2023 IEEE Symposium on Electromagnetic Compatibility & Signal/Power Integrity (EMC+SIPI) | 979-8-3503-0976-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/EMCSIPI50001.2023.10241752

Augmented Genetic Algorithm v2 with Reinforcement Learning for PDN Decap Optimization

Haran Manoharan[#1], Jack Juang[#2], Hanfeng Wang[*3], Jingnan Pan[*4], Kelvin Qiu [*5], Xu Gao[*6], and Chulsoon Hwang[#7]
[#]*EMC Laboratory,* Missouri University of Science and Technology, Rolla, Mo, USA
[*]*Google LLC, Mountain View, CA, USA*
[1]hm6h6, [2]jjryb, [7]hwangc@mst.edu, [3]hanfengw, [4]jingnan, [5]kqiu, [6]xugaon@google.com

***Abstract*— Genetic algorithms (GAs) use many hyperparameters, and tuning these parameters can determine the optimization performance. A GA with an augmented initial population was proposed for decap optimization but it had convergence issues by getting stuck in the local minimum. This work uses a reinforcement learning (RL) approach to adaptively tune the hyperparameters of GA during its operation. With this approach, the agent tries to change the parameters so that the GA does not get stuck in the local minimum. The proposed method combining the RL agent and Augmented GA showed better performance in terms of solution quality and time cost. Overall, in all the cases tested, the proposed method showed better performance than the Augmented GA without RL.**

Keywords—Reinforcement Learning (RL), Genetic Algorithm, Augmented Genetic Algorithm, Decap Optimization

I. Introduction

In a power distribution network (PDN), selecting decoupling capacitors (decaps) is essential in suppressing power supply ripples. Optimization of decaps is vital in saving cost and layout space. There are several works done in optimizing placement and the value of decaps using various approaches [1].

In [2], a deep reinforcement learning agent is used to optimize the decap placement and value. Here proximal policy optimization (PPO) algorithm was used. The DNN was trained with different boards having the same stackup but different shapes and decap locations. The agent was able to provide optimal results for a board with a different shape but with the same stackup. In [3], transformer network-based deep reinforcement learning is used for decap placement. This attention-based transformer network was used to parameterize optimization policy. Here, in addition to self-impedance, transfer impedance was also considered. An advantage actor-critic reinforcement learning-based method is proposed in [4], which has a large action space for decap optimization. All three methods have issues with generalization. They fail to produce optimal results for any new PCB that has never been used for training.

On the other hand, evolutionary algorithms do not require training, and they are generalized. Among the various evolutionary algorithms, the genetic algorithm is widely used for combinatorial optimization problems because of its population-based approach. Conventional GA is modified in [5] called Gene suppressed GA to make the GA converge faster. In [6], the Gene suppressed GA was further improved with an augmented initial population named augmented GA, and new genetic operators are introduced. Even though the algorithm showed excellent performance in terms of solution quality and time cost compared to other algorithms, it tends to have convergence issues by getting stuck in a local minimum. Tuning the hyperparameters of the GA is vital in getting faster convergence. Various techniques have been employed to adjust the parameters, including meta-EA [7] and design of experiments (DOE) [8]. In the design of experiments, every possibility of the hyperparameters is tried, and this process is computationally expensive and time-consuming. Hence a more generalized and computationally inexpensive method of tuning the hyperparameters is needed.

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

In this paper, a reinforcement learning agent is used to control the mutation probability of the GA adaptively. The agent chooses the mutation probability for the GA for every five generations. By this approach, convergence issues of the previous augmented GA are addressed. The agent tries tuning the mutation probability, so the GA does not get stuck in the local minimum. The proposed method is evaluated for solution quality and time cost in comparison to the previous version and a conventional genetic algorithm, using various test cases with different target impedances.

