Placement in Integrated Circuits using Cyclic Reinforcement Learning and Simulated Annealing

Dhruv Vashisht, Harshit Rampal,
Haiguang Liao, Yang Lu,
Devika Shanbhag, Elias Fallon,
Levent Burak Kara

Overview

Physical design and production of integrated circuits (IC) has become more challenging with the increase in sophisticated IC technology. Partition, analytical and annealing-based placers have been enriching the placement solution toolbox. However, long run time and lack of the ability to generalize restrict existing placement tools. We devise a learning-based placement tool based on cyclic application of reinforcement learning (RL) and simulated annealing (SA) by leveraging the advancement of RL.

Our results show that:

- Our method is majorly different with its combination of RL and SA
- The RL module is able to provide a better initialization for SA
- It leads to a better final placement design
- It leverages the RL model's ability to quickly get a good rough solution after training and the heuristics' ability to realize greedy improvements in the solution

Cyclic Reinforcement Learning

A cyclic model employs both RL and SA. While the RL agent learns locally by computing differences between perturbed states, it also learns globally from the SA's output which serves as the state space's horizon. Global reward r_g serves as the approximated value of the r^{th} (final) step V_r during the agent's training as stated in equations below:

$$V_t = \sum_{i=t}^{r-1} \gamma^i r_{li} + \sum_{i=r}^{\infty} \gamma^i r_{li} = \sum_{i=t}^{r-1} \gamma^i r_{li} + V_r$$

$$r_g = V_t - \sum_{i=t}^{r-1} \gamma^i r_{li}$$

The actor chooses perturbations with maximum reward. The critic computes the expected reward for each state-action pair. Over n epochs, the RL agent learns to generate a state closer to the global minimum compared to the initial random solution. The SA converges to a better optimized final solution with this improved initialization.

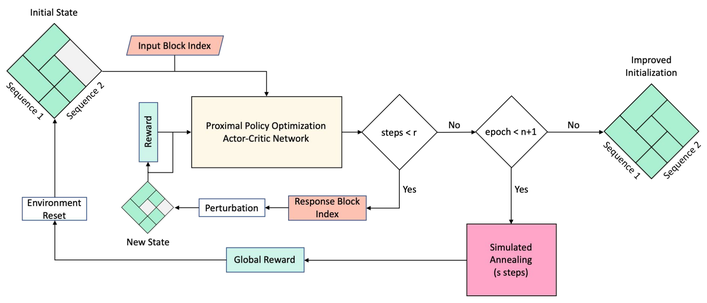

Figure 1: Algorithm for cyclic reinforcement learning and simulated annealing for placement

Experiments

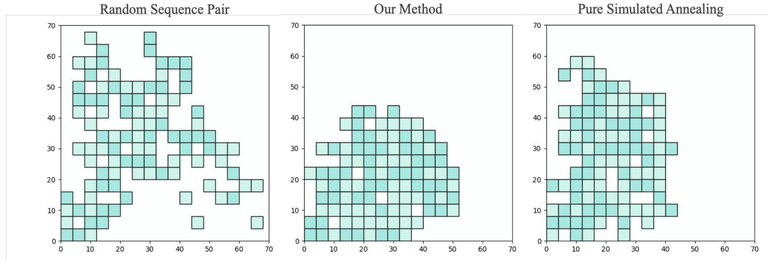

Figure 2: Placements for the lattice problem after 5000 steps of Simulated Annealing (SA). For the same number of SA steps, RL initialized placements (middle) are more compact than those initialized randomly (right)

The proposed model is tested on the lattice layout and ami49 benchmark as illustrated in Figure 2 and Figure 3 respectively. In both problems, SA achieves a better solution when initialized by our method.

The lattice layout consists of 100 blocks having the same dimensions. The RL agent is trained for 200 steps while SA takes 5000 steps in each of the three configurations.

