Deep-RMSA: A Deep-Reinforcement-Learning Routing, Modulation and Spectrum Assignment Agent for Elastic Optical Networks

Xiaoliang Chen[1], Jiannan Guo[2], Zuqing Zhu[2], Roberto Proietti[1], Alberto Castro[1], S. J. B. Yoo[1]

1. University of California, Davis, Davis, CA 95616, USA, Email: sbyoo@ucdavis.edu, xlichen@ucdavis.edu
2. University of Science and Technology of China, Hefei, Anhui 230027, China, Email: zqzhu@ieee.org

Abstract: This paper demonstrates Deep-RMSA, a deep reinforcement learning based self-learning RMSA agent that can learn successful policies from dynamic network operations while realizing cognitive and autonomous RMSA in EONs.

OCIS codes: (060.1155) All-optical networks; (060.4251) Networks, assignment and routing algorithms.

1. Introduction

By setting up variable-sized super-channels with series of spectrally continuous fine-grained frequency slots (FS's), elastic optical networking (EON) offers unprecedented flexibility for spectrum management in the optical layer. Routing, modulation and spectrum assignment (RMSA) is one of the fundamental mechanisms for provisioning in EONs. A number of RMSA schemes based on either heuristic algorithm design [1] or theoretical analysis [2] have been developed in the past years. However, all these schemes apply fixed RMSA policies and hence are unable to adapt to the complicated and dynamic EON conditions, *e.g.*, time-varying demand and spectrum state.

Google recently reported a human-level control paradigm leveraging deep reinforcement learning [3]. Specifically, they parameterized a convolution neural network (also known as *Q*-network) that can learn successful policies (*Q*-values) from high-dimensional sensory data (*i.e.*, images). Inspired by this work, we propose Deep-RMSA, a deep reinforcement learning based self-learning RMSA agent, to realize autonomous and cognitive RMSA for EONs. We structure a deep *Q*-network consisting of multiple convolution and fully connected layers to learn the best RMSA policies regarding different EON states (*e.g.*, connectivity and spectrum utilization) and lightpath requests. The training of the *Q*-network takes advantage of two key ideas from [3], *i.e.*, deployment of target action-value *Q*-network and experience replay, for avoiding the divergence of parameters. We test Deep-RMSA with a six-node EON topology and the simulation results verify its superiority over the baseline RMSA algorithm.

2. Deep-RMSA Design

Deep-RMSA successively learns the optimal RMSA policy based on its perception of network states (*e.g.*, topology, spectrum utilization and in-service lightpaths) and the feedback from the environment (*i.e.*, network operations) using deep reinforcement learning. Fig. 1 illustrates the schematic of Deep-RMSA. In particular, upon receiving a lightpath request $LR(s,d,b)$ (s and d are source and destination nodes, b is the demanded data rate), the RMSA engine fetches the current network state and calls the *Q*-network to compute the estimated action value (*i.e.*, *Q*) for each RMSA solution of *LR*. Conceptually, action value is defined as the cumulative future reward that Deep-RMSA can achieve by taking action (RMSA solution) *A* at state *S*, *i.e.*,

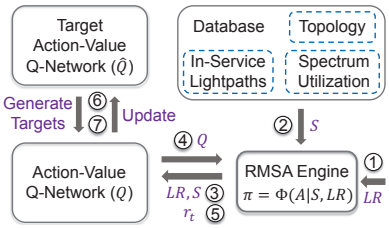

Fig. 1. Schematic of Deep-RMSA.

