Proceedings of the 2015
IEEE Conference on Robotics and Biomimetics
Zhuhai, China, December 6-9, 2015

Reinforcement Learning Approach to Learning Human Experience in Tuning Cavity Filters*

Zhiyang Wang[1], Jingfeng Yang[2], Jianbing Hu[2], Wei Feng[3] and Yongsheng Ou[2,3,4]

Abstract— **Owing to the rapid development of the communication industry, various kinds of radio frequency components are in great demand and put into mass production. Among them, passive devices such as microwave cavity filters, duplexers and combiners have experienced fast and unexpected upgrades. However, the tuning process of these products, which is always manually operated, still seems hard to be automatically replaced or improved because of the difficulties in extracting human experience. In this study, we make deep investigations into some previous automatic cavity filter tuning solutions, especially the ones using intelligent algorithms. In addition, we propose the method of intelligent tuning based on the reinforcement learning algorithm which dynamically extracts the human strategies during the tuning process. The experimental results prove the powerful performance of reinforcement learning in mastering human skills.**

I. Introduction

Microwave cavity filters are in great demand in satellite communication industry, mobile communication industry, radar systems and electronic countermeasures (ECM) systems. Especially the recent fast growing rate of mobile communication industry makes it more pressing in solving the problem restricting the development of microwave cavity filters. Because of the imprecisions in manufacturing and assembling processes, each cavity filter should be carefully tuned by a professional technician before leaving the factory. This operation is completed by manually tuning the screws and adjusting the screw nuts to fit the scattering characteristics (or S-parameters) shown in the vector network analyzer (VNA) to the design specifications (Fig. 1).

The complicated and nonlinear relationship between the scattering characteristics and the screw positions of the cavity filters makes the tuning process difficult and time-consuming. The tuning speed not only depends on the specific type and structure of the filters, but also stresses the importance of human roles in the tuning operation. The tuning time varies with different human experience and strategies from several minutes to several hours. This makes the improvement of the production efficiency severely be restricted by lacking of experienced technicians.

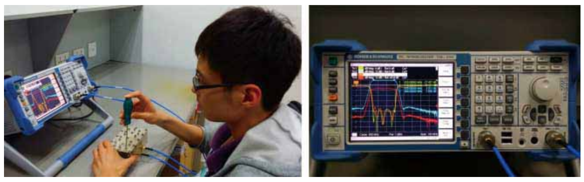
Fig. 1. Manually tuning of cavity filters by human experts. Screws are adjusted according to the scattering characteristics shown in the VNA. Left: The tuning operation. Right: S-parameters shown in the VNA.

In the year of 2014, the market for industrial robots in China was stimulated all of a sudden. Hundreds and thousands of small and medium-sized robotics enterprises have sprung up overnight. Some traditional industries are easy to be eliminated if refusing to face the revolution. Under the peculiar background, the automation reformation of the cavity filter tuning tasks seems particularly important and urgent. On the one hand, the manually tuning of filters consumes much time, slowing down the production efficiency of the enterprises. On the other hand, fewer and fewer young people tend to do those dull and repetitive tasks, which makes developing countries to be running short of cheap labour. Based on these, the research on intelligent tuning is raised.

The first thing to do is to explore an intelligent tuning algorithm which can provide appropriate tuning strategies, indicating which screw to adjust and to what extent will it achieve the optimal scattering characteristics. Second, we need to develop an advanced robotic system to reduce human labor as well as to increase tuning efficiencies. In this work only the first aspect is mainly considered and discussed since it is the key issue of the whole problem.

Some related studies on computer-aided tuning (CAT) techniques have been carried out for several years, but few developed tuning systems have been put into use. The main reason is that the generalization ability of those algorithms and systems is still weak, which could not follow the rapid production updates and various product kinds. Early in the 1980s to 1990s, researches on the CAT techniques rose up [1], [2]. The original version of the CAT provided a reference for tuning operators, which weakened the requirements for human experience. After several years, the automatic tuning system called RoboCAT was developed by the Canadian

*This work was supported by the National High-Tech Research and Development Program of China (863 Program, Grant No. 2015AA042303), the National Natural Science Foundation of China (Grant No. 61273335), the Guangdong Innovative Research Team Program (201001D0104648280), the Shenzhen Overseas Innovation and Entrepreneurship Research Program (KQCX2015033117354155) and the Shenzhen Fundamental Research Program (JCYJ20140718102705295).

