An Enhanced Deep Reinforcement Learning Algorithm for Decoupling Capacitor Selection in Power Distribution Network Design

Ling Zhang#1, Wenchang Huang#2, Jack Juang#3, Hank Lin*4, Bin-Chyi Tseng*5, and Chulsoon Hwang#6

#*Missouri S&T EMC Laboratory*
Missouri University of Science and Technology
4000 Enterprise Dr., Rolla, MO 65409
1lzd76, 6hwangc@mst.edu

**Advanced Electromagnetics & Wireless Communication R&D Center*
ASUSTek Computer Inc., Taipei
4Hank1_Lin, 5Bin-Chyi_Tseng@asus.com

***Abstract*—The selection of decoupling capacitors (decap) is a critical but tedious process in power distribution network (PDN) design. In this paper, an improved decap-selection algorithm based on deep reinforcement learning (DRL), which seeks the minimum number of decaps through a self-exploration training to satisfy a given target impedance, is presented. Compared with the previous relevant work: the calculation speed of PDN impedance is significantly increased by adopting an impedance matrix reduction method; also, the enhanced algorithm performs a better convergence by utilizing the techniques of double Q-learning and prioritized experience replay; furthermore, a well-designed reward is proposed to facilitate long-term convergence when more decaps are required. The proposed algorithm demonstrates the feasibility of achieving decent performance using DRL with pre-trained knowledge for more complicated engineering tasks in the future.**

Keywords*—machine learning, decoupling capacitor (decap), power distribution network, deep Q-learning, reward, prioritized experience replay, double Q-learning, target impedance.

I. Introduction

Placing decoupling capacitors (decap) is crucial to the power distribution network (PDN) design to lower down the impedance seen at IC ports and control the power supply fluctuation caused by large switching currents [1]-[5]. Typically, searching for the minimum number of decaps to satisfy a target impedance is desired in the industry to save cost and layout space. However, this process is usually tedious and time-consuming due to the tremendous search space induced by considerable decap library sizes and possible decap locations.

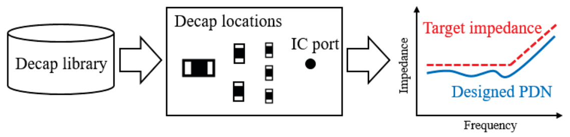


Fig. 1. Decap selection and placement in PDN.

Innumerable researches have emerged focusing on different approaches to decap optimization [6]-[8]. Some of them propose physics-based procedures to determine decap selections and locations [6], which are, however, too complicated to implement and duplicate in real applications. The genetic algorithm has also been applied for decap selection and placement [7][8], which seems an elegant solution to this problem. Nevertheless, the issue with the genetic algorithm is that it is nearly impossible to be reused, meaning every time it needs to start searching from zero.

In recent years, the magnificent achievement of deep learning and deep reinforcement learning (DRL) [9]-[11] is drawing the attention of scientists from different areas [12][13]. Since DRL has such an excellent performance to deal with various sophisticated tasks, it could be a nice solution for the decap placement problem. The biggest advantage of the DRL is that it can acquire knowledge and experience from training and exploration, which may make the trained neural network reusable for the same type of problem. Therefore, there have been some trials in this direction [14][15]. Reinforcement learning has been applied to predict both the decap locations and selection [14], but deep neural network (DNN) is not used in their model, which is thus difficult to handle more realistic and complicated scenarios. To overcome this limitation, DRL with DNN has also been implemented [15][16] to tackle larger search spaces with higher capacity. However, their DRL model cannot ensure optimum convergence, and the reward definition is too simple to address more practical situations with more required decaps.

In this paper, the objective is to further improve the DRL algorithm for decap selection mainly in three aspects to make it more practically useful: improve the impedance calculation speed, improve the convergence and stability, and improve the reward definition to enhance long-term convergence.

II. Basic Algorithm

To simplify the complexity, relieve some burden for the DRL algorithm, and ensure easier convergence, the whole challenge is divided into two parts: location prioritization and decap selection. The priority of decap locations is related to the inductance since better locations produce less inductance when connecting with decaps. After prioritizing the locations, then DRL can be applied to select the right decap to place on the locations sequentially.

