routing is to wire and connect the logic gates [9]. The connections are

This work was supported by the Natural Sciences and Engineering Council of Canada.

called *nets*. The routing solutions need to be produced while respecting wiring resource constraints, the design's timing performance constraints, and a plethora of manufacturing rules and constraints [1] [10].

Currently, combinations of several optimization algorithms are used in leading routers used in industry. Some main methods include single Boolean satisfiability optimization, parallel routing of nets, Differential Fault Analysis etc [11]–[13].

B. Machine learning

Machine Learning(ML) is an effective practical tool to optimize a design in the absence of good models to be used for optimization. ML models are used in a wide variety of applications such as [14]–[17], where they are used to make complex decisions involving large data; tasks that are typically more tedious and time-consuming for humans. At a very high level, ML predictive models are generated using supervised or unsupervised learning paradigms. Artificial neural network(ANN) models are generated using supervised learning approach. ANN's tune their model parameters (a.k.a. hidden layer edge weights) by solving a large-scale nonlinear regression problem on a large labeled sampled feature data set. A Deep neural nets (DNN) is a type of ANN that has many (10's to 100's) hidden layers between the input layer and the output layer and that has been particularly effective in image detection and computer vision problems among others [18]–[20].

One of the main obstacles in using supervised ML-based techniques for solving physical design problems, especially the routing problems, is the lack of large data-sets to learn from. For instance, the only design benchmark test sets that are available to academics are the ISPD 2018 and ISPD 2019 benchmarks which in total have 27 circuits [21] [22].

One way to overcome this problem is to use a reinforcement learning (RL) approach. RL has been successfully applied in many applications. For example, AlphaGo and AlphaGo Zero were developed to play the game of Go [23]. AlphaGo Zero was the inspiration of this research project. To train AlphaGo Zero, the machine's own experience was used. This means that AlphaGo Zero trained itself only from the data provided by self-play in a simulated game environment. The self-play allowed it to exceed human capabilities, and to operate in the combinatorial domain where human expertise may be less competitive. In each game, there is an agent (player in the game) which learns from the environment (board) to take an optimal action (i.e. move) on the basis of a policy (i.e. mapping function of states to actions) to achieve its goal(winning). The structure of AlphaGo Zero was implemented using a generalized RL algorithm [24]–[26]. Deep learning was also used in conjunction with RL to determine an optimal policy [18]–[20].

Monte Carlo Tree Search (MCTS) algorithms are used for selecting the best moves for the players in a game. MCTS provides the move probabilities at each node by using a roll-out scenario of the whole search tree [27]–[29].

III. Methodology

The proposed Alpha-PD-Router is an RL-based routing technique that uses a collaborative min-max game framework and physical design routing algorithms. The proposed framework is inspired by Alpha-Go Zero developed by Google [23] which was able to learn the game of Go without any human intervention.

We cast the routing problem as a two-player collaborative game: the first player is *Router* who performs routing. Router employs a path search algorithm such as A-star [30], [31] to perform initial routing without considering design rule violations. The second player is *Cleaner* which detects design rule violations, selects the best net to rip to fix the violation and rips it. After a net is ripped, Router takes a turn to reroute the net. If no violations exists and all the nets are routed, both Router and Cleaner win and a design rule violation free solution is produced. It should be emphasized that the two players have different strategies and rewards: The first player plays a routing game, and the second player plays a cleanup game. Unlike in the game of Go, our players must collaborate to find a win. A win is when all the nets are routed without any design rule violations.

In the rest of this section, we explain our approach. In Section III-A, we describe our proposed collaborative min-max game formulation for the routing problem. In Section III-B, we explain our terminology and functions of the Cleaner. In section III-C, we expand upon the Router's scheme to produce candidate good routing solutions. In Section III-D, we describe the RL-based training algorithm for the Router and the Cleaner.

