2024 IEEE International Symposium on Electromagnetic Compatibility, Signal & Power Integrity (EMC+SIPI) | 979-8-3503-6039-4/24/$31.00 ©2024 IEEE | DOI: 10.1109/EMCSIPI49824.2024.10705608

Graph Convolutional Neural Network Assisted Genetic Algorithm for PDN Decap Optimization

Haran Manoharan[#1], Jack Juang[#2], Ling Zhang[$3], Hanfeng Wang[*4], Jingnan Pan[*5], Kelvin Qiu [*6], Xu Gao[*7] and Chulsoon Hwang[#8]

[#]*EMC Laboratory, Missouri University of Science and Technology, Rolla, Mo, USA*
[1]*hm6h6,* [2]*jjryb,* [8]*hwangc@mst.edu*
[*]*Google Inc, Mountain View, CA, USA*
[4]*hanfengw,* [5]*jingnan,* [6]*kqiu,* [7]*xugao@google.com*
[$]*Zhejiang University, Hangzhou, China*
[3]*lingzhang_zju@zju.edu.cn*

***Abstract*—This paper proposes a hybrid algorithm combining reinforcement learning (RL) and a genetic algorithm (GA) for PDN decap optimization. The trained RL agent uses a graph convolutional neural network as a policy network and predicts the decap solution for a given PDN impedance and target impedance, which is seeded as an initial population to the GA. The trained RL agent is scalable regarding the number of decap ports. The main goal is to save computation time and find the near global minimum or global minimum. Generalization of the algorithm to different decap libraries is achieved through transfer learning, eventually reducing the training time of the RL agent. The proposed algorithm finds a decap solution satisfying target impedance twice as fast compared with genetic algorithms.**

Keywords—graph convolutional neural network, transfer learning, decap optimization, genetic algorithm

I. Introduction

Decap Optimization is a challenging combinatorial problem. Machine learning offers a powerful approach to combinatorial problems by handling complex combinations efficiently, learning from data to uncover hidden patterns, adapting to changing problem dynamics, and potentially discovering novel solutions. Several works using machine learning techniques [1- 4] have been conducted to find a better decap solution with less computation time.

In [1], a transformer network-based RL algorithm was proposed to find the optimal decap port for placement. The algorithm is scalable in terms of decap ports, but the algorithm was able to handle only one decap. In [2], a deep reinforcement learning-based algorithm using the actor-critic method was proposed for decap optimization, but this method was not generalizable to different PCB boards. In [3], Q learning-based reinforcement learning was proposed to optimize the decaps for solid-state drives, yet this method was also not generalized. Having a generalized model for different decap libraries is quite challenging without re-training the model, and evolutionary algorithms like genetic algorithms excel in this problem. A genetic algorithm is a population-based algorithm that does not require training.

Works in [4-7] used a genetic algorithm alone to predict the decap solution for a given PDN impedance and target impedance. It is known that initial population is one of the most critical factors determining the faster convergence and eventually finding the global minimum. Most of the algorithms developed so far start with a random initial population or augmented initial population [5-7]. In [5], the proportions of decaps in the decap library needed for specific boards and their target impedance are computed and used as weights to generate the initial population. A reinforcement learning agent was added to [5] in [6] to tune the mutation probability, which helps the algorithm not get stuck in local minima. In [7], the algorithm's initial population was improved by having a disproportionate initial population to help converge the algorithm faster for loose target impedances. This algorithm was also designed to minimize the cost of the decaps used. The algorithms [5-7] are computationally expensive and might get stuck in local minimum if their hyperparameters like population size and number of generations are not selected properly.

This work proposes a scalable reinforcement learning agent that uses graph convolutional neural networks as a policy network to predict the decap solution for a given board impedance and target impedance. This solution is seeded to the GA used in [6] to fine-tune and find the near or true global minimum. The trained RL agent works well for the decap library it is trained on. To make the algorithm generalizable, this work also leverages transfer learning, where the pre-trained model is fine-tuned to the new decap library to predict the solution based on the new decap library.