A. Port Prioritization Method

A simple method to calculate the relative importance of decap ports is proposed in [6], by comparing the inductance values seen at the IC port. The concept is directly applied here to figure out the priority of potential decap locations in a given PDN board: the candidate decap ports are shorted to ground respectively, and the best port is picked out when observing the least self-inductance at the IC port; afterward, the determined best port is shorted, and the remaining ports are shorted again respectively to find the second-best port. Fig. 2 exemplifies this process using a simple case with only three decap locations. This process considers not only the self-inductances of the IC and decap ports but also the mutual inductances between these ports.

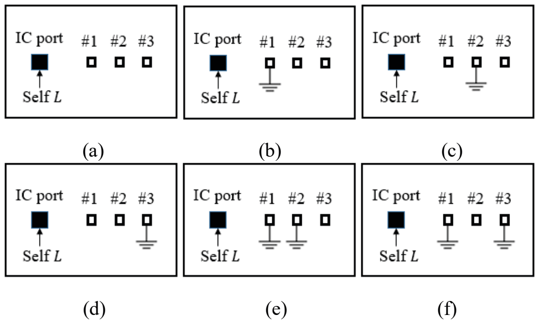


Fig. 2. Illustration of the port prioritization method based on inductance calculation. (a) Assume there are three decap locations to be evaluated: #1, #2, and #3. (b)(c)(d) Shorting port #1, #2, and #3 respectively, and compare the self-inductance seen at the IC port. (e)(f) If port #1 is considered as the best port with the least self-inductance at the IC port when being shorted, then port #2 and #3 are shorted again respectively to find the second-best port, and so on.

B. Decap Selection using DRL

Once the port prioritization is decided, only the optimal decap type needs to be determined to be placed on the prioritized locations in order, which can be accomplished by DRL according to [15]. A common algorithm of DRL–deep Q-learning–is utilized, which takes the observed state as input and outputs the Q-values for different possible actions [9] through a deep Q-network (DQN). The DQN can train and optimize itself through the reward of exploring different series of actions.

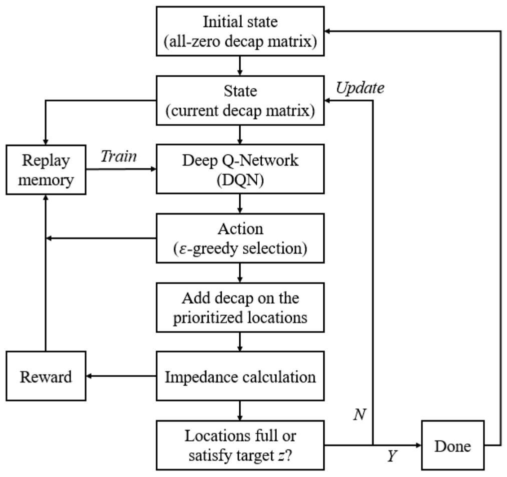

Initial state (all-zero decap matrix)
State (current decap matrix)
Update
Replay memory
Train
Deep Q-Network (DQN)
Action (ε-greedy selection)
Add decap on the prioritized locations
Reward
Impedance calculation
N
Locations full or satisfy target z?
Y
Done

Fig. 3. Flow chart of the decap selection algorithm based on DRL.

The algorithm process is described in Fig. 3, in which the decap matrix is the input state, and the decap selection is the output action. The dimension of the decap matrix corresponds to the number of decap locations, while the dimension of the decap selection is identical to the number of available decap types. The usage of the ε-greedy algorithm [9], which provides high probabilities of taking random selections in the beginning stage, is necessary to ensure enough exploration and reach the global optimum solution. Besides, a replay memory [9] stores an appropriate number of recent experiences that are sampled randomly to train the DQN, to avoid falling into a local minimum point but also ensure convergence.