A. The Proposed Min-max Game Framework

The Alpha-PD-Router is developed based on a collaborative game model to avoid a Nash Equilibrium [32], [33]. We have developed a min-max game formulation with the main objective divided into two "collaborating" sub-objectives assigned to two players. These two objectives are solved separately by minimizing one and maximizing the other. In the end, both of the sub-objectives work to optimize the main objective [34]–[36]. By using the concept of a min-max game, the aim of building a collaborative model is fulfilled rather than a resource competition model among many resource-demanding nets, the model typically utilized in hitherto routing approaches [37]. As alluded to earlier, the other advantage is that this formulation allows us to cast the routing problem into a potentially tractable two-player game rather than a huge multi-player game where the players count equals the number of nets (e.g. millions).

B. The Cleaner

An initial a set of routes are produced using A* without any consideration to possible violations that can be made. Hence, the input to the Cleaner is a routed solution that may contain violations. In Figure 1, an example of a routed circuit with three 2-pin nets on a 5X5 grid is presented. Each grid tile

contains a single vertical or horizontal routing resource. In this example, net terminals are placed as follows:

- Net-1 (yellow) with source, s1, and destination, d1, at grid locations 1 and 8.
- Net-2 (orange) with source, s2, and destination, d2, at grid locations 14 and 11.
- Net-3 (blue) with source, s3, and destination, d3, at grid locations 20 and 7.

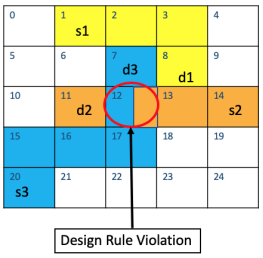

Fig. 1. First Function of the Cleaner is to find short violations for 2-pin nets

1) Detecting Violations: The first function of the Cleaner is to detect violations and tag nets responsible for the violation as candidates to be ripped. To make the training less complex, in Alpha-PD-Router, only considers short violations for now, although the model is flexible to include any type of violations in the future. In the example given in Figure 1, the Cleaner detects an overflow violation at grid location 12 and stores nets 2 and 3 (orange and blue) as candidates to rip.

2) Ripping Candidate Nets: Once violations are detected, the Cleaner rips the minimum number of nets to remove a violation while keeping all the other nets intact. In Figure 1, Net 2 or Net 3 need to be ripped to remove the violation. Figure 2(a), shows the routing solution if the blue net is ripped. Figure 2(b), shows the routing solution if orange net is ripped instead.

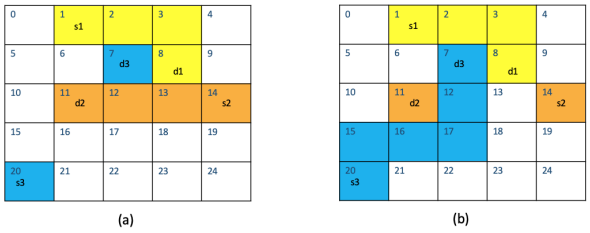

Fig. 2. Second Function of the Cleaner is to rip the candidate nets causing a short violation. 2(a) First candidate: blue blue is ripped . 2(b) Second candidate: orange net is ripped

3) Learning from Router's Reward: The Cleaner rips all the possible net candidates one by one and sends them to be re-routed by the Router. With each re-route, the Router issues a reward to inform Cleaner how good its job was from the Router's perspective. Cleaner aims to maximize these rewards by ripping the nets that make the Router's job easier. It is an RL-based model that consumes the results produced by Router.

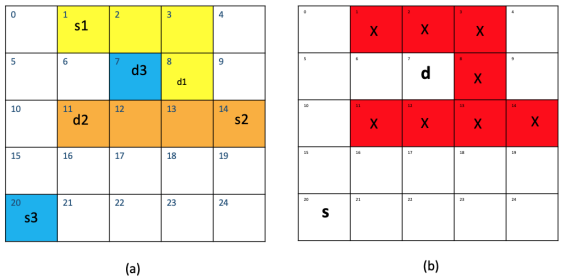

Fig. 3. (a) Input circuit board received from Cleaner (b) Input board converted into a routing format with one single net in focus.

On the basis of the highest rewards, training samples for the Cleaner are saved to be used in testing.