$$Q(S,LR,A,\pi) = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots, \tag{1}$$

where r_t is the immediate reward, γ is a discount factor imposed on future rewards and $\pi = \Phi(A|S,LR)$ defines the RMSA policy that Deep-RMSA applies. Apparently, taking RMSA solutions with larger *Q*-values grants higher total reward. In this work, a ε-greedy policy is used, with which the RMSA engine takes solutions corresponding to the largest *Q*-values with a probability $1-\varepsilon$, and random provisioning strategy is adopted otherwise. Note that, the random policy enables Deep-RMSA to explore new solutions such that it can avoid being trapped in local optima during the learning process. After the RMSA engine performing the RMSA operation, an immediate reward r_t can be observed. Specifically, $r_t = 1$ if *LR* is successfully provisioned, and $r_t = 0$ otherwise. Then, r_t together with the target generated by the target *Q*-network (estimation of future reward) form the label that is used for training the *Q*-network afterwards, *i.e.*, Deep-RMSA adjusts the parameters of the *Q*-network to make its output closer to the label (more accurate approximation of the real *Q*-value). Finally, the target *Q*-network is periodically updated with the *Q*-network to incorporate the up to date learned experience. The training of Deep-RMSA will be elaborated on in Section 2.2.

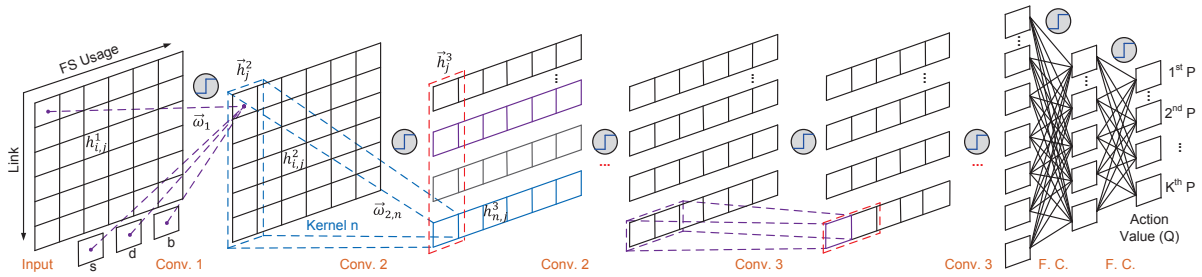

Fig. 2. Structure of deep Q-network. Conv.: Convolution, F.C.: Fully Connected.

2.1. *Q-Network*

Different from traditional Q-learning methods that keep record of each (S,A) entry and update its Q-value using linear approximations, we refer to [3] and parameterize an approximated action-value function $Q(S,LR,A|\vec{\omega}^t)$ using the deep neural network shown in Fig. 1(b) (*i.e.*, Q-network, where $\vec{\omega}^t$ is its parameter set at step t). Basically, the Q-network takes s, d and b of LR and the spectrum utilization on all the fiber links as the input and outputs the Q-value of each RMSA solution. In this work, we calculate K-shortest paths for each request, with each RMSA solution corresponding to the use of one of the K candidate paths. Note that, impairment-aware modulation format adaption and first-fit spectrum assignment scheme [4] are applied throughout all solutions for simplifying the model. Let $h^1_{i,j}$ denote the state of the j-th FS on link i ($h^1_{i,j}$ equates to 1 if the FS is available and 0 otherwise). The first layer incorporates s, d and b and performs a convolution operation (Conv. 1) to transform $h^1_{i,j}$ into $h^2_{i,j}$, *i.e.*,

$$h^2_{i,j} = f\left(\vec{\omega}_1 \cdot \left[h^1_{i,j}, s, d, b\right]^T + \beta_1\right), \forall i, j, \tag{2}$$

where $\vec{\omega}_1$ is the set of parameters, β_1 is the imposed bias and $f(\cdot)$ is the activation function. Here, s and d take the one-hot form, *i.e.*, an l-bit binary array (l is the number of nodes in the EON) with only one bit being equal to 1. Layer 2 is a standard convolution layer (Conv. 2) employing multiple convolution kernels, which each (kernel n) calculates,

$$h^3_{n,j} = f\left(\vec{\omega}_{2,n} \cdot \vec{h}^2_j + \beta_{2,n}\right), \forall j. \tag{3}$$