III. Algorithm Improvement

A. Impedance Calculation Speed

The calculation speed of every decap addition is important to the speed of the whole process. To accelerate the impedance calculation, the Z-parameter of the PDN board and the decaps are obtained and used directly for impedance calculation by adopting a segmentation method introduced in [17].

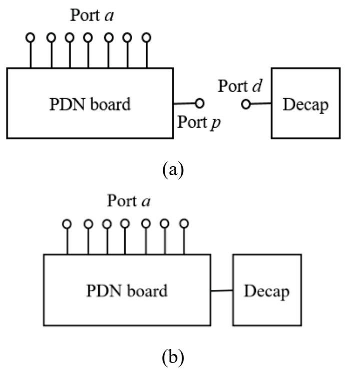


Fig. 4. The concept of using the segmentation method [17] to calculate the impedance matrix after connecting the PDN board with a decap. (a) Before connecting the decap. (b) After connecting the decap.

Fig. 4 depicts the decap addition problem, where Fig. 4(a) and (b) are the situations before and after connecting the decap, respectively. Port p is the port to be connected with a decap with port d, and port a (can be multiple ports) are the remaining ports of the PDN board. (1) expresses the relationship between the port voltages, port currents, and the Z-parameter matrix of the PDN board and the decap:

$$\begin{cases} \begin{bmatrix} U_a \\ U_p \end{bmatrix} = \begin{bmatrix} Z_{aa} & Z_{ap} \\ Z_{pa} & Z_{pp} \end{bmatrix} \cdot \begin{bmatrix} I_a \\ I_p \end{bmatrix} \\ U_d = Z_{dd} \cdot I_d \end{cases} \tag{1}$$

To connect port p with port d, the voltage and current of them must satisfy (2):

$$\begin{cases} U_p = U_d \\ I_p = -I_d \end{cases} \tag{2}$$

By organizing and simplifying (1) and (2), the relationship between U_a and I_a can be derived as (3):

$$U_a = \left[Z_{aa} - Z_{ap} \left(Z_{pp} + Z_{dd} \right)^{-1} Z_{pa} \right] \cdot I_a \tag{3}$$

(3) expresses the relationship between the port voltages, port currents, and the Z-parameter matrix of the new block shown in Fig. 4(b). Namely, the Z-parameter of the new block

with the connected decap is stated in (4), which is a simple matrix calculation that can be completed within fractions of a second.

$$Z_{aa}' = Z_{aa} - Z_{ap}\left(Z_{pp} + Z_{dd}\right)^{-1} Z_{pa} \tag{4}$$

For instance, for a PDN board with about 40 decap ports and 201 frequency points, converting Z-parameter to S-parameter takes around 0.4 seconds, which is the most dominant part; however, using Z-parameter directly, as shown in (4), only needs 0.005 seconds for each impedance calculation, which is a notable improvement on the speed of the whole decap selection process.

B. Double Q-Learning

As mentioned earlier, deep Q-learning is a commonly used DRL algorithm. The actor-critic algorithm is also a well-accepted method in deep Q-learning [9], the basic flow chart of which is shown in Fig. 5. There are two DQN with the same architecture but different parameters: the actor DQN and the critic DQN. The actor DQN is used to calculate the action by taking the input state, while the critic DQN is to help the training on the actor DQN.

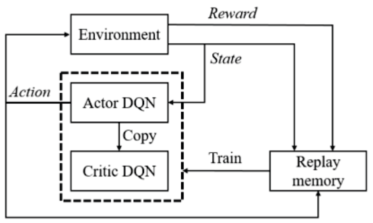


Fig. 5. The basic algorithm of actor-critic deep Q-learning.

