A Modified Genetic Algorithm for the Selection of Decoupling Capacitors in PDN Design

Jack Juang[#1], Ling Zhang[#2], Zurab Kiguradze[#3], Bo Pu[#4], Shuai Jin[*5], and Chulsoon Hwang[#6]
[#]EMC Laboratory, Missouri University of Science and Technology, Rolla, MO, USA[1]
jjryb, [2]lzd76, [3]kiguradzz, [4]bpdbh, [6]hwangc@mst.edu
*Google Inc., Mountain View, CA, USA

2021 IEEE International Joint EMC/SI/PI and EMC Europe Symposium | 978-1-6654-4888-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/EMC/SI/PI/EMCEurope52599.2021.9559292

***Abstract*—Decoupling capacitors are used to provide adequate and stable power for integrated circuits in printed circuit boards (PCB). For complicated and large designs, it is difficult to select capacitors to meet voltage ripple limits while also minimizing cost because the search space is too large. In this work, a new genetic algorithm (GA) is proposed for the selection and placement of capacitors to meet a target impedance using as few capacitors as possible. The GA is centered around controlling the number of unused port locations in the GA population solutions, with the result of smoothing out the GA convergence and speeding up the convergence rate. A result comparison is made of the proposed GA against other algorithms and found the GA competitive if not better for the select test cases.**

Keywords—genetic algorithm, decoupling capacitor, power distribution network, printed circuit board

I. Introduction

In power distribution networks (PDN) for printed circuit boards (PCBs), at higher frequencies, the inductances associated with the voltage regulator module (VRM) and current return paths becomes an increasing source of impedance. This presents significant power delivery issues on current switching events which greatly impacts the performance of integrated circuits (ICs). A common method to ensure reliable power delivery is defining a target impedance, which is based on the maximum allowable voltage ripple that can be tolerated by devices on the power rail for continued functionality. With increasingly small, dense, and fast designs, meeting the ripple voltage tolerances becomes more difficult.

To reduce the power issues associated with high frequencies, decoupling capacitors (decaps) are used to provide a local source of charge while also providing a lower inductance/impedance return path. The problem is, designs with large numbers of decap ports contain too many placement possibilities. Of all decap patterns, there exists an application-based 'best solution(s).' This may be the pattern that satisfies a target impedance using the minimum number of capacitors or the pattern that minimizes a bill of material cost. Very large search spaces make brute force methods impractical for finding the best solution.

For this decap placement problem, different search methods have been proposed and implemented. Among those is a physics-based method for minimizing inductance [1], iterative methods [2] and machine learning methods to quickly determine the best solution for any input [3][4]. Different genetic algorithms (GA) have also been implemented. [5][6]. Nearly all search or iterative methods though, rely on specific objective functions and/or made assumptions about how the best solution is most easily found. As an example, adding decaps based on the distance to an IC would always use the same decap ports. While these assumptions are based in physics and do make the search efficient by narrowing the search space, the tradeoff is that there is no way to verify that the reduced search space includes the best solution.

In this work, we propose a new GA to find the decap placement that minimizes the number of capacitors required to meet a target impedance. To accomplish this, a new search method of limiting the number of capacitors in GA solutions is introduced. Contrary to traditional GAs, we also use very general fitness functions to avoid directly narrowing the search space. For disambiguation, the proposed GA will also be referred to as the gene suppressed GA.

II. Proposed Genetic Algorithm Overview

First introduced by Holland, genetic algorithms are a class of optimization functions based on the principles of survival of the fittest [7]. Mimicking the process of natural selection, a GA population experiences the familiar pressures of survival fitness (selection), reproduction, and mutation. Iteratively, a GA population goes through cycles, called generations, where the most fit individuals will reproduce. The overall most fit individual, over all generations, is the best solution for the optimization problem. For the proposed algorithm, the code base for the GA is open-source, implemented in Python and freely available from [8]. Full documentation and the full code is available. The general structure for the proposed GA is described in Figure 1. The code for the genetic operators is unchanged.