II. Previous Work

A. Augmented GA V1 & V2

GA is a population-based optimization algorithm that is used to solve combinatorial problems. In [5], augmented GA V1 was proposed. Based on the PDN impedance at the last frequency point for the R type and both the transition and last frequency point for the RL type, target impedance decap type weights are generated. These weights are used to generate the initial population that will lead the GA to near global minimum. The results in [5] showed faster convergence time and better solution quality than normal GA. The decap solution was encoded as a set of real numbers [5] based on the decap library. Three mutation operators were introduced, and crossover operation was removed from the GA process.

In [6], it was found that the GA in [5] was heavily dependent on mutation probability (hyperparameter of GA). Hence, an RL agent was used to tune the mutation probability of the GA for every 5 generations. The RL agent first explores the action space (the mutation probability) by taking random actions. Then, in the exploitation phase, the agent takes action that will lead to a positive reward. This way, the GA will not get stuck at the local minimum and will try to find the near global minimum possible. Overall, the results in [6] were better in terms of solution quality (number of decaps) and time cost. In this work, the GA proposed in [6] is used.

This work was supported in part by the National Science Foundation (NSF) under Grant No. IIP-1916535.

B. *Augmented GA V3*

In [7], the initial population was further improved by having a disproportionate initial population. This was made to ensure that the population diversity of the GA is maintained. Population diversity is an important factor in GA that determines the flow of convergence and solution quality. Instead of filling all the decap ports for all the solutions in the initial population, the population consists of 100%, 80%, 60%, and 40% filled decap ports as solutions. This made the diversity of solutions unique and gave a faster convergence rate for boards with loose target impedance. This algorithm was improved to handle decap cost as another objective, finding a solution with a minimum number of decaps and less decap cost.

III. Graph Neural Network-based RL

Graph neural networks (GNNs) are a type of neural network designed to operate on graph structures directly [8]. They are particularly effective in learning from data that is represented as graphs, a common structure in many real-world applications, including social networks, molecular chemistry, and physical systems. There are three critical components for graphs:

1. Nodes: Represent entities or elements of the graph. In our case, each port in the impedance matrix is a node.

2. Edges: Define the relationships or interactions between nodes. In the impedance matrix, these represent the connections between different ports. Fig. 1 shows the graph representation of a 50 decap port example case. There are 51 ports, including one IC port.

3. Node Features: Attributes or properties of each node. Here, impedance values at different frequency points and target impedance are used as node features.

GNNs perform computations at the node level, applying the same operation across all nodes regardless of the total number of nodes in the graph. This means that the network can seamlessly process graphs with varying numbers of nodes [9]. In this algorithm, a graph convolutional network (GCN) is used. It is a type of GNN that generalizes convolutional neural networks (CNNs) to graph-structured data. It is particularly suited for this decap optimization task because it can efficiently handle the variable-sized input (different numbers of ports) and capture the complex relationships in the impedance matrix. Each port's impedance across frequency points and the corresponding target impedance are encoded as node features.

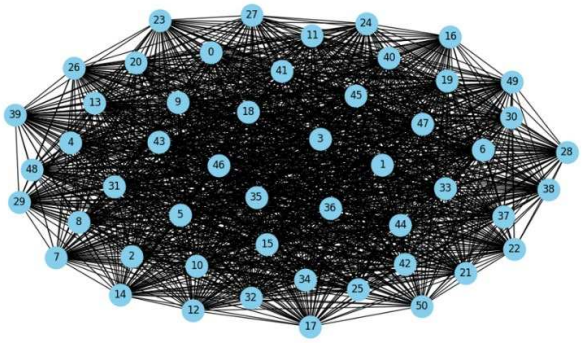

Fig. 1. Graph representation of impedance matrix (50 decap port example case).