The training process of Q-learning can be summarized using (5), where α is the learning rate, γ is the discount rate, R is the reward, s and a are the state and action, and s' and a' are the next state and the next action respectively. The discount rate, a value between 0 and 1, tells how important future rewards are to the current state.

$$Q_{(k+1)}(s,a) \leftarrow (1-\alpha)Q_k(s,a) + \alpha\left(R + \gamma \cdot \max_{a'} Q_k(s',a')\right) \tag{5}$$

In the actor-critic algorithm, the two DQN both are used for the training. The loss function is expressed in (6):

$$J\left(\theta_{actor}\right) = \frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)} - Q\left(s^{(i)}, a^{(i)}, \theta_{actor}\right)\right)^2 \tag{6}$$

$y^{(i)}$ is the target Q-value expressed in (7):

$$y^{(i)} = R^{(i)} + \gamma \cdot \max_{a'} Q\left(s'^{(i)}, a', \theta_{critic}\right) \tag{7}$$

where m is the size of the sampled batch; $s^{(i)}$, $a^{(i)}$, $R^{(i)}$, and $s'^{(i)}$ are the state, action, reward, and the next state of the i^{th} sampled memory; θ_{critic} and θ_{actor} are the critic and the actor's parameters; $Q\left(s'^{(i)}, a', \theta_{critic}\right)$ is the critic DQN's prediction of the Q-value expected from the next state $s'^{(i)}$ if action a' is chosen; $Q\left(s^{(i)}, a^{(i)}, \theta_{actor}\right)$ is the actor DQN's prediction of the Q-value for state $s^{(i)}$ and action $a^{(i)}$.

To train the parameters of the actor DQN, the typical gradient descent method can be used as shown in (8), where η is the learning rate for training.

$$\theta_{actor} \leftarrow \theta_{actor} - \eta \frac{\partial\left(J\left(\theta_{actor}\right)\right)}{\partial\left(\theta_{actor}\right)} \tag{8}$$

(7) can be understood as estimating the Q-value by taking the maximum value of the critic DQN's prediction of the Q-value expected from the next state. However, at the beginning of the training, the critic DQN is not trained and thus can output very abnormally large Q-values due to random initial parameters, which may cause severe instability in the training process [18].

To stabilize the training process, a double Q-learning method is proposed in [18]. Instead of directly using the maximum value of the critic DQN's estimation from the next state to obtain the target Q-value, the action causing the actor DQN's maximum Q-value from the next state is selected first, and then the selected action is used to obtain the critic DQN's Q-value from the next state. Through this indirect way of calculating the target Q-value, the Q-value overestimation issue can be greatly mitigated. The process of acquiring the target Q-value in this double Q-learning method is summarized in (9):

$$y^{(i)} = R^{(i)} + \gamma \cdot Q\left[s'^{(i)}, \arg\max_{a'} Q\left(s'^{(i)}, a', \theta_{actor}\right), \theta_{critic}\right] \tag{9}$$

Fig. 6 shows the comparison of the loss convergence between the double Q-learning and the original deep Q-learning algorithm when the discount rate γ is equal to 0.95. Apparently, double Q-learning shows a more stable and faster convergence.

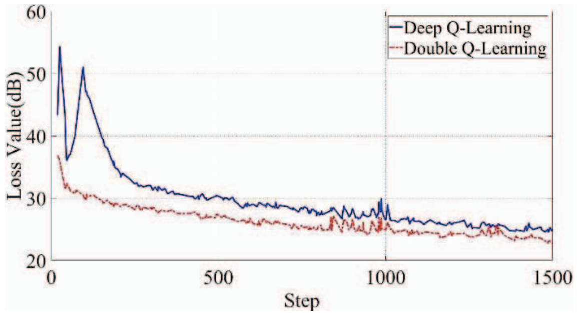


Fig. 6. The loss convergence comparison between deep Q-learning and double Q-learning.

C. Prioritized Experience Replay

In deep Q-learning, as introduced earlier, a replay memory is used to store a certain number of experiences. The size of the replay memory can not be too large or too small. The replay memory is sampled randomly in batches to train the DQN. The problem is that good experiences can be easily forgotten once they are thrown into the replay memory, due to the random sampling. To tackle this problem, a prioritized-experience-replay method is proposed [19]. In this method, better experiences have high probabilities of being sampled, which performs faster convergence than the traditional random sampling strategy.