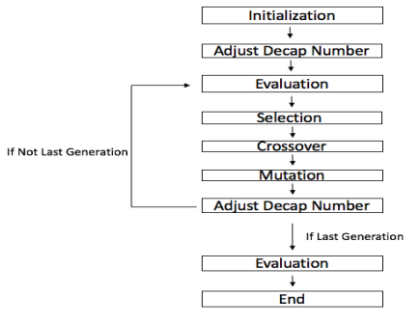

Fig. 1: Genetic Algorithm Flow Chart

A. Genetic Algorithm Structure

The first stage of the GA is initialization, where the initial population of the GA is generated. Each member of the GA is referred to as a chromosome made up of genes [7]. To fit the GA

This paper is based upon work supported by the National Science Foundation under Grant No. IIP-1916535 and a Google Faculty Research Award.

scheme, each decap placement pattern is encoded as a vector of real, non-negative integers. A decap port location is represented by the vector index, with the length of the vector equal to the total number of ports. The specific capacitor placed at a port location is represented by the value at the corresponding index. An example solution is shown in Fig. 2. The mapping of integer numbers to capacitor type is given in Table 1, with an integer value of '0' representing no decap. For initialization, the population was filled by randomly generated solutions.

2	1	10	5	3	4	0	10

Fig. 2: Example solution with 8 decaps. Decap number 5 is placed in port 4.

Table 1: Decoupling Capacitor Library

Type #	Decap Parameters		
	Capacitance (uF)	ESL (nH)	ESR (mΩ)
1	0.1	0.19	34.7
2	0.47	0.18	18.3
3	1	0.22	15.2
4	2.2	0.20	7.2
5	4.7	0.28	7.1
6	10	0.26	5.2
7	22	0.27	4.0
8	47	0.15	2.9
9	220	0.41	1.9
10	330	0.46	1.2

The next stage of the GA is evaluation, where the fitness of a solution is judged. In our case, the lower the fitness score given by a fitness function, the higher its fitness. Two different fitness functions are used for evaluating solutions; one for solutions satisfying the target impedance and one for those that don't. For a solution satisfying the target impedance, the fitness function used is given by (1):

$$Score = -(Total\ \#of\ Ports - \#Used\ Ports + 1) \quad (1)$$

If the target impedance is not satisfied, the fitness score given is proportional to the largest difference between points of the target impedance, the target_z, and the solution_z, the impedance seen looking into an IC on the power rail. For the test cases in this paper, only the PDN AC impedance associated with the vertical vias and plane capacitance is considered and calculated using a BEM and node voltage method [9]; the horizontal routing was not considered. The effect of the thickness of the power and ground layers is assumed negligible on the vertical AC impedance. The algorithm still applies with inputs that consider DC resistance and horizontal routing. The frequency range targeted is 10 kHz to 20 MHz. The fitness score is given by the following fitness function (2):

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

To create the next generation, first, selection occurs to choose the parents. A percentage of the current generation, set at 30%, is selected through the roulette wheel method [10] to join the next generation. Equivalently this means the crossover rate, the number of solutions created by crossover, is 70%. These solutions are the potential parents for new solutions. An elitism component [7] is included where a percentage of the highestt fitness solutions are guaranteed to join the next generation. The elitism percent is set at 1% with a minimum of 1 solution joining the next generation. The remainder of the population is generated through uniform crossover [11] of randomly selected pairs of the potential parents. Finally, mutation occurs where every gene has a chance of being changed. In our case, the decap placed at a particular port may have its decap type changed or be removed altogether. The mutation rate is set at 10%. After mutation, one generation is completed. The entire process repeats again with fitness evaluation, generation after generation, until a defined number of generations have passed.

B. Solution Size and Size Variation

A change is proposed here to the traditional GA scheme. A distinction is made between gene value 0 and genes 1 – 10; by our fitness function, more expressions of gene 0 lead to a better score. The frequency of gene 0 appearing will be controlled by the GA to make the search more efficient.