The GCN processes these features through its layers, considering the graph's topology (how ports are interconnected). Here, the output of the GCN is connected to the feedforward neural network and outputs a set of

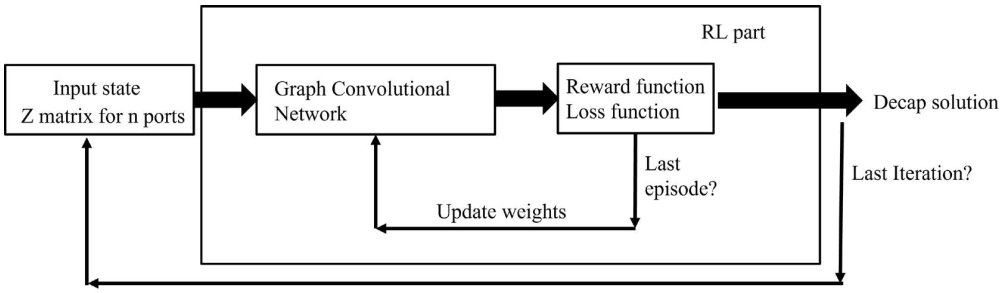

Fig. 2. RL agent training process.

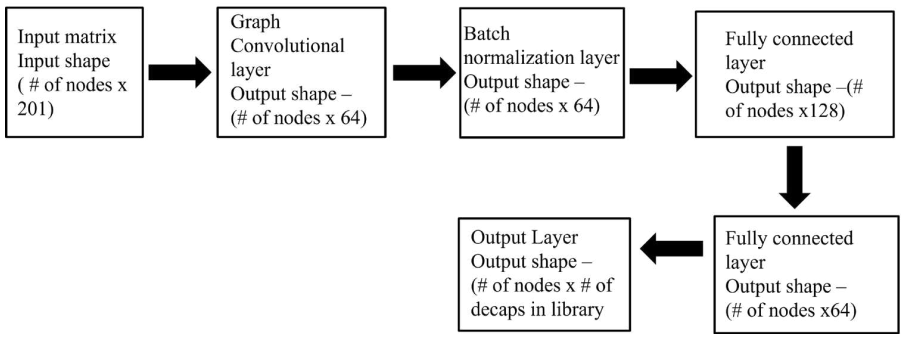

Fig. 3. Graph neural network architecture

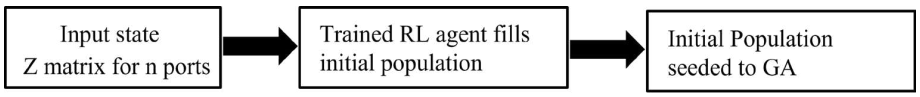

Fig. 4. Proposed hybrid algorithm flow

probabilities for each decap option at each port. These probabilities guide the RL agent in choosing the optimal decap placements to meet the impedance targets with minimal usage of decaps.

A. Training

Training is essential for the neural network to learn the features of the task. Fig. 2 shows the training process of the proposed algorithm. The inputs (the impedance matrix and target impedance) are encoded as graphs. The features are fed to the neural network to give out probabilities of decaps in the decap library for all the ports. If there are m number of decaps and n number of ports, the output is $n \times m$ probabilities.

Based on the probabilities, the RL agent places the decaps and sees the reward for the decap solution. For a solution satisfying target impedance, the reward is given as (1), and for not satisfying target impedance, the reward is given as (2) [5].

$$\text{Reward} = \text{-(Total \# of Ports - \# of Ports Used)} \tag{1}$$

$$\text{Reward} = \max\left(\frac{\text{solution\_z(f) - target\_z(f)}}{\text{target\_z(f)}}\right) \tag{2}$$

The rewards are backpropagated through REINFORCE [1], and the neural network weights are updated. The neural network architecture used as a policy network is shown in Fig. 3. The total number of episodes was 200, and the first 120 episodes were used for the exploration and the remaining for the exploitation phases. This process was done for different decap boards and different target impedances. The training was done for ten 50 decap port cases and ten 75 decap port cases. The total training time was 5 hours. The test cases are arbitrarily shaped boards generated, and their impedances were obtained using the boundary element method and the node voltage method [8]. For each case, the target impedance was varied from loose to tight target impedance.