In this paper, a similar idea with the prioritized experience replay is presented – a second separate memory, with a much smaller size than the replay memory, is constructed to store the good experiences, which are merged with the randomly-sampled experiences from the replay memory and used for training every time, as illustrated in Fig. 7. In this way, the good experiences will not be forgotten easily, and the convergence can be enhanced.

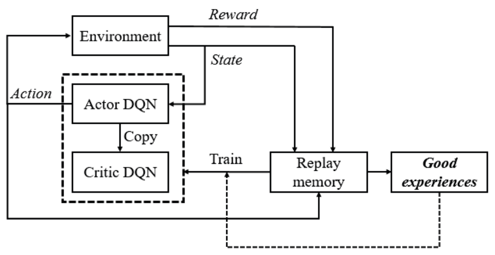
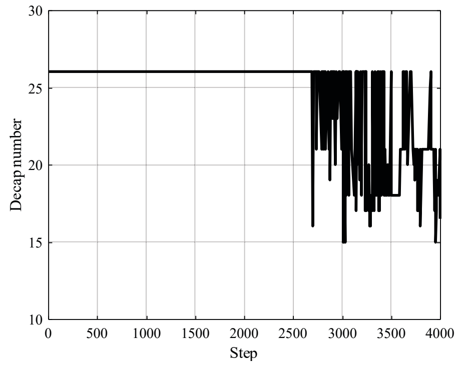
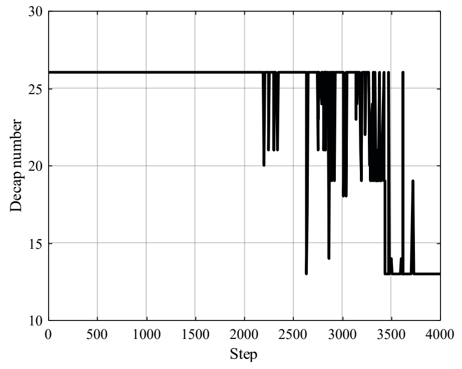


Fig. 7. Prioritized experience replay of deep Q-learning.

Fig. 8(a)(b) compares the convergence of decap number without and with using the prioritized experience replay when a PDN board with 26 decap ports is tested. Using the previous method of sampling training batches randomly from the replay memory, the algorithm shows a poorer convergence compared with the strategy of using a second memory to store the good experiences, which can steadily converge to the optimal solution of 13 decaps.



(a)



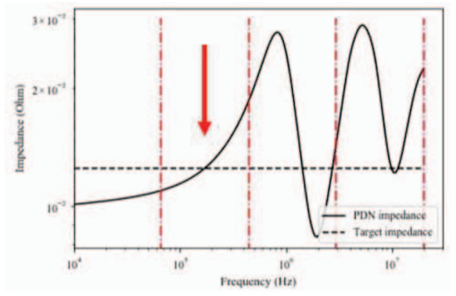
(b)

Fig. 8. (a) Change of decap number in one training without using the prioritized experience replay, which has a poor convergence. (b) Change of decap number in one training after using the prioritized experience replay, which can converge to a 13 decap solution.

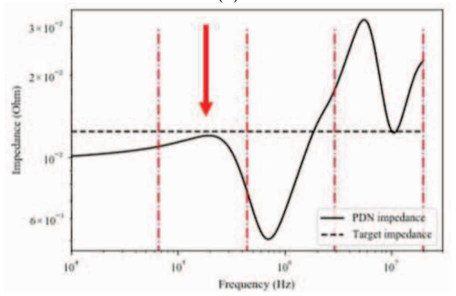
D. Reward definition

The reward definition is critical to the long-term performance of the DRL algorithm, which provides the direction to optimize the DQN parameters. In the previous work [15], the reward is given at the last step if the target impedance is satisfied. This reward definition has been shown effective in finding the best solution with 5 decaps from a decap library of 10 different decap types. Namely, the total number of possible cases is roughly 10^5, which is still manageable through many times of trial and error. However, when the number of decaps becomes large such as 10, this strategy is infeasible, because the number of possible cases has now become approximately 10^{10}. The total number of possible combinations grows exponentially with the number of decap locations and the decap library size. Due to the enormous search space in real scenarios, finding the optimal solution is nearly impossible through full search. Therefore, the reward function needs to be improved. In other words, some reward needs to be assigned in the intermediate steps to enhance the convergence under a long series of actions.