The same combination of steps for RL and SA steps is used on the ami49 benchmark, consisting of 49 blocks, having unequal dimensions. The first row is a configuration with 3 fixed blocks while the configuration in the second row has no fixed blocks.

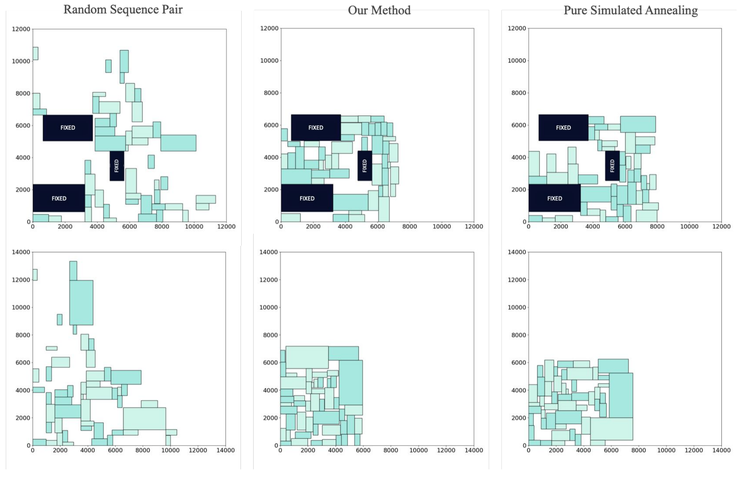

Figure 3: Placements for ami49 benchmark. 3 blocks are assumed fixed in the top row. All blocks are free in the bottom row. For the same number of SA steps, RL initialized placements (middle) are more compact than those initialized randomly (right).

Results

Preliminary results reveal that the agent is able to outperform the baseline on both benchmarks. Table 1 below summarizes the average costs of solutions from both methods on the lattice problems of different sizes. For all of them, our proposed methods achieved lower average cost compared with the baseline pure SA with random initialization. For 100 blocks lattice problem, the standard deviation for RL Initialization is 146 and that of random initialization is 133.

Number of Blocks	RL Initialization	Random Initialization
100	3,386	3,532
225	14,348	14,844
400	38,060	38,612
625	85,329	86,064

Table 1: Average cost (in μm) of 10 experiments on Lattice block list

For the ami49 results, Fig. 3 presents the final placement and Table 2 compares the average cost of solutions from both the methods. The ami49 benchmark has two configurations:
Configuration 1 - Three fixed blocks (Fig. 3 top row)
Configuration 2 - No fixed blocks (Fig. 3 bottom row)

Configuration 1 has pre-specified coordinates to be maintained in the final placement making it more challenging as the agent tries to make perturbations given fixed blocks. However, the results reveal that the agent is still able to outperform the baseline (pure SA with random initialization). Configuration 2 consisting of no fixed blocks also shows our method's improvement from the baseline. The standard deviation for RL Initialization and random initialization indicates small variance of results and actual improvement of RL Initialization.

Type of Blocks	RL Initialization	Random Initialization
w/ Fixed Blocks	49	53
w/o Fixed Blocks	43	46

Table 2: Average cost (in mm^2) of 10 experiments on ami49 block list

References

Mirhoseini, Azalia, et al. (2020). "Chip Placement with Deep Reinforcement Learning". In: arXiv preprint arXiv:2004.10746

Cai, Qingpeng, et al. (2019). "Reinforcement Learning Driven Heuristic Optimization". In: arXiv preprint arXiv:1906.06639

Markov, Igor L., Jin Hu, and Myung-Chul Kim (2015). "Progress and challenges in VLSI placement research". In: Proceedings of the IEEE 103.11.

H.Murata,K.Fujiyoshi,M Kaneko (1998). "VLSI/PCB placement with obstacles based on sequence pair.". In: IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems Volume 17 Issue 1.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov (2017). "Proximal Policy Optimization algorithms". In: arXiv preprint arXiv:1707.06347.

https://s2.smu.edu/ manikas/Benchmarks/Block/ami49.yal