B. RL+GA

Fig. 4 shows the overall proposed hybrid algorithm framework. The RL agent predicts the decap solution for a given input matrix and target impedance. This process is repeated to fill the initial population. If the initial population of the GA is 20, then the process is repeated 20 times. It is to be noted that predicting a decap solution single time by the trained RL agent, the time taken is in milliseconds. Then, the GA used in [6] is used to fine-tune and find the solution with a minimum number of decaps. The decap library used is shown in Table I.

IV. Comparison

A comparison of the proposed approach to previous works was carried out for different test cases. The test cases included one 25 decap port, two 50 decap ports, two 75 decap ports, and three 150 decap port cases. The target impedances of the cases were varied to test the efficiency of the proposed algorithm. The impedances of the test cases were generated using the node voltage method [8]. All the algorithms used the same decap library as in Table I and the exact system specification. Table II compares the proposed approach to the other approaches [5-7]. The comparison of the solution quality is plotted in Fig. 5. The population size and number of generations for all the algorithms were set to 20 and 100, respectively.

TABLE I. Decap Library

Decap #	***Name***	***Cap. (uF)***	***ESL (nH)***
1	GRM31CR60J227ME11	220	0.3
2	GRM32EC80E337ME05	330	0.4
3	GRM033C80J104KE84	0.1	0.2
4	GRM033R60J474KE90	0.47	0.19
5	GRM155B31C105KA12	1	0.2
6	GRM155C70J225KE11	2.2	0.2
7	GRM185C81A475KE11	4.7	0.27
8	GRM188B30J226MEA0	22	0.22
9	GRM219D80E476ME44	47	0.25

The proposed approach can generally find a better solution than the previous works within less computation time. In 75 decap ports case #1, the algorithm found a solution of 13 decaps in 91 seconds, while [7] gave a solution of 14 decaps in 173 seconds. In all the 150 decap ports test cases, the proposed approach found a better solution than the algorithm in [7]. In terms of time cost for all these 150 decap port cases, the proposed approach is 3 times faster than the algorithm used in [7]. With the RL agent predicting the initial population, the algorithm converges faster and hence can explore more to find better solutions available. As the number of ports increases, matrix manipulations take significant computation time for each decap solution in the population. In the proposed approach, since the number of generations required to converge is less than others, the computation time is less. Hence, with less computation time, the proposed approach is able to find better solutions.

TABLE II. Comparison of Solution Quality

Test Case #	**Minimum # of decaps**			
	This work	***[5]***	***[6]***	***[7]***
25 decap ports	**5**	6	6	5
50 decap ports #1	**13**	16	15	15
50 decap ports #2	**22**	25	24	22
75 decap ports #1	**13**	18	16	14
75 decap ports #2	**14**	17	15	14
150 decap ports #1	**43**	51	45	44
150 decap ports #2	**32**	39	34	34
150 decap ports #3	**14**	20	16	16

TABLE III. COMPARISON OF TIME COST

Test Case #	Time Taken (seconds)			
	This work	*[5]*	*[6]*	*[7]*
25 decap ports	**4**	42	40	30
50 decap ports #1	**27**	350	340	290
50 decap ports #2	**28**	357	363	260
75 decap ports #1	**90**	349	284	180
75 decap ports #2	**91**	343	274	173
150 decap ports #1	**724**	3540	2884	2040
150 decap ports #2	**543**	2879	2554	1854
150 decap ports #3	**467**	2634	2056	1456