The solution size is defined as the number of decaps used in the current best-known solution. The proposed change is to limit the number of decaps in all solutions around the solution size. With a solution size of 20, there is no need to consider solutions using > 20 decaps so solutions should be restricted to <= 20 decaps. Solutions with 20 decaps are still considered as they may provide alternate search paths for the GA. This parameter is dynamically updated and initially set as the total number of decap locations.

While the upper limit is defined by the solution size, a lower limit is defined by the size variation. Without a lower limit on the number of decaps, the search space may be too large for efficient search. If the size solution is 20 and the size variation is 5, then all solutions in the population are allowed only 20 – 15 decaps inclusively. It is more probable to find solutions nearer to the current solution size number than one with far fewer decaps. The size variation parameter was set at a rounded 10% of the total number of decap ports.

Let S be the solution size and V the size variation. Fig. 3 describes the changes made to a solution with N decaps. For adding and removing decaps, the decap ports are randomly chosen from the solution. When adding, decaps are selected from the those already present in the solution. A solution that does not utilize decap number 8 will not have capacitor 8 as an option for adding. This is to avoid changing the overall behavior of the solution too much, such as by adding new resonances. Adjustments to the decap number occur after initialization and after mutation.

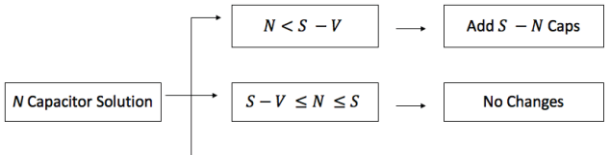

Fig. 3: Modifications to be made to a solution of N capacitors, with solution size S and size variation V.

III. GA VERIFICATION

To verify the performance of the proposed GA, 3 test boards were generated using code from [12]. The shapes and decap port layouts are given in Fig. 4. The stackups are given in Tables 2 – 4. The dielectric relative permittivity is 4.4. The thickness of PWR and GND layers is again assumed to have negligible effects on the total via inductances and capacitances compared to dielectric layer thickness. As such, PWR and GND layers are modeled as having 0mm thickness. Through-vias connect the appropriate layers and are used as decap ports.