[1]Zhiyang Wang is with Guangdong Provincial Key Laboratory of Robotics and Intelligent System, Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, 518055 Shenzhen, P.R.China

[2]Jingfeng Yang, Jianbing Hu and Yongsheng Ou are with Shenzhen College of Advanced Technology, University of Chinese Academy of Sciences, 518055 Shenzhen, P.R.China

[3]Wei Feng and Yongsheng Ou are with Key Laboratory of Human-Machine Intelligence-Synergy Systems, Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, 518055 Shenzhen, P.R.China

[4]Yongsheng Ou is also with The Chinese University of Hong Kong, Hong Kong. He is the corresponding author. `ys.ou@siat.ac.cn`

company COM DEV [3]. Similar methods during the same time period also include the one based on circuit model parameter extraction [4], the one based on poles and zeros of the input reflection coefficient and the one using time domain [5]. Those methodologies were based on coupling matrix or the physical structure of the specific filter, which more or less increased the reliance on the particular filter kind.

A more black-box way is to apply data-driven modeling techniques. A. R. Mirzai et al. proposed the first intelligent tuning algorithm based on machine learning in 1989 [6]. For more than 20 years afterwards, researchers have proposed different tuning algorithms based on adaptive network models [7], fuzzy logic controllers [8], [9], Artificial Neural Network (ANN) [10] and Support Vector Regression (SVR) [11] etc. J. J. Michalski originally used ANN to model the filter [10]. The input data is the S-parameters obtained from the VNA and the output data is the deviation from the current screw positions to the standard ones. The data sets are collected by randomly detuning a tuned filter, recording different pairs of the S-parameters and the corresponding screw deviations. The most striking feature of their method is that the model is purely data-driven, meaning that the model avoids considering the physical model of the filter, but only considers the screw deviations. Some similar works were tried to improve the modeling performance [12]. The readers could refer to [12], [13] for a more comprehensive review. Fig. 2 draws a sketch of these ideas.

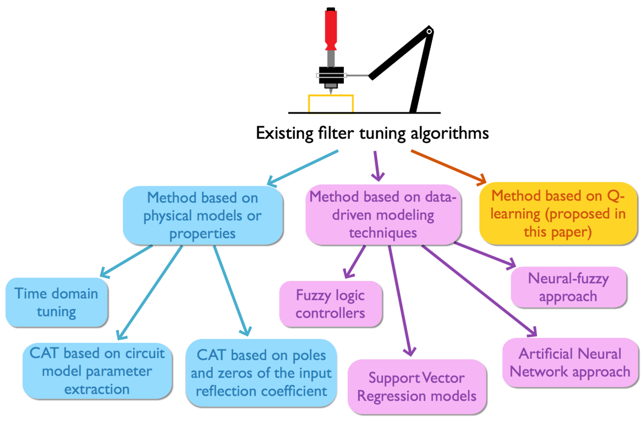
Existing filter tuning algorithms
Method based on physical models or properties
Method based on data-driven modeling techniques
Method based on Q-learning (proposed in this paper)
Time domain tuning
CAT based on circuit model parameter extraction
CAT based on poles and zeros of the input reflection coefficient
Fuzzy logic controllers
Support Vector Regression models
Neural-fuzzy approach
Artificial Neural Network approach

Fig. 2. Existing tuning techniques overview.