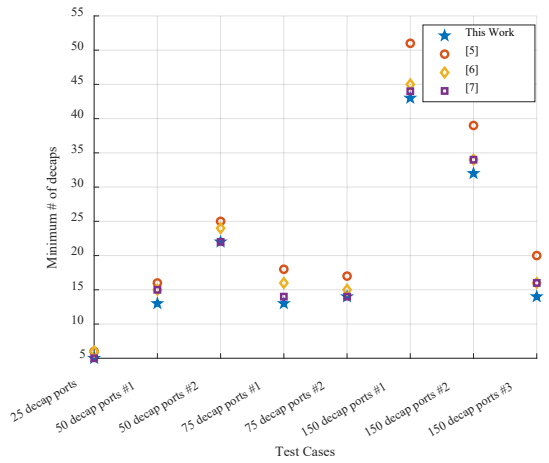

Fig. 5. Comparison of this work to [5-7] in terms of solution quality

V. TRANSFER LEARNING

The proposed algorithm works well if the decap library is not changed. But the algorithms in [5-7] can work for any decap library. Instead of re-training the model for a different decap library, this work uses the approach of transfer learning for this task. Transfer learning (TL) is a powerful technique in machine learning where knowledge gained while solving one task is applied to a different but related task [10].

Here, the task of placing the decaps is the same; only the decap library changes. Another challenge in the domain is the change in output size. The output size of the neural network is $n \times m$, where m is the number of decaps in the library, and for a new decap library with size m_new, the output size should be different $n \times m\_new$. In this proposed approach, the weights of all the layers except the output layer are not re-trained. Only the last layer of the neural network weights in Fig. 3 is updated in the training process. Hence, the training time is less than re-training the model from the start. The initial training time was 5 hours, but with transfer learning, the re-training to the new decap library is 30 minutes.

The proposed transfer learning approach is tested on 3 different test cases with an entirely different decap library (d decaps) other than the one used in section IV, as given in Table IV.

TABLE IV. NEW DECAP LIBRARY

Decap #	*Name*	*Cap. (uF)*	*ESL (nH)*
1	GRM152R60J104KE19	0.1	0.22
2	GRM152R60J474ME15	0.47	0.23
3	GRM035R60E475ME01	4.7	0.28
4	GRM033R60G225ME44	2.2	0.27
5	GRM032R60G105ME05	1	0.16
6	GRM152R60G105ME15	1	0.2

The training process is the same as the previous one, but the number of training cases is less. The time taken for the fine training of the model on the different decap library is 30 minutes. The approach is tested with other works that can work for any decap library [5-7]. Tables V & VI compare the proposed approach with [5-7] regarding solution quality and time cost, respectively. The comparison in terms of solution quality to [5-7] is plotted in Fig. 6.

TABLE V. COMPARISON OF SOLUTION QUALITY (TL PART)

Test Case #	Minimum # of decaps			
	This work	*[5]*	*[6]*	*[7]*
50 decap ports	**20**	22	21	21
75 decap ports #1	**42**	44	44	43
75 decap ports #2	**18**	19	19	18

TABLE VI. COMPARISON OF TIME COST (TL PART)

Test Case #	Time Taken (seconds)			
	This work	*[5]*	*[6]*	*[7]*
50 decap ports	**24**	369	357	267
75 decap ports #1	**160**	578	563	358
75 decap ports #2	**130**	363	356	262

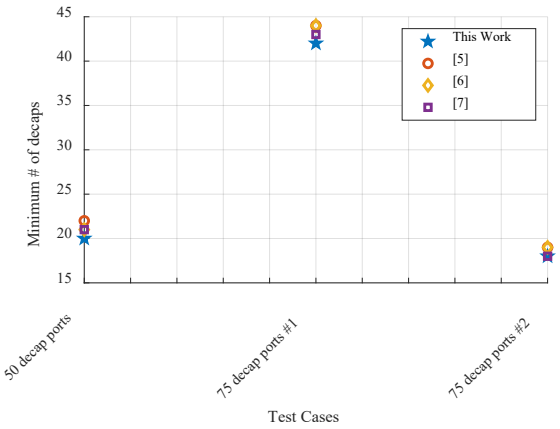

Fig. 6. Comparison of this work to [5-7] in terms of solution quality (Transfer learning part)

With 30 minutes of training, the algorithm was fine-tuned to the new decap library and produced results comparatively better than the previous works. In the case of 75 decap ports #1, the proposed approach found a better solution of 42 decaps within 160 seconds. Overall, the

proposed approach is comparatively faster and has improved solution quality.