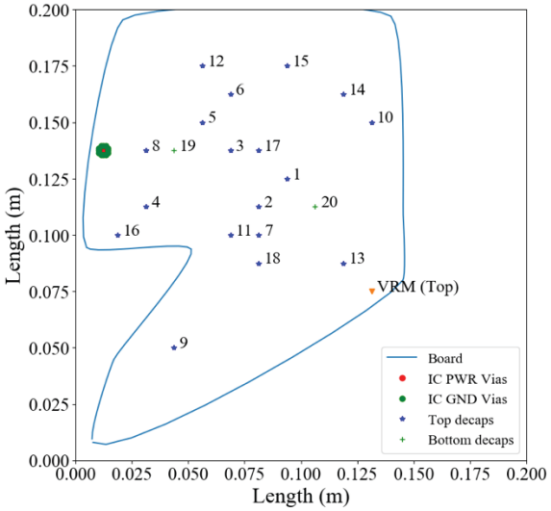

Fig. 4a: Board Shape and Decap Port Layout, Case 1

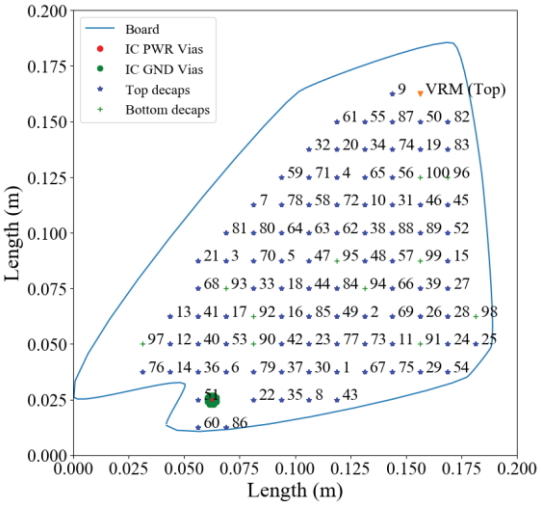

Fig. 4b: Board Shape and Decap Port Layout, Case 2

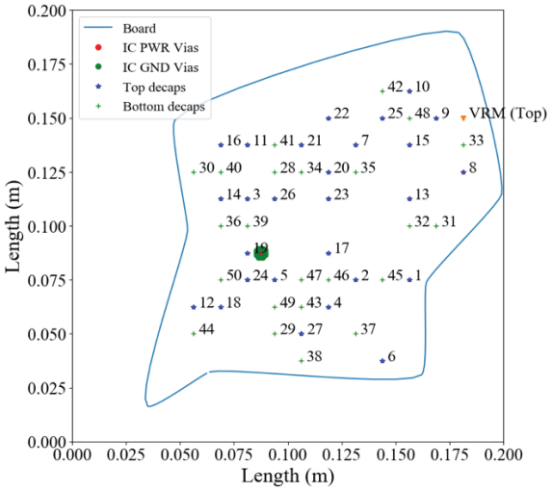

Fig. 4c: Board Shape and Decap Port Layout, Case 3

Table 2: Stack up for Case 1

Layer Type (Top Down)	Thickness (mm)
GND	0
Dielectric	0.3
PWR	0
Dielectric	1.7
GND	0
Dielectric	0.7
GND	0

Table 3: Stack up for Case 2

Layer Type (Top Down)	Thickness (mm)
GND	0
Dielectric	0.4
PWR	0
Dielectric	0.7
GND	0
Dielectric	1.2
GND	0

Table 4: Stack up for Case 3

Layer Type (Top Down)	Thickness (mm)
GND	0
Dielectric	0.2
GND	0
Dielectric	0.2
GND	0
Dielectric	0.3
PWR	0
Dielectric	0.3
GND	0
Dielectric	0.3
GND	0
Dielectric	0.3
GND	0
Dielectric	0.5
GND	0
Dielectric	0.2
GND	0

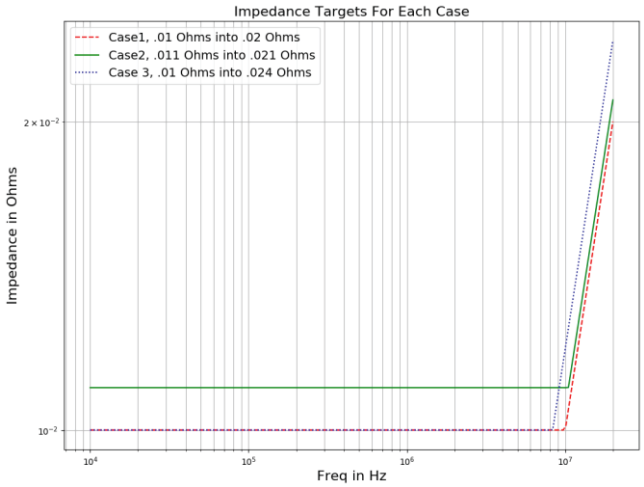

Fig. 5: Impedance Targets in Each Case. The Slope is +20dB/decade. The frequency range is 10 kHz to 20 MHz

Test board 1 (Case 1) has 20 capacitors ports. Test board 2 (Case 2) has 100 decap ports and test board 3 (Case 3) has 50. The decap port locations were randomly selected and placed on both top and bottom layers. The VRM was modelled as a series RL circuit with R = 3 mOhms and L = 2.5 nH and placed at the port farthest from the IC, but on the top layer. To set the target impedances for each case, the gene suppressed GA was run repeatedly, and the target impedance varied, until a reasonable number of capacitors that could satify the target impedance was found. The impedances are of RL type and are given in Fig. 5. Results of the gene suppressed GA are compared against the open-source GA without any modifications (same fitness function), against the method described in [6] in the ideal case, and against the reinforcement learning algorithm described in [3].