Even though the data-driven modeling techniques are proved to be efficient in solving the tuning problem, they inevitably pay great attention to the specific filter product, which weaken their generalization abilities. In other words, most of these methods are modeling the cavity filter, not the human experience. In the tuning problem, the role of human strategy is undoubtedly important. Experienced operators may complete a tuning task in several minutes while a beginner may take several hours or even longer. The learning process of a human expert in tuning the filter reminds us of the idea of Reinforcement Learning (RL), which originates from the learning process of animals. When an animal obtains rewards from some kind of action, dopaminergic neurons within the animal are stimulated and the trend to retake that kind of action is therefore reinforced. This biological process is extracted to build the important theoretical basis of RL. One of the best-known approaches to realizing RL by neural networks is called Temporal Difference (TD) learning. TD learning was first proposed by R. Sutton and A. Barto in 1981 [14], and was developed into Q-Learning by C. Watkins in 1989 [15]. In these algorithms, the state of an agent is represented as an input to a learning system (always be realized by a neural network), and the system outputs the predicted future returns (or Q-values) at that state. The agent then selects the next action to take according to the predicted future rewards. The weights and biases of the network are iteratively trained and updated to increase the prediction accuracy of the system.

In 2013, a technology company located in London proposed their findings in training a Q-network with Convolutional Neural Network (CNN) to make a computer learn to play simple Atari games [16]. The name of the company, DeepMind, was thus be remembered by the public. In 2015, they proposed their work in Nature after some small modifications [17]. In their method, all the Atari games were implemented in the Arcade Learning Environment (ALE). A Q-network was built and trained with CNN, which had a deep architecture and performed well in directly processing images, generating the name of Deep Q-Learning. The state was formed by the pixel images drawn from the game screens and input to the Q-network. The output of the network was the action-value function represented by a set of Q-values each of which corresponded to a valid action. At each learning epoch, an appropriate action was selected in terms of the predicted Q-values and executed by the ALE. Then the game score and the next screens could be automatically obtained and saved into a memory. A minibatch of samples was then randomly drawn from the memory to train the Q-network. In their work, the sequences of $\{state,\ action,\ reward,\ new\ state\}$ are saved in a memory as previous experience. To train the Q-network, examples are randomly drawn from the memory to break the correlations between consecutive samples. This technique helps avoid local minimum and divergence [17].

The work of DeepMind which employed Q-learning to train computers to play Atari games highly motivated our work. We consider similar techniques but apply them in our tuning problem. At present, we do not use deep architectures but instead apply some dimensionality reduction techniques to extract features. The remainder of this paper is organized as follows. In Section II we describe the problem formulation of the tuning task as well as the tuning target in detail. Afterwards, we propose the main idea of the intelligent tuning based on Q-learning in Section III. The experiments are illustrated in Section IV. Finally, Section V concludes the paper and provides directions for further research.

II. Problem Formulation

A. *The Tuning Process*

The flow chart in Fig. 3 depicts a simple workflow of an experienced operator. The whole tuning process is straightforward. First the tuning operator connects the device under test (DUT) with the VNA by cables. Then the VNA displays the S-parameters in the form of several wave curves. After that, the operator observes these curves and comes up with some pre-trained tuning strategies. In accordance with these strategies, the operator adjusts the screws and screw-nuts installed on the DUT to change the S-parameters. Then the expert observes the new wave curves and determines the following tuning actions. That circle repeats until the design specifications are satisfied.

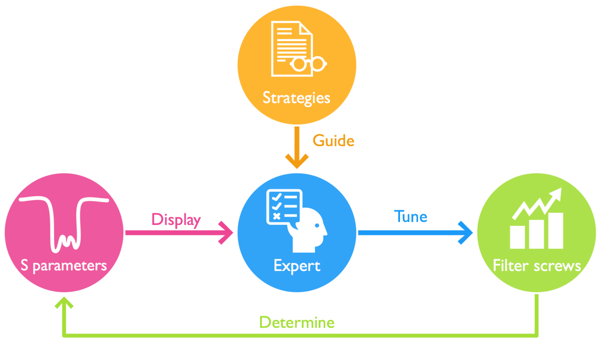

Fig. 3. Workflow of the tuning process. The operator is guided by his/her experience along with the S-parameters measured by the VNA.