C. The Router

The Router in Alpha-PD-Router is responsible for re-routing the ripped nets without producing any new violations. The solution from the Cleaner is given to the Router. This solution is a partially routed circuit. An example of the output of the Cleaner and the input to the Router is shown in Figure 3(a). In this figure, Net 3 with the blue color is ripped.

1) Router Setup: The Router converts the solution from Cleaner to a routing format in which the focus is only to route the ripped nets. This is done by considering all other nets as blockages to ensure no new violations occur. An illustration is shown in Figure 3 (b) where routed nets, Net 1 and Net 2, are considered as blockages (shown in red). Then, the ripped Net 3 which starts at source, s, and ends at destination, d, is totally rerouted by Router.

2) Routing Environment: Once the routing environment is setup, we need to build the path from the source to destination for each ripped net. To make the routing faster and more efficient, the Router plays a smaller game where all the pins are trying to connect to at least one other pin in the net and the routing grid circuit is considered to be the board. If the pins players meet somewhere in the middle or at source or destination, then a win is declared. A loss is declared when there are no legal moves available for any of the players. This situation can arise when the source and destination of the current net are completely surrounded by the blockages, or all the grids around all the players are already occupied. This setup works as a game, in order for Router to learn the best strategy for routing a net.

For example, for Net 3, two players are considered as shown in Figure 4(a) and (b). In Figure 4(a), player 1 is shown with s1 and d1 as source and destination, respectively. In 4(b), player 2 is shown with s2 and d2 as source and destination.

3) RL-Based Learning: In the proposed Alpha-PD-Router, the methodology behind learning moves that will lead to route source to destination is inspired by the AlphaGo Zero model. The move prediction in Router is optimized by the feedback from the MCTS algorithm to the neural network (NNET) architecture. Each time, one player can move in any of the four

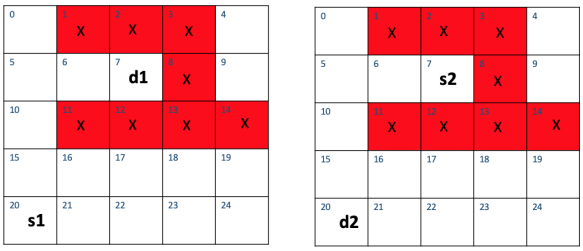

Fig. 4. Dividing original circuit board received in a two-player game (a) Net 1 considered in router part as player 1 with source and destination s1 and d1. (b) Net 2 considered in Router part as player 2 with source and destination s2 and d2.

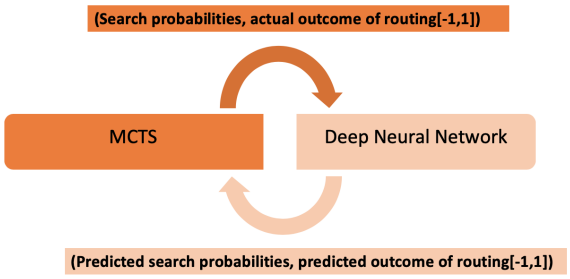

Fig. 5. RL-based Feedback method of the Alpha-PD-Router with MCTS and NNET architecture to improve the win probabilities of each move

directions (North, South, East, West). The circuit board at each move is passed to as input NNET to predict the next move. On the other side, the move probabilities found by MCTS using exploration and exploitation techniques of tree search [27] [28] along with the actual outcome of the routing value win (1) and lose (-1) are compared with move probability and outcome predicted by NNET. The aim of this feedback scheme is to generate solutions with minimum costs based on the feedback between the MCTS and NNET. This relationship is shown in Figure 5. With this RL-based feedback system, the NNET produces a better quality prediction that is used by MCTS to explore better moves ahead.

4) Reward Generation for the Router: With each routing solution, a reward based on the routing length and success of routing is calculated and passed to the Cleaner. The reward function $r(n)$ associated with routing net n is as follows:

$$r(n) = 1 - l(n) * \alpha$$

where $l(n)$ refers to the number grids occupied from source to destination and $\alpha = 0.04$ for the aforementioned 5x5 grid example . In cases of losing scenarios, a large negative reward (e.g. -25, for this example) is added to the final reward to make it a highly undesirable routing outcome when training the NNET. This simple reward function is experimentally tested and is minimized by choosing the shortest route from source to destination.