The unmodified version of the open-source GA is near identical to that of the gene suppressed GA, the only difference is that the number of capacitors allowed is not enforced. To check for search potential and consistency, both GAs' were ran 5 times, for each test case, with the population size and the number of generations = 50. They were rerun another 5 times, for each test case, with population size and the number of generations = 100.

The method proposed in [6] is an iterative GA, but rather than risk a poor recreation, a full search was performed. Decaps were selected one by one, by considering every decap type and location, and fixing the one that best minimizes the cost function. Decaps are placed until the target is met or until all ports are filled. The solution found then, is the best possible solution based on the cost function of [6].

The reinforcement learning method described in [3] involves training a machine learning model to find the minimal number of decaps needed to satisfy the target impedance. A port sequence is first calculated by [1] to determine the order of ports to be used. The good convergence of the model and its ability to generalize was not considered, only the best solution the that could be found. The algorithm was run three times, for each test case, and the best solution found was recorded.

A. Gene Suppressed and Open Source GA

The minimum number of decaps found by the GAs for each case is given in Table 5, along with the average time for each case. The GAs was run on a virtual Linux server with an Intel Xeon Gold 5118 processor. Better solutions are generally found with the gene suppressed GA, with large improvements as the number of decap ports increases. However, the time consumed increases considerably with the # of decap ports, population size, and generation number, making this method currently impractical for large industry design. It may take upwards of hours or more for large number of decap ports designs, depending also on GA parameters. One reason for improved results may be that genetic drift [13] is reduced with the introduction of the solution size and size variation parameters. Genetic drift describes the change in the frequency of genes in a population as the algorithm converges towards a local or global extrema. As better solutions are found, the frequency of gene 0 will increase in the population, especially with crossover considered. Solutions with too many empty ports are less likely to meet the target impedance and the number of such solutions appearing in the population will only increase with crossover. By controlling the number of decaps that can exist in a solution, the effect of genetic drift is lessened and the convergence can be improved. The convergence curve for the number of decaps found for the gene suppressed GA and open-source is shown in Fig 6 and for Case 2. Due to the elitism implementation, the plot is non-increasing.

Table 5: Open-Source vs Gene Suppressed GA Results

Genetic Algorithm	Number of Decaps		
	Case 1	*Case 2*	*Case 3*
Gene Suppressed GA Population and # Gen. = 50	**16 ~1 min**	**41 ~70 min**	**28 ~11 mins**
Open Source GA Population and # Gen. = 50	**16 ~1 min**	**71 ~75 mins**	**35 ~11 mins**
Gene Suppressed GA Population and # Gen. = 100	**15 ~5 mins**	**23 ~4.3 hrs**	**24 ~43 mins**
Open Source GA Population and # Gen. = 100	**15 ~5 min**	**66 ~5 hrs**	**32 ~45 mins**

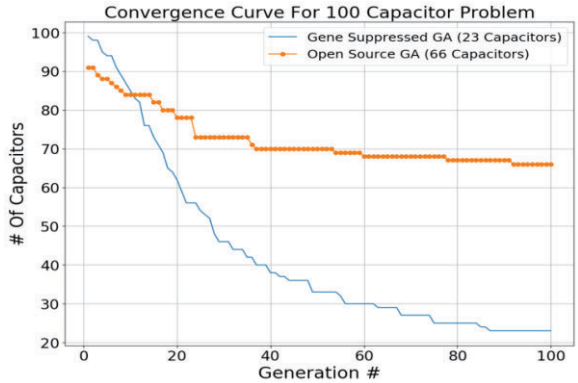

Fig. 6: Convergence of Gene Suppressed vs Open Source GA, Case 2

Table 6: Gene Suppressed vs Other Methods Results

Algorithm	Number of Decaps		
	Case 1	*Case 2*	*Case 3*
Gene Suppressed GA Population and # Gen. = 50	**16**	**41**	**28**
Gene Suppressed GA Population and # Gen. = 100	**15**	**23**	**24**
Ideal Solution of [6]	**No Solution**	**23**	**33**
Reinforcement Learning Method	**17**	**21**	**27**