B. Design Specifications for a Filter Product

The design specifications (i.e. our tuning objectives) are usually monitored from the scattering characteristics, or so called S-parameters, which are shown as different wave curves in the VNA. Generally, the design specifications for a filter product (also including duplexer and combiner) include Return Loss, Insertion Loss, Pass-band Fluctuation and Out-of-band Rejection. For a two-port single-channel filter, the S-parameters are shown in the form of four waveforms, S_{11}, S_{12}, S_{21} and S_{22} respectively. The representation of S_{ij} basically means the energy measured at port i injecting to port j. It is apparent that S_{11} and S_{22} are used to represent the Return Loss, denoting that the energy reflected to the source port. S_{12} and S_{21} denote the Insertion Loss, which means the energy transferred to the destination port. The S-parameters have both magnitude part and phase part. Fig. 4 provides an example of the magnitude of S-parameters in decibels (dB). This figure is also the most common figure for tuning operators to watch. The design specifications require the absolute values of the return loss within the passband should be as large as possible (i.e. under the red dashed line in Fig. 4). Furthermore, the absolute values of the insertion loss within the passband should be as small as possible (always close to zero). There are also requirements for the pass-band fluctuation and the out-of-band rejection which can be measured from S_{11} (or S_{22}) and S_{12} (or S_{21}) curves. In our study, we simplify the tuning objective by only considering the return loss requirement. In the future work, we might take additional design requirements into consideration.

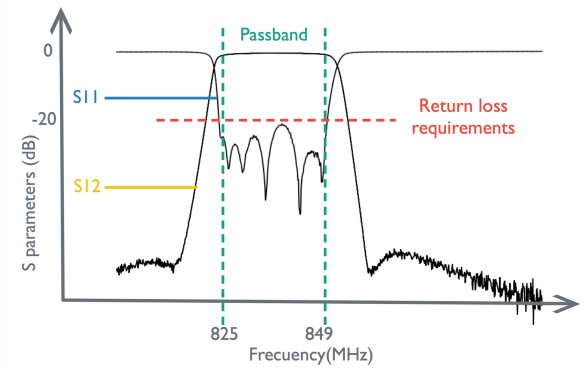

Fig. 4. Design specifications of a filter product shown as S-parameters curves. We currently only consider the return loss requirement.

C. Difficulties in Tuning a Filter Product

The difficulties of tuning a filter whatever manually or automatically are concluded as follows. The biggest challenge is that the S-parameters do not have a fixed relationship with the screw positions. Even small adjustments of the screws may bring enormous influence on the S-parameters. Moreover, the combination of screw positions generating the ideal characteristics may not be unique. People without tuning experience may feel hard to catch the best state and thus never progress towards the right direction. The second challenge is the difficulty in generalization. The filter products have broad varieties. This makes different products differ in their design specifications, i.e. tuning objectives. Therefore, the experience for tuning one kind of product not necessarily be valid for another product kind. Finally, even for the same type of the product, the tuning situations may vary from different individuals. The manufacturing and assembling procedures introduce errors inevitably, which presents a high demand on the robustness of the algorithms.

After carefully studying the tuning problem and deep investigations into existing tuning algorithms, we propose the intelligent tuning based on RL, which is also inspired by the learning process of a human operator.

III. Intelligent Tuning Based on Q-Learning

The work of DeepMind in training a Q-network to play games indicates a good intuition for our work. However, the task of tuning a filter differs largely from playing Atari games. In this section, we mainly describe the bold idea in detail and specify some particular considerations and alternations we make.

A. The Environment

The essence of RL is the interaction with the environment. We need the feedback information returned from the environment to determine the next action. For Atari game experiments, one can simulate the training process in the ALE and the scores are easily returned from the environment. But for our tuning task, it is a totally different story. It is the best to develop an advanced robotic tuning system which not only handles the tuning actions smoothly and accurately, but also automatically returns all the information during the tuning process. Unfortunately, the automatic tuning system is still under way. Therefore, we have to use a simulated way at the present stage. In brief, we take a filter product, randomly insert the screws into different positions, and record the corresponding S-parameters from the VNA. In the same way, we randomly place thousands of screw positions and record the S-parameters accordingly and build a data-driven model to simulate the relationship (Fig. 5). Since the S-parameters curve may have hundreds of sampling points, we apply Principle Component Analysis (PCA) to reduce its dimensionality before training the model.