II. Augmented GA v1

The flowchart of the Augmented GA proposed in [6], denoted as Augmented GA v1 in this paper, is shown in Fig. 1. Instead of using a random initial population as in conventional GA, the Augmented GA v1 uses an augmented initial population. This is done by finding the best proportion of decaps needed for a specific board before starting the optimization. By doing so, this approach aims to determine the solution which might yield an ideal solution prior to commencing optimization, consequently improving the efficiency of the GA search. Two frequency points are considered for an RL-type target impedance: the final frequency critical point and the transition frequency critical point. For an R-type target impedance, only the last frequency point is considered. Decap weights are generated for each decap in the decap library by adding one decap at a time. This is done to know the proportions of decaps needed to satisfy the impedance at the final frequency critical point for R type target and transition and the final frequency point for the RL-type target.

The decap solution is encoded as a vector of real numbers where the index corresponds to decap ports, and values in each index correspond to the decap type in the decap library. An encoded example solution is shown in Fig. 2. Here, the value 7 in index 3 means the 7th decap in the decap library is placed in the 3rd port. The value 0 in index 5 means no decap is placed in the 5th port. The main goal of the GA is to find a solution satisfying the target with minimal decap ports used. For solutions that satisfy the target impedance, the fitness is given by (1), and for solutions that do not satisfy the target impedance, the fitness is given by (2) [6].

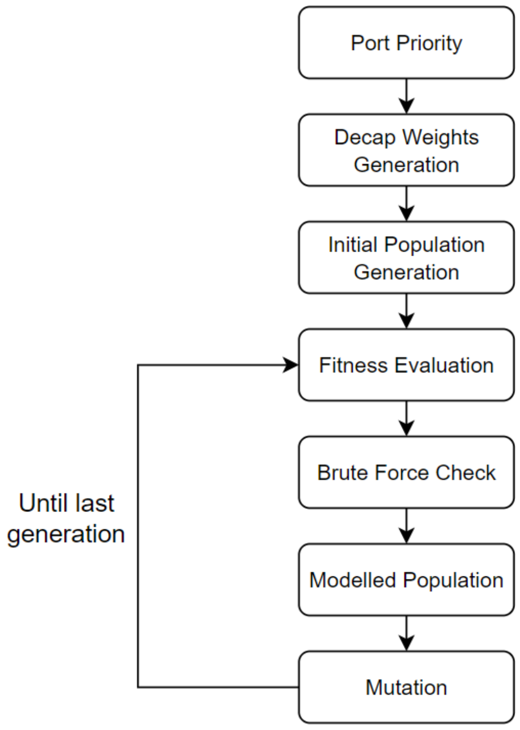


Fig. 1. Augmented GA v1 flowchart.

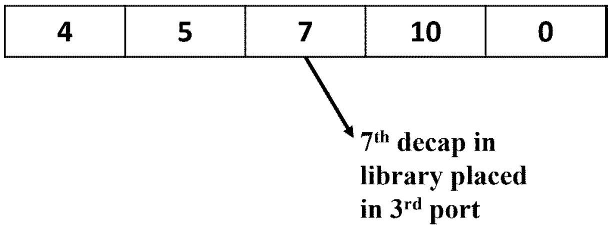


Fig. 2. Encoded decap solution example

$$\text{Fitness} = -(\text{Total \# of Ports} - \text{\# of Ports Used}) + 1 \quad (1)$$

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

This fitness function makes the GA find the solution satisfying the target impedance while simultaneously optimizing the number of decaps. Performing crossover operations on entire decap solutions would make more changes at once, making improvements difficult to achieve. Hence the crossover operator was removed.

The Augmented GA relies heavily on mutation operators to generate new solutions. New mutation operators were introduced, namely interchange mutation and shift mutation. In the interchange mutation, the decap type is interchanged with the other decap types in the solution. In the shift mutation, the decap ports are shifted either left or right by random steps. There is also the custom mutation, where the decap types are mutated to any other type in the decap library. An example of custom mutation is shown in Fig. 3. Here, the decaps in ports 2 and 4 are mutated.