B. *Gene Suppressed GA and Other Methods*

The minimum number of decaps found by the proposed GA, the ideal solution of [6], and the reinforcement learning method are given in Table 6. The performance of the proposed GA varies from finding a better or equivalent solution to at worst a competitive one. The inherent randomness in the search of any GA means that the best solution may not always be found despite its potential performance. The performance of the gene suppressed GA is as good if not better than the ideal solution of [6]. The method in [6] is a GA though, and the randomness of a GA search may still result in finding the global minimum solution. Compared to the reinforcement learning method, the proposed GA is competitive and sometimes better.

One possible reason for the sometimes-better results of the proposed algorithm is because no assumptions are made by the GA about how to find the best solution. In the method of [6], one decap is added at a time, as best minimizes the cost function. Per its cost function, bigger decaps should be added first because a bigger decap can quickly bring down the impedance in the low and medium frequency range. Progressively, smaller decaps will be added. In addition, the port locations are indirectly selected so that with a decap connected, the inductance associated with that location is just right to maximize the effect of that decap by shifting its resonance point. As a result, searches by [6] should be somewhat consistent in behavior and result.

In the reinforcement learning case, the order in which the ports are filled was fixed. The capacitor selection was not fixed. Exploration was done in the search space so different placement patterns could be tested. This exploration offers the benefit of considering alternate search paths but all search paths are restricted to the calculated port fill order. For Case 2, using the calculated port sequence resulted in the best solution. In Case 3, the solution found by the proposed GA is better, but no full search was done to check if an equal or better solution exists using the port sequence.

Without a full search, there is no way of confirming that any specific objective function, or any assumptions made about the global minimum solution, is always true. No direct assumption is made by the gene-suppressed GA about the best solution. The fitness function for solutions that satisfy the target impedance is proportional only to the number of unused ports. Solutions using fewer decaps will be favored by the GA, but not directly port locations or decap types. Characteristics of the same solutions however, may still be passed on repeatedly, pushing the GA towards a local minimum. Regardless, the possibility that better solutions can exist using decap ports and decaps that is unique from the currently known best solution is not closed off by the GA fitness functions. The downside though, is that there is no basis to judge two solutions using the same number of decaps, but using different locations and decap types, because they'd be rated the same fitness. From the results, limiting the expression of gene 0 help lead towards good solutions despite a less focused search space. But in some cases, like Case 2, it may not be sufficient.

The best decap placement pattern for Case 3, for each decap optimization method, is depicted in Fig. 7. Their impedance curves are given in Fig. 8. Although there is no guarantee that the GA in [6] and the reinforcement learning method is unable to find an equal or better solution for Case 3, the port locations used by the proposed GA are ones that would not be considered by the ideal case of [6] and the port sequence of the reinforcement method. For instance, capacitor port 11 is unused in the ideal solution of [6] and for the reinforcement learning case, port 11 is not within the first 27 ports of the calculated port sequence (Best solution found by reinforcement learning method).