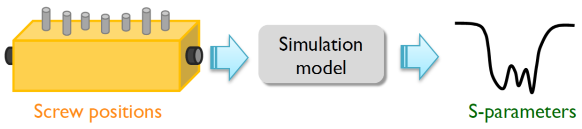

Fig. 5. The simulation of the tuning environment.

The simulation model plays an important role of the environment we need in the RL process. It receives actions (change in screw positions) and gives out new states (simulated S-parameters). This model might be inaccurate because the screw positions are not guaranteed to be recorded without any mistake, but given enough data pairs, we may find more or less a good model to fit them. In the long run, with the help of an automatic tuning system, the data might be more accurately and conveniently recorded.

B. Preliminaries

In order to run the RL algorithm and train the Q-network, some important settings should be defined. In particular, we need specify the state, action, reward and the value function of the tuning task for Q-learning.

In our settings, each state is defined as an S-parameters curve drawn from the VNA, which is represented by a 401×1 vector. The length of the vector depends on the number of sampling frequency points and can be changed if needed. In this work, we temporarily do not consider a deep architecture. Instead, we use PCA to extract features from the raw data. The action is the tuning operation we may take, i.e. the screw rotation angle. Each tuning screw has two valid actions representing two directions of rotation (up and down). All actions share the same fixed rotation amount. The reward is determined by the behavior of each action we take, i.e. the change of the distance from the current curve to the requirement line. We only focus on the sampling points within the passband and define a variable d at each frequency point f to represent the distance. The distance is determined according to the following intuition: if the point is above the return loss (denoted by RL) requirement line (e.g. -20 dB), we define d as the distance from the point to the requirement line; if the point is under that line or just on that line, the value of d is set to be zero.

$$d(f) = \begin{cases} |S_{11}(f) - RL| & if\ S_{11}(f) > RL \\ 0 & if\ S_{11}(f) \leq RL \end{cases} \tag{1}$$

Then the whole distance of the current S-parameters curve is computed by taking the Euclidean norm of all the d values.

$$\text{Cost}(state) = |d(f)|^2 \tag{2}$$

Thus we can define the reward for Q-learning by evaluating the change of S-parameters curve quantitatively. If the cost value of a state is decreasing, the system gets a reward of 1. Otherwise the system gets a reward of 0.

The value function computing the expected future returns, i.e. Q-values is defined on the basis of the Bellman equation. In a word, the optimal action to take at each step is the one which maximizes the expected future return [16]. Then the value function is defined as

$$Q(s_t, a_t) = r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) \tag{3}$$

where s_t, a_t and r_t are the state, action and reward at step t. γ is a discounting factor set to 0.9. Given a state, the system takes an action, obtains the reward and the new state and saves the sequence $\{state,\ action,\ reward,\ new\ state\}$ into the memory. Then a random mini-batch of sequences is extracted from the memory to train the Q-network. After that, the action a_{t+1} which maximizes the Q-value of the next state s_{t+1} is selected to perform in the next cycle.

C. The Learning Model

In order to estimate the value function, one can follow the Bellman equation or build a neural network. Fig. 6 provides the process of Q-learning in more detail. The training of the Q-network is implemented by an optimization process shown in (4). It can be realized by a stochastic gradient descent (SGD), minimizing the distance between the target Q-network and the current hypothesis model.

$$\min_{w,b} \left(\underbrace{r_t + \gamma \max_{a_{t+1}} \hat{Q}\left(s_{t+1}, a_{t+1}; w, b\right)}_{\text{target}} - \underbrace{Q\left(s_t, a_t; w, b\right)}_{\text{hypothesis model}} \right)^2 \tag{4}$$

IV. Experiment

A. Implementation Details

Based on the preliminary theories proposed in the previous sections, we conducted a simple experiment to validate our idea. The filter example we used was a combiner produced by a Chinese communication equipment manufacturer. The combiner had four channels interacted with each other and we only used one channel which was separated from the others and had seven tuning screws (four resonators and three couplings). We asked a tuning expert to tune the product to meet the tuning specifications. For simplicity, we only used two resonator screws as actions because of their huge influence on the overall reflection characteristics, and kept the other five screws remain unchanged. Therefore, we got four valid actions corresponding to clockwise and counterclockwise rotations at a fixed angle of the two screws. The states for Q-learning were drawn from the S-parameters