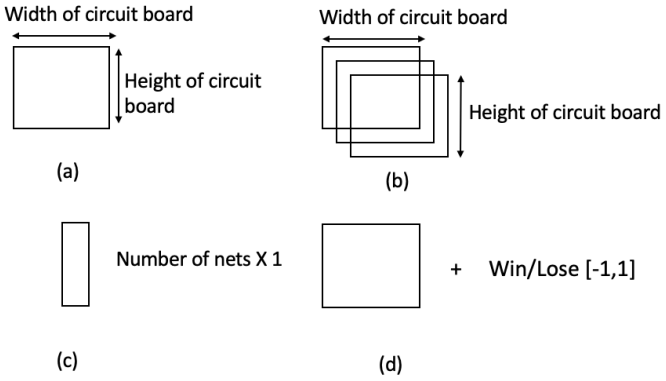

Fig. 6. (a) Input structure of 2D original circuit board to the Cleaner's NNET. (b) Input structure of a 3D matrix to Router's NNET. (c) Output structure of candidate nets to rip vector from the Cleaner's NNET. (d) Output structure of move probabilities and predicted outcome of Router's NNET.

D. Cleaner and Router Training

1) NNET architecture: The NNET layer architecture of the Cleaner and the Router is adopted from [38]. In [38], a Tick Tac Toe game is developed based on AlphaGo Zero terminology. Since the Tick Tac Toe grid structure is similar to the one used in routing, it was experimentally observed that the respective NNET works well with the routing problem structure. The NNET consists of 4 convolution blocks, 2 dense blocks and at the end an activation layer. Each block is completed by batch normalization and relu activation functions. In Router, the activation layer is a combination of softmax function and tanh function because of having 2 separate outputs. Whereas in Ripper, activation layer only considers softmax fuction to generate a single output.

2) Input and Output data: For the Cleaner and the Router, two similar NNET architectures are used to generate candidate solutions. These NNETs are trained separately with different input and output data structures. In Figures 6, the input and output structures of the Cleaner and the Router are presented.

In Figure 6(a), the input structure of the Cleaner is shown which is a 2D matrix with each entry having a crossing net value or zero when empty. Similarly, in Figure 6(b) the input to the Router's NNET is shown. It is a 3D matrix consisting of the value of the current grid's net, grid bin index of source and grid bin index of destination of that net.

The output of the Cleaner's NNET is represented in 6(c) which is a vector of size of the number of nets and contains the score associated with each net. The net with the highest score needs to ripped to resolve a violation. The output of the Router is shown in Figure 6(d) which consists of a 2D matrix with each grid representing move probabilities associated with it and a single win/lose score for the predicting routing games outcome.

To summarise the methodology, in Figure 7, the flow from A-Star to Cleaner to Router and again to Cleaner is shown.

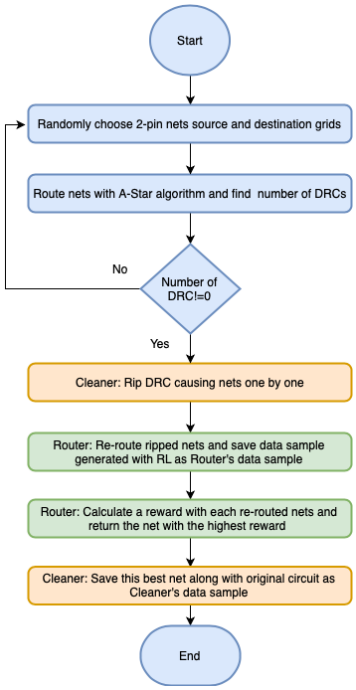

Fig. 7. Alpha-PD-Router RL-based framework to generate Cleaner and Router data samples