The first strategy in the reward definition is to divide the whole frequency range into multiple regions and assign a corresponding reward if a new region meets the target impedance. Fig. 9 demonstrates an example of dividing the whole frequency range into four regions and giving a reward corresponding to region 2 since region 2 is the new region from Fig. 9(a) to Fig. 9(b) to meet the target impedance. In this manner, the reward is easier to be obtained in the middle steps instead of the final step only, which solves the difficulty of handling a large search space using the previous reward definition. Also, this rewarding strategy mimics the way how engineers place different decaps to suppress corresponding frequency regions.



(a)



(b)

Fig. 9. The reward definition by dividing the whole frequency range into multiple regions and giving a reward once there is a new region satisfying the target impedance.

To ensure that under the circumstances where there is no solution to the target impedance, the algorithm can still converge to a solution as close to the target impedance as possible, an additional penalty is imposed to the model if all the decap locations are full and the target impedance is still not satisfied, as shown in Fig. 10, according to the area of the portion exceeding the target impedance. The penalty is namely a negative reward, which is another reinforcement to strengthen the learning process, considering that in real applications, a solution approaching the target impedance to the greatest extent is desired even if no solution exists.

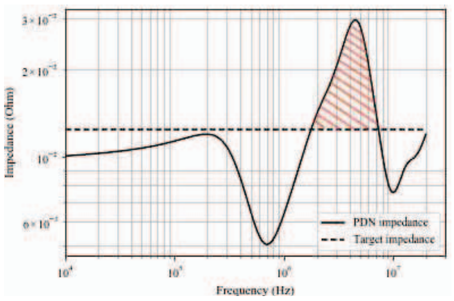


Fig. 10. A penalty, namely a negative reward, is imposed according to the area of the portion exceeding the target impedance marked as the shaded region if all locations are full and the target impedance is still violated.

Through the combination of the above two reward terms, the algorithm shows excellent performance, compared with the previous reward design. Fig. 11 compares the training score of the two different reward functions tested on a PDN board with 14 decap ports and a library of 10 decap types. Using the previous reward, the score is always zero even after a long time searching, since a solution with 14 decaps is nearly impossible to find through purely random search. However, by utilizing the new reward proposed in this paper, the algorithm can gradually get some score and eventually converge to the solution with the highest score, which is a substantial breakthrough.

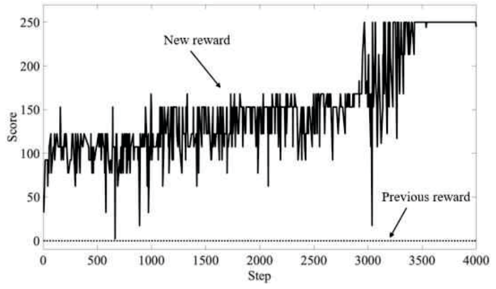


Fig. 11. Comparison of training score using two reward definition: the previous reward defined in [15] and the new reward proposed in this paper.

IV. Verification of the Algorithm

A. Test Example

To test the performance of the enhanced DRL algorithm, the S-parameter of a PDN board with 1 IC port and 14 decap ports is input to the algorithm. There are 10 different decap types in the decap library, and their capacitance values are 0.1uF, 0.47uF, 1uF, 2.2uF, 4.7uF, 10uF, 22uF, 47uF, 220uF, and 330uF, respectively. Each of these capacitors has different equivalent series resistance (ESR) and equivalent series inductance (ESL) values. The target impedance is 0.0125Ω, and the frequency of interest is from 10kHz to 20MHz. The neural network structure in the actor DQN and the critic DQN is the same with the structure used in [15].