Since these convolution kernels perceive the state of every spectrum location j across all the links, we expect each of them to extract the state of a path segment from the EON topology. Layer 2 is followed by another one or more layers applying Conv. 2, which further combine the features obtained in layer 2, potentially enabling the extraction of the states of longer end-to-end paths. As the FS's allocated on a lightpath in EONs should be continuous, we next deploy a few Conv. 3 layers to make the Q-network perceive the states of continuous FS-blocks. Specifically, each kernel n in Conv. 3 layers merges every two adjacent nodes from the previous layer x into one node according to,

$$h^{x+1}_{n,i,j} = f\left(\vec{\omega}_{x+1,n} \cdot \left[h^x_{i,2j-1}, h^x_{i,2j}\right]^T + \beta_{x+1,n}\right), \forall i, j. \tag{4}$$

Again, such hierarchical feature extraction mechanism potentially enables the Q-network to learn the states of variable-sized FS-blocks on different paths. Finally, we deploy two fully connected layers to calculate the Q-values using the features learned by convolution layers.

2.2. *Training*

Deep-RMSA iteratively adjusts the parameters of the Q-network to minimize the estimation error over action values. Recall that $Q(S,LR,A|\vec{\omega}^t)$ estimates the Q-value of taking RMSA solution A for LR at state S. Specifically, according to Eq. 1, we have,

$$Q(S,LR,A|\vec{\omega}^t) = \hat{r}_t + \gamma E\left[\max_{A'}\left\{\hat{Q}(S',LR',A'|\vec{\omega}^t_-)\right\}\right], \tag{5}$$

where $\hat{r}_t$ is the estimation of immediate reward, S' is the evolved network state after the RMSA operation for LR, and $\hat{Q}(S',LR',A'|\vec{\omega}^t_-)$ is the target action-value Q-network specifically for estimating the future reward. Here, inspired by [3], we deploy $\hat{Q}(S',LR',A'|\vec{\omega}^t_-)$ which is derived from $Q(S,LR,A|\vec{\omega}^t)$ but is less frequently updated to avoid the oscillation or divergence of the parameters during training. Since the information of future requests is unavailable, average Q-value incorporating all the possibilities of the upcoming request LR' is used to estimate the future reward.

By replacing $\hat{r}_t$ with the observed r_t, we apparently obtain a more accurate estimation of the Q-value. Therefore, the training of $Q(S,LR,A|\vec{\omega}^t)$ aims to minimize the loss function defined as,

$$L\left(\vec{\omega}^t\right)=\left(r_t+\gamma E\left[\max_{A'}\left\{\hat{Q}(S',LR',A'|\vec{\omega}^t_-)\right\}\right]-Q(S,LR,A|\vec{\omega}^t)\right)^2. \quad (6)$$

More specifically, we take advantage of the idea of experience replay from [3], store the RMSA experience $\left(S,LR,A,r_t+\gamma E\left[\max_{A'}\left\{\hat{Q}(S',LR',A'|\vec{\omega}^t_-)\right\}\right]\right)$ in the replay memory and periodically perform batch training by retrieving randomly batches of RMSA experience from the memory. Experience reply breaks the correlations between samples and therefore can potentially reduce the variance of the training [3]. Meanwhile, $\hat{Q}(S',LR',A'|\vec{\omega}^t_-)$ is replaced with $Q(S,LR,A|\vec{\omega}^t)$ every few training steps.