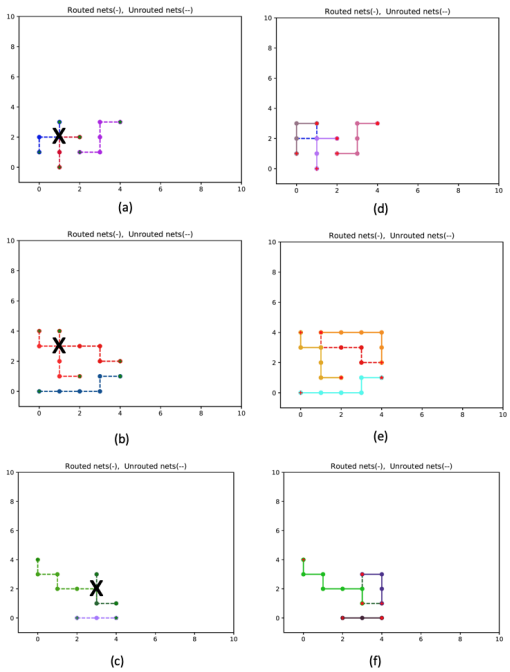

Fig. 8. (a), (b), (c) show sample problems with routing violations. (d),(e), (f) show violation-free solutions determined by the Alpha-PD-Router [39].

IV. Experiments and Results

The Alpha-PD-Router is trained and tested on a 1.3 GHz Intel i5 processor with 16 GB RAM. Python(3.6) programming language is utilized to develop the Alpha-PD-Router. Tensorflow(1.14) and Keras(2.1.6) libraries are to save and load the NNET weights and samples while training and testing. The Alpha-PD-Router is programmed using the Pycharm 2018.2 community edition environment.

As a proof of concept, Alpha-PD-Router has been trained and tested on the aforementioned 2D 5X5 routing grid. The Alpha-PD-Router was originally trained with 157 combinations of 2-pin nets, with 3 nets in each combination. This scenario generated 5480 training samples for the Cleaner and 43400 samples for the Router. The Alpha-PD-Router is tested with 99 test cases. Each test case means a routed circuit with three 2-pin nets. Out of 99 test cases, 95 were new to the Alpha-PD-Router.

With this setup, the Alpha-PD-Router was able to resolve 75 violations. This statement means that the Cleaner was able to take 75 correct decisions to rip a DRC causing net out of all nets.

Furthermore, the Router was able to re-route all those ripped nets and generate DRC free routing solutions. These results show that Alpha-PD-Router was able to learn and train the Cleaner and the Router to make correct decisions from a rather low number of data samples and with little processing power. In Figure 8, the input and output of sample test cases are shown. On the left side, Figures 8(a),(b) and (c) are the routing solutions generated by the A-star algorithm. The respective violation-free routing solutions generated by the Alpha-PD Router are shown on the right side in Figure 8(d),(e) and (f).

The results obtained from these experiments show the efficacy of the proposed framework to be able to route a simple circuit. Moreover it took less than 3 hours to train and test the Cleaner and Router nnet from scratch with the RL generated samples. To improve the accuracy of decisions, we conjecture wee will need more samples which in turn requires more processing power.

V. Conclusions and Future Work

In this work, a routing model called Alpha-PD-Router is presented to correct the short violation in the routing stage. Alpha-PD-Router is one of the first RL-based frameworks which doesn't require any prior knowledge of the problem environment or external data in physical design. We have demonstrated the feasibility of this self-sufficient collaborate game approach in solving simple 2-pin net routing problems.

Our goal is to extend this game theory-based method to generate high-quality solutions on actual routing benchmarks. In the future, the circuit sizes to expand the learning experience and applicability of the work will be increased to obtain better results. It should be mentioned that, once the circuit is increased, extensive processing resources will be needed, as was the case with Google's AlphaGo Zero. Furthermore, an analysis of the strategies that the Router and the Cleaner have learnt will be performed to develop better routing and rip-up and re-route algorithms.

REFERENCES

[1] A. B.Kahng, J. Lienig, I. L.Markov, and J. Hu, *VLSI Physical Design: From Graph Partitioning to Timing Closure*, vol. 1. 2011.