B. Verification Result

After applying the new techniques discussed above, the convergence of the loss, the score, and the decap number during the training are shown in Fig. 12. The loss converges steadily and the decap number also successfully converges to a 13 decap solution. The PDN impedance of the solution is plotted in Fig. 13. The entire training process only takes about 10 minutes. Moreover, no better solutions could be found through other methods such as commercial tools and the genetic algorithm. Using the DRL algorithm proposed in [15], even no solution can be found at all. This strongly corroborates the effectiveness of the improved DRL algorithm in handling more realistic scenarios.

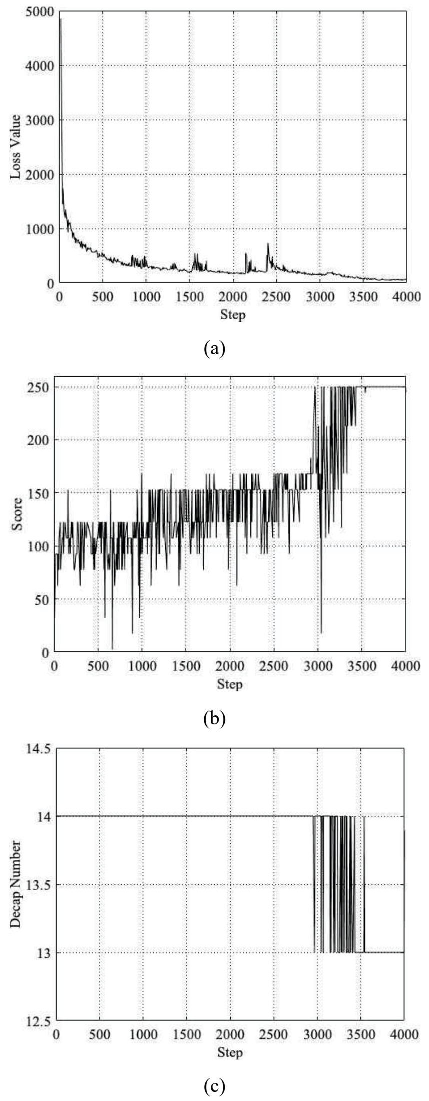


Fig. 12. (a) Change of the loss during training. (b) Change of the score during training. (c) Change of the decap number during training.

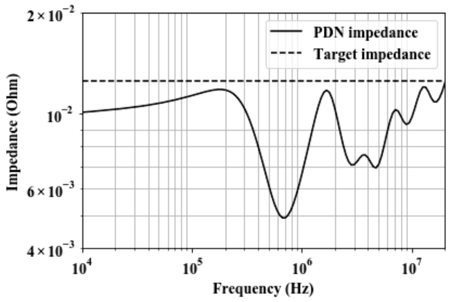


Fig. 13. PDN impedance of the 13-decap solution found by DRL with a target impedance of 0.0125Ω.

V. Conclusion

In this paper, a more sophisticated deep reinforcement learning (DRL) algorithm for decoupling capacitor (decap) selection in power distribution network (PDN) is developed to deal with more practical situations with more required decaps. The DRL algorithm can learn to select the optimum decaps to satisfy a given target impedance with as few decaps as possible, through the reward from an exploration and training process. Compared with the previous DRL algorithm for decap selection, the enhanced algorithm mainly improves three aspects: an impedance matrix reduction method is utilized to significantly increase the impedance calculation speed; the double Q-learning and prioritized-experience-replay techniques enhance and stabilize the convergence; furthermore, a new reward function is well designed to empower the algorithm to find the optimum solution even when tens of decaps are required. The algorithm is validated on a PDN board with 14 decap ports and successfully converges to a 13 decap solution in just about 10 minutes, which can not be achieved using the previous DRL algorithm.

The decent performance of the improved algorithm demonstrates the great capability of DRL, but it is limited to one single PDN board. In future work, more PDN boards will be used to train one neural network that can place decaps for a new board without being trained again. Furthermore, DRL can be considered to achieve professional levels in more complicated engineering tasks.