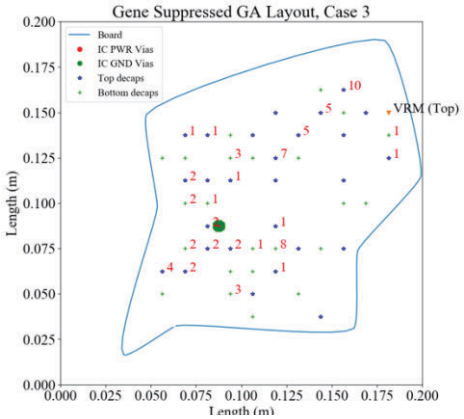

Fig. 7a: Gene Suppressed GA Capacitor Layout, Case 3
*Ports Used: [4, 5, 7, 8, 10, **11**, 12, 14, 16, 17, 18, 19, 20, 24, 25, 26, 28, 29, 33, 36, 39, 46, 47, 50]*

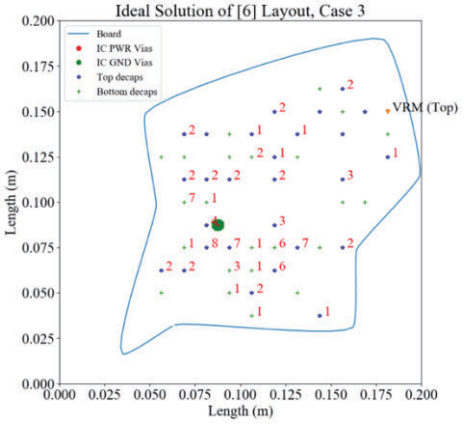

Fig. 7b: Ideal Solution of [6] Capacitor Layout, Case 3.
Ports Used: [1, 2, 3, 4, 5, 6, 7, 8, 10, 12, 13, 14, 16, 17, 18, 19, 20, 21, 22, 23, 24, 26, 27, 29, 34, 36, 38, 39, 43, 46, 47, 49, 50]

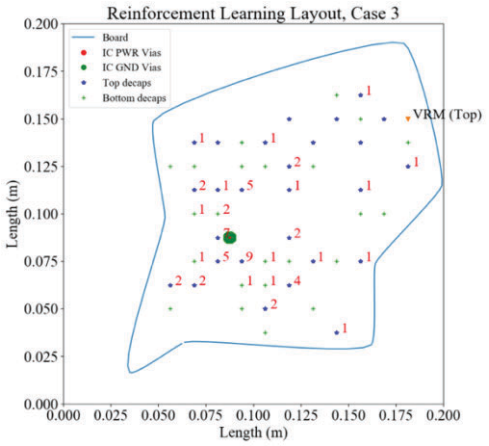

Fig. 7c: Reinforcement Learning Capacitor Layout, Case 3 Ports Used: [1, 2, 3, 4, 5, 6, 8, 10, 12, 13, 14, 16, 17, 18, 19, 20, 21, 23, 24, 26, 27, 36, 39, 43, 47, 49, 50]

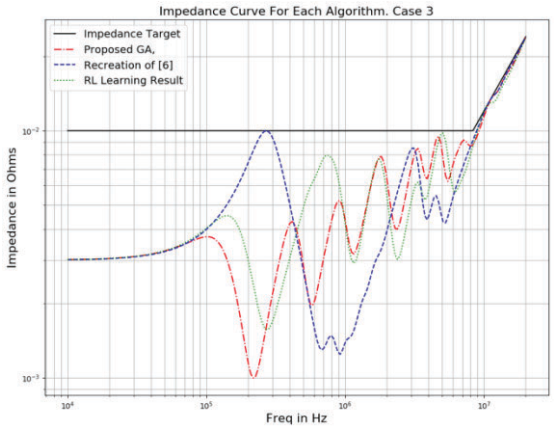

Fig. 8: Resulting Impedance Curves of Best Solution for Each Algorithm

IV. Conclusion

In this work, a new GA for the decoupling capacitor problem in power distribution networks of PCBs was presented. The gene suppressed GA can find the best known minimum capacitor number, or at least a comparable solution, to satisfy a user defined target impedance. This is verified by comparing against two other algorithms. The convergence curve can also be smoothed out and the search made more efficient by limiting the number of capacitors in solutions. From our experimental results, there is a relationship between the solution found and the GA parameters. Larger population sizes and longer iterations result in finding better solutions, especially for PDN with large numbers of capacitor ports, but at the expense of much longer algorithm time due to increase in the number of calculations.