3. Evaluation and Discussion

We evaluate the performance of Deep-RMSA with the six-node EON topology shown in Fig. 3(a). Each link in the EON accommodates 64 FS's. The deep Q-network consists of 2 Conv. 2 layers (each with 16 convolution kernels), 3 Conv. 3 layers (each with 1 convolution kernel) and 2 fully connected layers ($[128,50]$). We calculate $K=5$ candidate paths for each *s-d* pair, *i.e.*, the number of nodes in the output layer is 5. γ and ε are set to be 0.99 and 0.1 respectively. We generate dynamic lightpath requests according to the Poisson process, with *s* and *d* randomly selected and *b* evenly distributed within $[25,100]$ Gb/s. An episode of RMSA operation terminates when every 200 requests are handled (to avoid infinite future reward). We invoke a batch training operation when every 3 episodes elapse and the target Q-network is updated every 2 training operations. We use the shortest path routing and first-fit spectrum assignment algorithm (SP-FF) as the baseline algorithm. Fig. 3(b) shows the performance evolution of Deep-RMSA for various numbers of *LR*s trained. We can observe significant reduction in request blocking probability from Deep-RMSA, which indicates that Deep-RMSA is able to capture successful features from the EON state and learn the correct RMSA policies. Comparison between Deep-RMSA (trained with 600*k* *LR*s) and SP-FF in Fig. 3(c) demonstrates that Deep-RMSA significantly outperforms the baseline algorithm (4.1× blocking reduction in average). Note that, when evaluating the performance of Deep-RMSA, we rely only on the policies learned by the agent and set $\varepsilon=0$ to disable the exploration mechanism.

We should also note that as a very preliminary work of applying deep reinforcement learning to solve networking problems, the proposed Deep-RMSA design still confronts a number of challenges remaining to be addressed. Firstly, the design of the Q-network and the parameters used for the training process can be refined, such that Deep-RMSA can extract better representations of network states and converge to optimal policies quickly even for larger topologies. Meanwhile, different from general learning tasks that can be exactly modeled as Markov decision processes, the actions and rewards of Deep-RMSA are also determined by the received *LR*s in addition to the network states. Therefore, prediction of future *LR*s is required, and the additional complexity induced as well as the potential oscillation of the learned parameters need to be carefully handled.

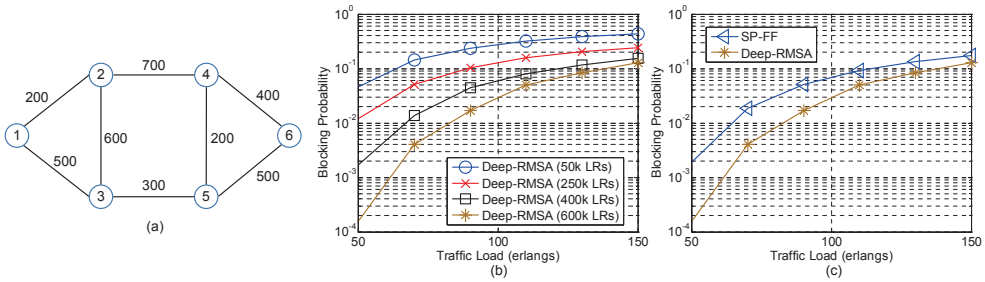

Fig. 3. (a) Six-node EON topology (link length in km) and (b)-(c) results on request blocking probability: (b) performance evolution of Deep-RMSA during training and (c) comparison between Deep-RMSA (trained with 600*k* *LR*s) and SP-FF.

4. Conclusion

In this work, we demonstrated a deep reinforcement learning based self-learning RMSA agent that can learn successful policies from dynamic network operations and realize cognitive and autonomous RMSA in EONs.

References

[1] Z. Zhu *et al.*, *J. Lightw. Technol.*, vol. 31, pp. 15-22, Jan. 2013.
[2] H. Wu *et al.*, *IEEE/ACM Trans. Netw.*, vol. 25, pp. 2391-2404, Aug. 2017.
[3] V. Mnih *et al.*, *Nature*, vol. 518, pp. 529-533, 2015.
[4] X. Chen *et al.*, *J. Lightw. Technol.*, vol. 34, pp. 3867-3876, Aug. 2016.

Acknowledgements: This work was supported in part by ARL W911NF-14-2-0114, DOE DE-SC0016700, and NSF NeTS 1302719.