VI. Conclusion

With a graph neural network as a policy network, the proposed RL agent is scalable and can adapt to any number of decap ports. The decap solution produced by the RL agent will eventually lead the GA to the global or near-global minimum and thus reduces the convergence time of the GA. For the 75 decap port cases tested, the proposed algorithm takes half the computation time taken by the algorithm in [7]. With the use of transfer learning, the neural network is adapted to a new decap library and performs better in terms of solution quality and time than other previous works. In the future, the algorithm can be further improved to predict decap solutions without the need for a genetic algorithm. This will significantly reduce the computation time.

References

[1] H. Park et al., "Deep Reinforcement Learning-Based Optimal Decoupling Capacitor Design Method for Silicon Interposer-Based 2.5-D/3-d ICs," *IEEE Trans. Compon. Packaging Manuf. Technol.*, vol. 10, no. 3, pp. 467-478, March 2020, doi: 10.1109/TCPMT.2020.2972019.

[2] L. Zhang, W. Huang, J. Juang, H. Lin, B. -C. Tseng and C. Hwang, "An Enhanced Deep Reinforcement Learning Algorithm for Decoupling Capacitor Selection in Power Distribution Network Design," *2020 IEEE International Symposium on Electromagnetic Compatibility & Signal/Power Integrity (EMCSI)*, 2020, pp. 245-250, doi: 10.1109/EMCSI38923.2020.9191512.

[3] J. Shin et al., "Reinforcement Learning-Based Decap Optimization Method for High-Performance Solid-State Drive," *2021 IEEE International Joint EMC/SI/PI and EMC Europe Symposium*, Raleigh, NC, USA, 2021, pp. 718-721, doi: 10.1109/EMC/SI/PI/EMCEurope52599.2021.9559162.

[4] F. De Paulis et al., "A Methodical Approach for PCB PDN Decoupling Minimizing Overdesign with Genetic Algorithm Optimization," *2022 IEEE International Symposium on Electromagnetic Compatibility & Signal/Power Integrity (EMCSI)*, Spokane, WA, USA, 2022, pp. 238-243, doi: 10.1109/EMCSI39492.2022.9889490

[5] J. Juang *et al*., "Augmented Genetic Algorithm for Decoupling Capacitor Optimization in PDN Design Through Improved Population Generation," submitted to *IEEE Transactions on Signal and Power Integrity*.

[6] H. Manoharan et al., "Augmented Genetic Algorithm v2 with Reinforcement Learning for PDN Decap Optimization," *2023 IEEE Symposium on Electromagnetic Compatibility & Signal/Power Integrity (EMC+SIPI)*, Grand Rapids, MI, USA, 2023, pp. 255-258, doi: 10.1109/EMCSIPI50001.2023.10241752.

[7] H. Manoharan et al., " Augmented Genetic Algorithm V3 for Multi-Objective PDN Decap Optimization," submitted to *2024 IEEE Joint International Symposium on Electromagnetic Compatibility, Signal & Power Integrity: EMC Japan/Asia- Pacific International Symposium on Electromagnetic Compatibility*.

[8] L. Zhang et al., "Efficient DC and AC Impedance Calculation for Arbitrary-Shape and Multilayer PDN Using Boundary Integration," in *IEEE Transactions on Signal and Power Integrity*, vol. 1, pp. 1-11, 2022, doi: 10.1109/TSIPI.2022.3164037.

[9] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang and P. S. Yu, "A Comprehensive Survey on Graph Neural Networks," in *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4-24, Jan. 2021, doi: 10.1109/TNNLS.2020.2978386.

[10] Z. Zhu, K. Lin, A. K. Jain and J. Zhou, "Transfer Learning in Deep Reinforcement Learning: A Survey," *in IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 11, pp. 13344-13362, 1 Nov. 2023, doi: 10.1109/TPAMI.2023.3292075.