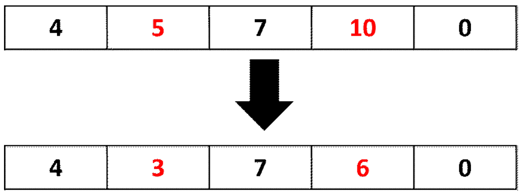


Fig. 3. Example of custom mutation (first row: before mutation, second row: after mutation)

A brute force check function was added, which reduces the number of decaps in the solution by iteratively removing the decaps from the solution. The procedure involves removing one decap at a time and checking if the target impedance is still met. The order in which the ports are removed is based on inductance calculated port priority. This inductance seen from each port is extracted from the Z -parameters. If the target impedance is still met after removing the decap for all ports, they will remain empty. However, if the impedance target is not met, the decap for that particular port will be retained, and the next port will be evaluated.

III. Augmented GA v2

A. *Tuning of Mutation Probability*

The typical problem with the GA is premature convergence, i.e., the algorithm gets stuck in local minima and cannot find the global minimum. By changing the mutation probability, the algorithm was able to find a better solution. Fig. 4 shows a convergence graph of the GA for 100, 75, and 50 decap port cases with mutation probability (*Pm*) of 0.4 and 0.6. For the 100 decap port example case, when the algorithm ran with 0.4 mutation probability, it gave a solution of 69 capacitors needed to satisfy the target impedance. Still, when it was run with 0.6 mutation probability, a better solution of 64 capacitors was obtained. When the algorithm was run with a mutation probability of 0.4 for the 75 decap port case, a better solution of 37 capacitors was obtained compared to the solution of 39 capacitors when it was run with a *Pm* of 0.4.

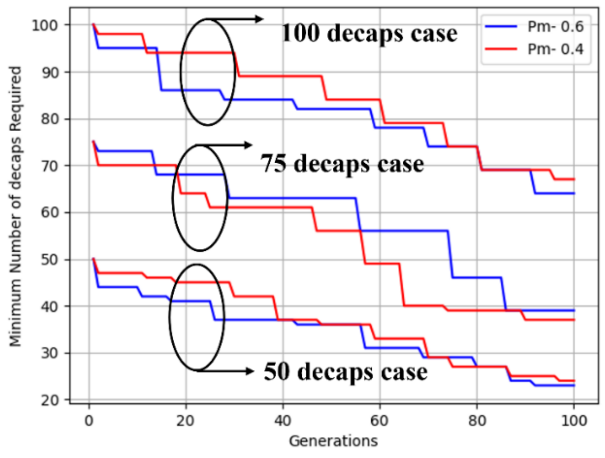


Fig. 4. Algorithm convergence graph for three cases (100, 75, and 50 decap ports).

It is observed from Fig. 4 that for each board and its specific target impedance, the mutation probability must be tuned separately to find the better solutions possible. Instead of randomly changing the mutation probability over generations, an intelligent way of tuning is required to achieve better performance. Hence in this algorithm, an RL agent is used to tune the mutation probability of the GA over generations adaptively.

B. Reinforcement Learning Overview

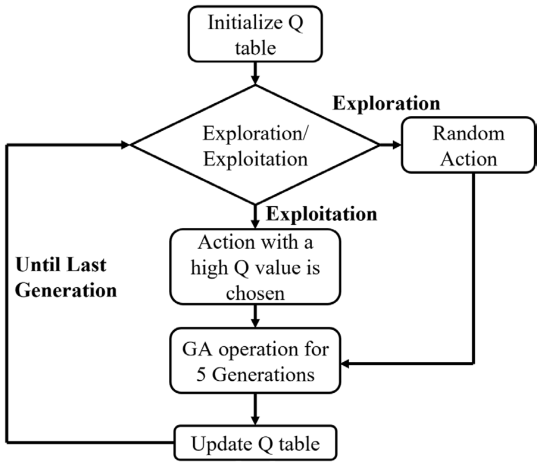
Reinforcement Learning is a feedback-based ML technique that enables the agent to learn in an interactive environment using feedback from its actions and experiences.

The proposed model is based on the Q-learning model [10], which iteratively improves the off-policy method. The RL parameters are defined as follows.

- **State (S):** The state is the current situation of the agent, in this case, it's either fitness improvement observed or no fitness improvement observed.
- **Action Space (A):** The action space contains the mutation probabilities ranging from 0 to 1 with intervals of 0.1.
- **Reward (R):** The reward R is determined based on the fitness returned by the GA for a specific mutation probability. If the fitness is improved, the reward is a positive value of 100, and if the fitness is not improved, the reward is a negative value of 100.