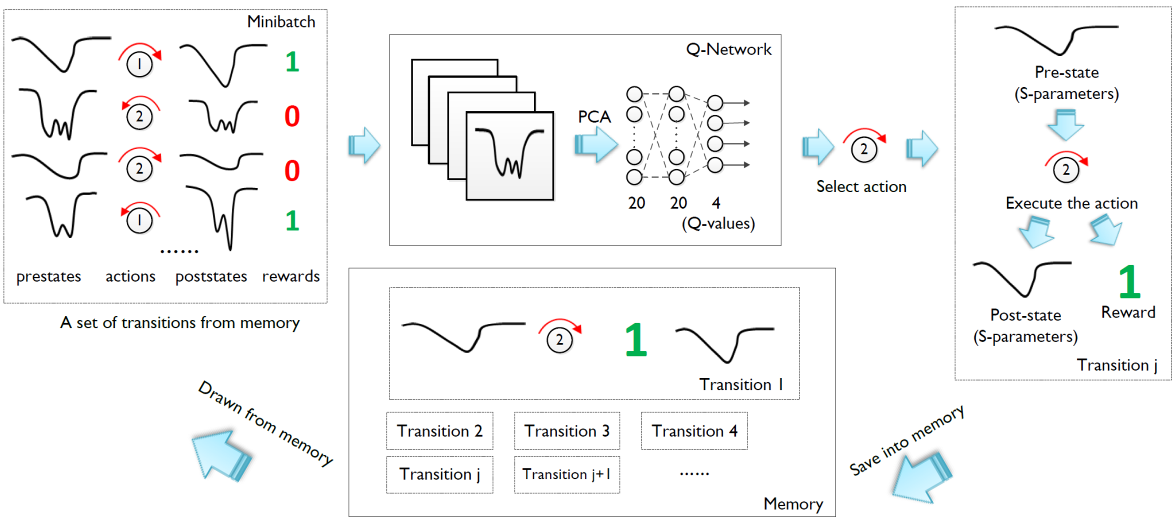

Fig. 6. The overview of the Q-learning system.

curves. We used the simulated environment described in Section III to produce the possible state for each action.

We built a single layer feedforward neural network trained with Back-propagation (BP) algorithm as the Q-network. The architecture was 20-20-4 which was much simpler than CNN. The input to the network was 20 neurons, which was a reduced form of the S-parameters. We used PCA to reduce the dimensionality of the original S-parameters curves into 20 features. The output layer had four neurons indicating the Q-values for four valid actions.

We used a memory size of 500000 to save sequences of $\{state,\ action,\ reward,\ new\ state\}$. Each time when the Q-network was updated, a minibatch of transitions (i.e. 50 sequences) was randomly drawn from the memory to train the Q-network. These details follow the techniques used in [16], [17]. We also used the ε-greedy skill to choose the action to execute. ε is a float number between 0 and 1 representing the probability of randomly choosing an action while $1-\varepsilon$ represents the probability of choosing an action with the largest predicted Q-value. At the beginning of the training, ε was set to 0.99 and decreased as the training proceeded.

$$\varepsilon = 0.99 - \frac{\text{number of passed states}}{1000000} \tag{5}$$

During testing, the ε was fixed at 0.1 to avoid local minimum.

B. Results

We trained the Q-network for 100 epochs, each with 1000 maximum tuning steps. As can be seen from the results, at the beginning of the training process the Q-network model seemed hard to achieve the desired state and failed the tuning after 1000 frames (tuning steps). As training proceeded the step for successfully tuning gradually decreased and converged to about 100 times. After 100 training epochs, we tested the trained Q-network with 100 random states and we found that the probability of successfully tuning reached 95% and in most of the cases the filter could be tuned out within only 50 steps.