References

[1] K. Koo, G. R. Luevano, T. Wang, S. Özbayat, T. Michalka and J. L. Drewniak, "Fast Algorithm for Minimizing the Number of decap in Power Distribution Networks," in *IEEE Transactions on Electromagnetic Compatibility*, vol. 60, no. 3, pp. 725-732, June 2018, doi: 10.1109/TEMC.2017.2746677.

[2] S. Han and M. Swaminathan, "A Non-Random Exploration based Method for the optimization of Capacitors in Power Delivery Networks," *2020 IEEE 29th Conference on Electrical Performance of Electronic Packaging and Systems (EPEPS)*, San Jose, CA, USA, 2020, pp. 1-3, doi: 10.1109/EPEPS48591.2020.9231448.

[3] L. Zhang, et al., "Decoupling Capacitor Selection Algorithm for PDN Based on Deep Reinforcement Learning," *2019 IEEE International Symposium on Electromagnetic Compatibility, Signal & Power Integrity (EMC+SIPI)*, New Orleans, LA, USA, 2019, pp. 616-620, doi: 10.1109/ISEMC.2019.8825249.

[4] H. Park *et al*., "Reinforcement Learning-Based Optimal on-Board Decoupling Capacitor Design Method," *2018 IEEE 27th Conference on Electrical Performance of Electronic Packaging and Systems (EPEPS)*, San Jose, CA, 2018, pp. 213-215, doi: 10.1109/EPEPS.2018.8534195.

[5] J. Y. Choi and M. Swaminathan, "Decoupling Capacitor Placement in Power Delivery Networks Using MFEM," in *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 1, no. 10, pp. 1651-1661, Oct. 2011, doi: 10.1109/TCPMT.2011.2165954.

[6] de Paulis, Cecchetti, Olivieri, Piersanti, Orlandi, and Buecker, "Efficient Iterative Process based on an Improved Genetic Algorithm for Decoupling Capacitor Placement at Board Level," *Electronics*, vol. 8, no. 11, p. 1219, Oct. 2019.

[7] K. F. Man, K. S. Tang and S. Kwong, "Genetic algorithms: concepts and applications [in engineering design]," in *IEEE Transactions on Industrial Electronics*, vol. 43, no. 5, pp. 519-534, Oct. 1996, doi: 10.1109/41.538609.

[8] R. Solgi, 2020. *Rmsolgi/Geneticalgorithm*. [online] GitHub. Available at: <https://github.com/rmsolgi/geneticalgorithm> [Accessed 15 January 2021].

[9] L. Zhang. J. Juang, Z. Kiguradze, et al. "Efficient DC and AC Impedance Calculation for Arbitrary-shape and Multi-layer PDN Using Boundary Integration," *IEEE Transactions on Electromagnetic Compatibility*; to be submitted.

[10] A. Shukla, H. M. Pandey and D. Mehrotra, "Comparative review of selection techniques in genetic algorithm," *2015 International Conference on Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE)*, Noida, 2015, pp. 515-519, doi: 10.1109/ABLAZE.2015.7154916

[11] A. Bala and A. K. Sharma, "A comparative study of modified crossover operators," *2015 Third International Conference on Image Information Processing (ICIIP)*, Waknaghat, 2015, pp. 281-284, doi: 10.1109/ICIIP.2015.7414781.

[12] L. Zhang, J. Juang, Z. Kiguradze, S. Jin, S. Wu, Z. Yang, J. Fan, and C. Hwang, "PCB-Level Decap Placement Using Deep Reinforcement Learning", *IEEE Transactions on Microwave Theory and Techniques*, to be submitted

[13] A. Rogers and A. Prugel-Bennett, "Genetic drift in genetic algorithm selection schemes," in *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 298-303, Nov. 1999, doi: 10.1109/4235.797972.