[2] M. N. Ayob, Z. M. Yusof, A. Adam, A. F. Z. Abidin, I. Ibrahim, Z. Ibrahim, S. Sudin, N. Shaikh-Husin, and M. K. Hani, "A particle swarm optimization approach for routing in vlsi," in *2010 2nd International Conference on Computational Intelligence, Communication Systems and Networks*, pp. 49–53, July 2010.

[3] C. Sechen, "Chip-planning, placement, and global routing of macro/custom cell integrated circuits using simulated annealing," in *Proceedings of the 25th ACM/IEEE Design Automation Conference*, DAC '88, (Los Alamitos, CA, USA), pp. 73–80, IEEE Computer Society Press, 1988.

[4] A. F. Tabrizi, L. Rakai, N. K. Darav, I. Bustany, L. Behjat, S. Xu, and A. Kennings, "A Machine Learning Framework to Identify Detailed Routing Short Violations from a Placed Netlist," *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, pp. 1–6, 2018.

[5] A. F. Tabrizi, N. K. Darav, L. Rakai, A. Kennings, and L. Behjat, "Detailed routing violation prediction during placement using machine learning," in *2017 International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*, pp. 1–4, April 2017.

[6] Z. Qi, Y. Cai, and Q. Zhou, "Accurate prediction of detailed routing congestion using supervised data learning," in *2014 IEEE 32nd International Conference on Computer Design (ICCD)*, pp. 97–103, Oct 2014.

[7] P. Tu, C. Pui, and E. F. Y. Young, "Simultaneous reconnection surgery technique of routing with machine learning-based acceleration," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–1, 2019.

[8] N. A. Sherwani, *Algorithms for VLSI Physical Design Automation*. Norwell, MA, USA: Kluwer Academic Publishers, 1993.

[9] J. Ao, S. Dong, S. Chen, and S. Goto, "Delay-driven layer assignment in global routing under multi-tier interconnect structure," pp. 101–107, 2013.

[10] L. Rakai, L. Behjat, S. Areibi, and T. Terlaky, "A multilevel congestion-based global router," *VLSI Des.*, vol. 2009, pp. 6:1–6:1, Jan. 2009.

[11] X. Xu, B. Yu, J.-R. Gao, C.-L. Hsu, and D. Z. Pan, "PARR: Pin-Access Planning and Regular Routing for Self-Aligned Double Patterning," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 21, pp. 42:1—-42:21, may 2016.

[12] M. Khasawneh and P. H. Madden, "HydraRoute: A Novel Approach to Circuit Routing," in *Proceedings of the 2019 on Great Lakes Symposium on VLSI*, GLSVLSI '19, (New York, NY, USA), pp. 177–182, ACM, 2019.

[13] M. Khairallah, R. Sadhukhan, R. Samanta, J. Breier, S. Bhasin, R. S. Chakraborty, A. Chattopadhyay, and D. Mukhopadhyay, "Dfarpa: Differential fault attack resistant physical design automation," in *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 1171–1174, March 2018.

[14] R. Bitton and A. Shabtai, "A machine learning-based intrusion detection system for securing remote desktop connections to electronic flight bag servers," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–1, 2019.

[15] G. P. Herrera, M. Constantino, B. M. Tabak, H. Pistori, J.-J. Su, and A. Naranpanawa, "Long-term forecast of energy commodities price using machine learning," *Energy*, apr 2019.

[16] J. Salminen, V. Yoganathan, J. Corporan, B. J. Jansen, and S.-G. Jung, "Machine learning approach to auto-tagging online content for content marketing efficiency: A comparative analysis between methods and content type," *Journal of Business Research*, vol. 101, pp. 203–217, aug 2019.

[17] R. Jinnouchi, F. Karsai, and G. Kresse, "On-the-fly machine learning force field generation: Application to melting points," tech. rep., 2019.

[18] Deep AI, "Neural Network Definition — DeepAI."

[19] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," *CoRR*, vol. abs/1312.5602, 2013.

[20] R. S. Sutton and A. G. Barto, *Reinforcement Learing: An Introduction*. The MIT Press, Cambridge, Massachusetts, London, second edi ed.