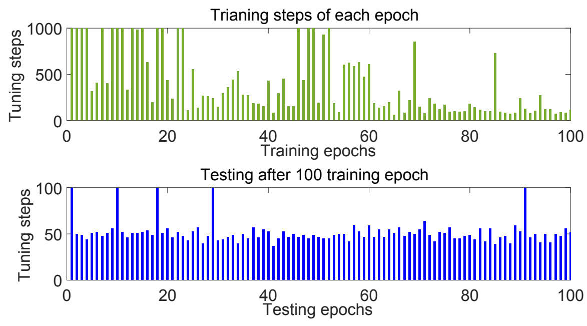

Fig. 7. Tuning times for 100 training and testing epochs for the Q-network realized by a single layer feedforward neural network. After long time of training the Q-network could tune a randomly detuned filter within about 50 tuning steps.

We also took out a set of fixed random states (100 S-parameters) for monitoring their average maximum Q-values during the training (i.e. the maximum of the four Q-values at for each state). Fig. 8 shows that the average maximum Q-value for the fixed set of hand-out states was increasing during the first training epoch and remained at a certain level, indicating that the Q-network gradually grasped the strategies for maximizing the Q-values and became confident in predicting future expected rewards.

Fig. 9 provides a testing tuning process in which the Q-network could successfully drive the curve to reach the desired line. The yellow dashed line represents the return loss objective, which was -21 dB in our experiment. After 48 steps the filter was successfully tuned out.

C. Discussions

The results proved the effectiveness of Q-learning in finding the optimal screw positions. However, we still observed some cases of local minimum when testing the algorithm.

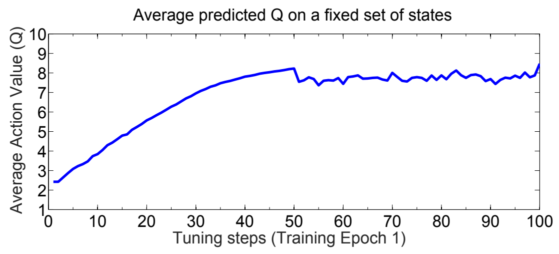

Fig. 8. Average maximum Q-values on a fixed set of randomly generated states for the Q-network realized by a single layer neural network.

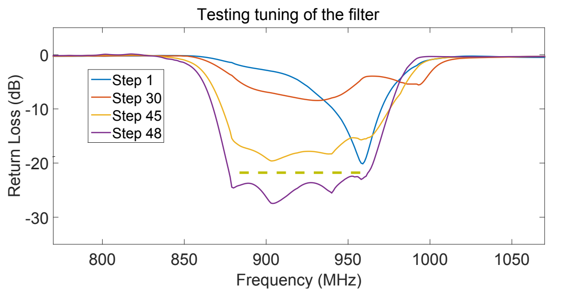

Fig. 9. Successfully tuning process in one of the testing epochs. The algorithm could find the tuning objective in about 50 tuning steps

There are numerous reasons for that. First, in our current work, only the return loss has been considered as an important feature. However, the insertion loss is also essential for a human operator. Both two characteristics should be taken into account to improve the performance. Second, our environment for Q-learning is not a real system but a simulation. The data pairs were collected manually to build the model to temporarily reproduce the real world, which may bring in errors. Third, the metric for computing the rewards is a key issue. We now use the Euclidean norm to compute the distance between the current curve and the target curve. The change trend of the distance was used as a metric for determining the reward. However, this metric may not be comprehensively considered since it may cause the problem of the local minimum. Actually, the reward we currently use is only the process reward, indicating performance of the current step. The tuning objective, i.e. whether the algorithm successfully completes the tuning, should also be considered as part of the reward.

The experimental results also exposes the weakness of RL which is purely by trial and error and takes a large amount of time. In our current experiments, only two screws are used as valid actions. As the number of screws increases, the tuning difficulty may grow exponentially. This may be unacceptable for actual production conditions. We may intend to apply Prioritized Sweeping to improve the performance of Q-learning in time and quality.