C. GA+RL

The flowchart for GA+RL is shown in Fig. 5. In the exploration stage, the agent explores the action space by choosing random mutation probability and running the GA operation for 5 generations. After 5 generations, the fitness of the GA is observed, the corresponding reward to assigned, and the Q table is updated. After the exploration, the stage is finished exploitation stage starts and here, the agent chooses a mutation probability with the maximum Q value in the Q table. In this way, the GA can find a better solution which is the best solution.



Initialize Q table
Exploration
Exploration/ Exploitation
Random Action
Exploitation
Action with a high Q value is chosen
Until Last Generation
GA operation for 5 Generations
Update Q table

Fig. 5. Flowchart of the GA+RL algorithm.

IV. Validation

Validation of the proposed algorithm was carried out by comparing it with the previous version and commercial tools in terms of both solution quality and time cost.

A. Comparison with Previous Works

The comparison of the proposed Augmented GA v2 was compared with the previous version v1 and the conventional GA [10]. There are 30 test cases (10 each for 75, 100, and 150 decap ports). These test cases were created using the boundary integral method [11-12]. There are ten decap types in the decap library ranging from 0.1 µF to 330 µF[5]. Fig. 6 and Fig. 7 compare the solution quality and time cost of the proposed algorithm with the previous version, respectively.

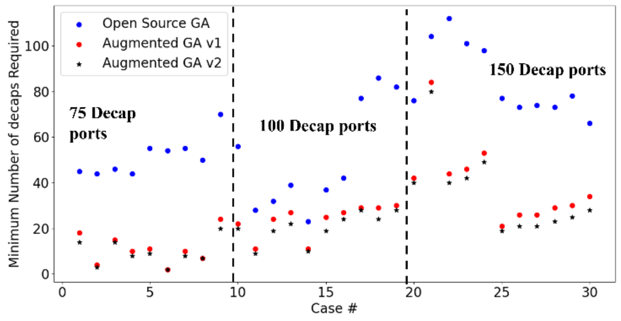


Fig. 6. Comparison of the algorithm- solution quality.

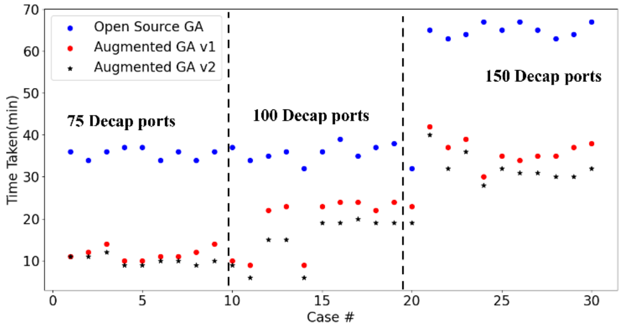


Fig. 7. Comparison of the algorithm- time cost.

From both Fig. 6 and Fig. 7, it can be seen that the proposed algorithm can achieve better solution quality and be less computationally expensive compared to the previous version and the conventional GA. For instance, in one case of 150 decap ports, the conventional GA took around 70 mins to find a solution of 60 decaps, while the Augmented GA v1 took around 40 mins to find a solution of 30 decaps. The v2 outperformed both by finding a solution of 25 decaps in 30 mins.

Fig. 8 compares the convergence graph for the proposed algorithm and the previous version. For the three boards (100, 75, and 50 decap ports), the algorithm, with the help of the RL agent, can converge to the global minimum faster. In the case of the 100 decap port case, the proposed algorithm can find the solution of 62 capacitors in the 61st generation, while the previous version found a solution of 64 capacitors in the 91st generation. It is also to be noted that the previous version may find the global minimum if it was run for more generations, but it will significantly affect the time cost. But the proposed algorithm can do that in less computation time.