[21] S. Mantik, G. Posser, W.-K. Chow, Y. Ding, and W.-H. Liu, "Ispd 2018 initial detailed routing contest and benchmarks," in *Proceedings of the 2018 International Symposium on Physical Design*, ISPD '18, (New York, NY, USA), pp. 140–143, ACM, 2018.

[22] W.-H. Liu, S. Mantik, W.-K. Chow, Y. Ding, A. Farshidi, and G. Posser, "Ispd 2019 initial detailed routing contest and benchmark with advanced routing rules," in *Proceedings of the 2019 International Symposium on Physical Design*, ISPD '19, (New York, NY, USA), pp. 147–151, ACM, 2019.

[23] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. Van Den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of Go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[24] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, (Cambridge, MA, USA), pp. 1057–1063, MIT Press, 1999.

[25] J. Zheng and A. Siami Namin, "A markov decision process to determine optimal policies in moving target," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, (New York, NY, USA), pp. 2321–2323, ACM, 2018.

[26] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, feb 2015.

[27] Z. Liu, M. Zhou, W. Cao, Q. Qu, H. W. F. Yeung, and V. Y. Y. Chung, "Towards understanding chinese checkers with heuristics, monte carlo tree search, and deep reinforcement learning," *CoRR*, vol. abs/1903.01747, 2019.

[28] C. B. Browne, P. Edward, D. Whitehouse, S. M.Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A Survey of Monte Carlo Tree Search Methods," *IEEE Transactions on Computational Intelligence and AI in games*, vol. VOL. 4, no. NO. 1.

[29] G. Chaslot, S. Bakkes, I. Szita, and P. Spronck, "Monte-Carlo Tree Search: A New Framework for Game AI," in *Fourth Artificial Intelligence and Interactive Digital Entertainment Conference*, (Palo Alto, California), 2008.

[30] Xiang Liu and Daoxiong Gong, "A comparative study of a-star algorithms for search and rescue in perfect maze," in *2011 International Conference on Electric Information and Control Engineering*, pp. 24–27, April 2011.

[31] I. Chaari, A. Koubaa, H. Bennaceur, A. Ammar, M. Alajlan, and H. Youssef, "Design and performance analysis of global path planning techniques for autonomous mobile robots in grid environments," *International Journal of Advanced Robotic Systems*, vol. 14, no. 2, p. 1729881416663663, 2017.

[32] J. F. Nash, "Equilibrium points in n-person games," *Proceedings of the National Academy of Sciences*, vol. 36, no. 1, pp. 48–49, 1950.

[33] C. A. Holt and A. E. Roth, "The Nash equilibrium: A perspective," *Proceedings of the National Academy of Sciences*, vol. 101, no. 12, pp. 3999–4002, 2004.

[34] P. O. M. Scokaert and D. Q. Mayne, "Min-max feedback model predictive control for constrained linear systems," *IEEE Transactions on Automatic Control*, vol. 43, pp. 1136–1142, Aug 1998.

[35] H. Aissi, C. Bazgan, and D. Vanderpooten, "Min–max and min–max regret versions of combinatorial optimization problems: A survey," *European Journal of Operational Research*, vol. 197, no. 2, pp. 427–438, 2009.

[36] C. Jin, P. Netrapalli, and M. I. Jordan, "What is local optimality in nonconvex-nonconcave minimax optimization?," 2019.

[37] M. Khasawneh and P. H. Madden, "Hydraroute: A novel approach to circuit routing," in *Proceedings of the 2019 on Great Lakes Symposium on VLSI*, GLSVLSI '19, (New York, NY, USA), pp. 177–182, ACM, 2019.

[38] S. Thakoor, S. Nair, and M. Jhunjhunwala, "Learning to Play Othello Without Human Knowledge," 2016.

[39] U. Gandhi, *A Reinforcement Learning-Based Framework to Generate Routing Solutions and Correct Violations in VLSI Physical Design*. Unpublished thesis type, Univeristy of Calgary, 2019.