V. Conclusion and Future Work

In this work, we introduced the issue of cavity filter tuning in great detail. The difficulties of tuning a filter product were analyzed and the motivation to intelligent tuning research was accordingly raised. Historical overview of recent cavity filter tuning techniques was made. In addition, we proposed a reinforcement learning approach to solve the cavity filter tuning problem. The intelligent tuning system was able to learn the tuning strategies by trail-and-error training of the Q-network and iteratively updating the predicted future returns. The experimental results showed the effectiveness of the Q-network in providing the best action after training for hundreds and thousands of times. In future work, we will on the one hand continue developing the Q-learning approach, and on the other hand make more efforts to extract important features from the raw data, or apply deep learning algorithms to automatically learn features from data.

Acknowledgment

We thank Kristjan Korjus, Junbo Xin and Shaokun Jin for discussions, Qin Lv for work on the data acquisition system.

References

[1] L. Accatino, "Computer-Aided Tuning of Microwave Filters," *IEEE MTT-S Int. Microw. Symp. Dig.*, vol. 50, no. 12, pp. 2781-2788, 1986.
[2] G. Macchiarella and M. Santoniccolo, "An original technique for computer-aided tuning of microwave filters," *The 31st European Microwave Conference (EuMC)*, pp. 1-4, 2001.
[3] COM DEV Ltd., "Robotic Computer-aided Tuning," *Microw. J.*, March, 2006.
[4] H. L. Thal, "Computer-Aided Filter Alignment and Diagnosis," *IEEE Trans. Microw. Theory Tech.*, vol. 26, no. 12, pp. 958- 963, 1978.
[5] J. Dunsmore, "Tuning band pass filters in the time domain," *IEEE MTT-S Int. Microw. Symp. Dig.*, vol. 3, pp. 1351-1354, 1999.
[6] A. R. Mirzai, C. F. N. Cowan, and T. M. Crawford, "Intelligent alignment of waveguide filters using a machine learning approach," *IEEE Trans. Microw. Theory Tech.*, vol. 37, no. 1, pp. 166-173, 1989.
[7] P. Harscher and R. Vahldieck, "Automated computer-controlled tuning of waveguide filters using adaptive network models," *IEEE Trans. Microw. Theory Tech.*, vol. 49, no. 11, pp. 2125-2130, 2001.
[8] V. Miraftab and R. R. Mansour, "Fully automated RF/microwave filter tuning by extracting human experience using fuzzy controllers," *IEEE Trans. Circuits Syst. I Regul. Pap.*, vol. 55, no. 5, pp. 1357-1367, 2008.
[9] O. Moreira-Tamayo, J. Pineda de Gyvez, and E. Sanchez-Sinencio, "Filter tuning system using fuzzy logic," *Electronics Letters*, vol. 30, no. 11. p. 846, 1994.
[10] J. J. Michalski, "Artificial neural networks approach in microwave filter tuning," *Prog. Electromagn. Res. M*, vol. 13, pp. 173-188, 2010.
[11] J. Zhou, B. Duan, and J. Huang, "Influence and tuning of tunable screws for microwave filters using least squares support vector regression," *Int. J. RF Microw. Comput. Eng.*, vol. 20, no. 4, pp. 422-429, 2010.
[12] J. J. Michalski, J. Gulgowski, T. Kacmajor, and M. Mazur, "Artificial Neural Network in Microwave Cavity Filter Tuning," in *Microwave and Millimeter Wave Circuits and Systems: Emerging Design, Technologies, and Applications*, 1st ed., New York, USA: Wiley, 2013, pp. 27-50.
[13] R. J. Cameron, R. Mansour, and C. M. Kudsia, "Computer-Aided Diagnosis and Tuning of Microwave Filters," in *Microwave filters for communication systems: fundamentals, design and applications*, New York, USA: Wiley, 2007, ch. 19, pp. 671-710.
[14] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction.* Cambridge, MA, USA: MIT Press, 1998.
[15] C. J. C. H. Watkins, "Learning from Delayed Rewards," Ph.D. dissertation, King's College, Camb. Uni., Cambridge, UK, 1989.
[16] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," presented at the Annu. Conf. on Neural Info. Proc. Systems (NIPS) Deep Learning Workshop, Lake Tahoe, CA, USA, Dec. 5-10, 2013.
[17] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529-533, 2015.