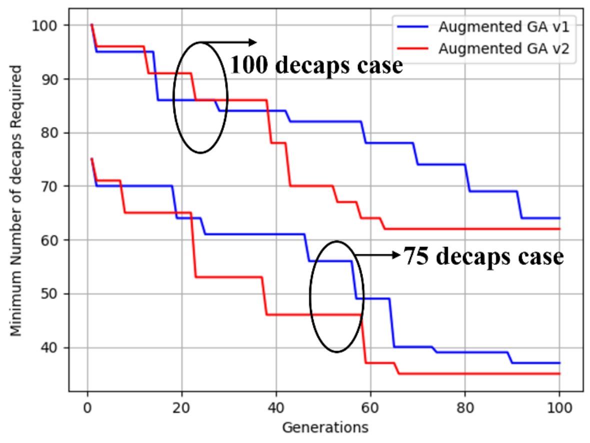


Fig. 8. Comparison of convergence of algorithm v1 and v2 for two cases (100 and 75 decap ports).

B. *Comparison with Commercial Tools*

The proposed algorithm is compared with the commercial tools available - Cadence Optimize PI and ANSYS SIWAVE PI advisor. A real board design with 23 decap ports was used for this comparison. The same decap library was used in these tools. Table I shows the comparison for the example. In terms of both solution quality and time cost, the proposed method is significantly effective.

TABLE I. COMPARISON WITH COMMERCIAL TOOLS

TOOL	Minimum # of decaps found	*Time taken*
Proposed method	3	72 seconds
Optimize PI	3	195 seconds
PI Advisor	4	662 seconds

V. CONCLUSION

In this work, with the help of a reinforcement learning agent, the mutation probability of the Augmented GA was tuned for every five generations. The agent helps the GA in finding the best solution available. The proposed method performs better than the previous version and as well as the commercial tools in terms of both solution quality and time cost.

REFERENCES

[1] S. Hemaram and J. N. Tripathi, "Computational Intelligence Based Selection and Placement of Decoupling Capacitors: A Comparative Study," *IEEE Electromagnetic Compatibility Mag.*, vol. 11, no. 2, pp. 49-59, 2nd Quarter 2022.

[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 Int. Symp. on Electromagnetic Compatibility & Signal/Power Integrity (EMCSI)*, 2020, pp. 245-250.

[3] 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.

[4] S. Han, O. W. Bhatti and M. Swaminathan, "Reinforcement Learning for the Optimization of Decoupling Capacitors in Power Delivery Networks," *2021 IEEE Int. Joint EMC/SI/PI and EMC Europe Symp*., 2021, pp. 544-548.

[5] J. Juang, L. Zhang, Z. Kiguradze, B. Pu, S. Jin and C. Hwang, "A Modified Genetic Algorithm for the Selection of Decoupling Capacitors in PDN Design," *2021 IEEE Int. Joint EMC/SI/PI and EMC Europe Symp.*, 2021, pp. 712-717.

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

[7] John J Grefenstette, “Optimization of control parameters for genetic algorithms,” *IEEE Trans. on systems, man, and cybernetics*, vol. 16, no. 1, pp. 122-128, Jan. 1986.

[8] Arif Arin, Ghaith Rabadi, and Resit Unal, “Comparative studies on design of experiments for tuning parameters in a genetic algorithm for a scheduling problem”, *Int. Journal of Experimental Design and Process Optimisation,* vol.2, pp. 102–124, May. 2011.

[9] D. Pandey and P. Pandey, "Approximate Q-Learning: An Introduction," *2010 Second Int. Conf. on Machine Learning and Computing*, 2010, pp. 317-320,

[10] R. Solgi, “Genetic Algorithm”, pypi.org, Available: https://pypi.org/project/geneticalgorithm/ (accessed Sept. 1, 2022).

[11] L. Zhang et al., “Efficient DC and AC Impedance Calculation for Arbitrary-Shape and Multilayer PDN Using Boundary Integration,” in *IEEE Trans. on Signal and Power Integrity*, vol. 1, pp. 1-11, 2022.

[12] L. Zhang, “PDN modeling for high-speed multilayer PCB boards and decap optimization using machine learning techniques,” Ph.D. dissertation, Missouri Univ. Sci. Technol., Rolla, MO, USA, 2021.