Acknowledgment

This paper is based upon work supported partially by the National Science Foundation under Grant No. IIP-1916535.

References

[1] J. Kim et al., "Improved target impedance and IC transient current measurement for power distribution network design," 2010 *IEEE International Symposium on Electromagnetic Compatibility*, Fort Lauderdale, FL, 2010, pp. 445-450.

[2] C. Huang et al., "Power integrity with voltage ripple spectrum decomposition for physics-based design," *2016 IEEE International Symposium on Electromagnetic Compatibility (EMC)*, Ottawa, ON, 2016, pp. 318-323.

[3] Y. Kim et al., "An efficient path-based equivalent circuit model for design, synthesis, and optimization of power distribution networks in multilayer printed circuit boards, " in *IEEE Transactions on Advanced Packaging*, vol. 27, no. 1, pp. 97-106, Feb. 2004.

[4] J. Kim et al., "Closed-Form Expressions for the Maximum Transient Noise Voltage Caused by an IC Switching Current on a Power Distribution Network," *IEEE Trans. Electromagn. Compat.,* vol. 54, no. 5, pp. 1112-1124, Oct. 2012.

[5] X. Zhu et al., "I-V Method Based PDN Impedance Measurement Technique and Associated Probe Design," *2019 IEEE International Symposium on Electromagnetic Compatibility, Signal & Power Integrity (EMC+SIPI)*, New Orleans, LA, USA, 2019, pp. 30-33.

[6] 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," *IEEE Trans. Electromagn. Compat.*, vol. 60, no. 3, pp. 725-732, June 2018.

[7] 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.

[8] S. Piersanti, F. de Paulis, C. Olivieri, and A. Orlandi, "Decoupling Capacitors Placement for a Multichip PDN by a Nature-Inspired Algorithm," *IEEE Trans. Electromagn. Compat.*, vol. 60, no. 6, pp. 1678-1685, Dec. 2018.

[9] V. Mnih, K. Koray, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. "Playing atari with deep reinforcement learning," *arXiv preprint arXiv*:1312.5602, 2013.

[10] D. Silver et al., "Mastering the game of Go without human knowledge," *Nature*, vol. 550, p. 354--, 2017.

[11] A. Krizhevsky et al., "Imagenet classification with deep convolutional neural networks." *In Advances in neural information processing systems, pp.* 1097-1105, 2012.

[12] J. Xu, L. Zhang, M. Sapozhnikov and J. Fan, "Application of Deep Learning for High-speed Differential Via TDR Impedance Fast Prediction," *2018 IEEE Symposium on Electromagnetic Compatibility, Signal Integrity and Power Integrity (EMC, SI & PI)*, Long Beach, CA, 2018, pp. 645-649.

[13] Z. Zhang, L. Zhang, Z. Sun, N. Erickson, R. From and J. Fan, "Solving Poisson's Equation using Deep Learning in Particle Simulation of PN Junction," 2019 *Joint International Symposium on Electromagnetic Compatibility, Sapporo and Asia-Pacific International Symposium on Electromagnetic Compatibility (EMC Sapporo/APEMC)*, Sapporo, Japan, 2019, pp. 305-308.

[14] 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.

[15] 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.

[16] H. Park et al., "Deep Reinforcement Learning-Based Optimal Decoupling Capacitor Design Method for Silicon Interposer-Based 2.5-D/3-D ICs," in *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 10, no. 3, pp. 467-478, March 2020.

[17] J. Kim et al., "Chip-Package Hierarchical Power Distribution Network Modeling and Analysis Based on a Segmentation Method," in *IEEE Transactions on Advanced Packaging*, vol. 33, no. 3, pp. 647-659, Aug. 2010.

[18] Van Hasselt, Hado, Arthur Guez, and David Silver. "Deep reinforcement learning with double q-learning." In *Thirtieth AAAI conference on artificial intelligence*, 2016.

[19] T. Schaul, et al. "Prioritized experience replay." *arXiv preprint arXiv:1511.05952*, 2015.