Mask RSA: End-To-End Reinforcement Learning-based Routing and Spectrum Assignment in Elastic Optical Networks

Masayuki Shimoda(1), Takafumi Tanaka(1)

(1) Network Innovation Laboratories, Nippon Telegraph and Telephone Corporation, Yokosuka, Japan, masayuki.shimoda.wp@hco.ntt.co.jp

Abstract *We propose Mask RSA, an end-to-end deep reinforcement learning-based routing and spectrum assignment for elastic optical networks. Mask RSA masks unassignable choices, and decides routing and spectrum assignment concurrently for higher performance. Mask RSA outperforms KSP-FF under various traffic loads in small and large networks.*

Introduction

Dynamic routing and spectrum assignment (RSA) using deep reinforcement learning (DRL)[1] has been reported to be promising for dynamic network operations. Previous studies fall into two categories: DRL-based routing and DRL-based spectrum assignment (SA). For DRL-based routing, DeepRMSA[2] selects one of K-shortest path (KSP) based on artificial features; it outperforms a heuristic algorithm. Regarding DRL-based SA, the massive number of frequency slot (FS) candidates in elastic optical networks (EONs)[3] is a significant problems in training. R.Shiraki et al. introduced the semi-flexible grid network to reduce the number of FS candidates, allowing a high performance SA algorithm to be trained successfully.[4]

The above studies conduct routing and SA separately, so any resulting algorithm would be sub-optimal. Our research motivation is combining DRL-based routing and DRL-based SA for higher performance. The key difficulty is that the number of assignable FS candidates varies time to time, and even among KSP due to modulation level. Previous work[4] calculates *value* of every candidate (including assignable and unassignable ones) one-by-one, and selects the best *value* candidate. If the previous work is simply extended, it incurs high computation costs since inference of the deep neural network (DNN)[5] runs $(\#FS - \#ReqSlot + 1) \times K$ times. When an algorithm selects one in the fixed number of assignable candidates[2], its decision would be sub-optimal since all assignable ones are not evaluated. Thus, for higher performance, any combination of DRL-based routing and SA must handle the changing number of assignable candidates.

978-0-7381-4679-9/21/$31.00 ©2021 IEEE

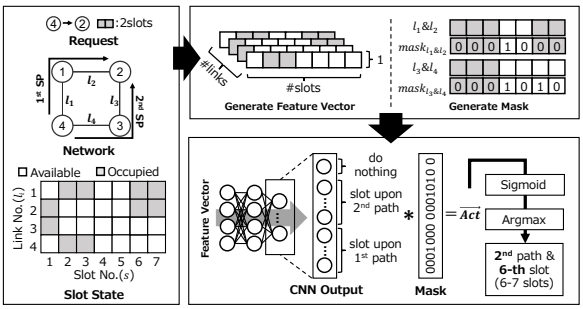

Fig. 1: Overview of Mask RSA inference with $K = 2$. Based on utilization of whole FSs and mask, Mask RSA decides path and FSs concurrently.

In this paper, we proposes *Mask RSA*, an end-to-end DRL-based RSA that solves routing and SA concurrently. To handle the dynamic changes in the number of assignable candidates, we introduce a mask that extracts assignable candidates to simplify candidate selection. Also, the concurrent inference step is run just once, not multiple times, thus reducing computation cost significantly with high performance. This paper is the first to detail of an end-to-end DRL-based RSA that outperforms KSP-FF.

The explanation of Mask RSA is roughly divided into three sections: problem formulation, inference, and training.

Mask RSA: Problem Formulation

RSA is mapped to discrete *Action* space $\mathcal{A}$. Mask RSA implements routing by selecting one of the KSP, and SA is done by selecting the first index of used FSs. Thus, *Action* space in Mask RSA is defined as $\mathcal{A} = \{1, 2, ..., S \times K, S \times K + 1\}$, where S and K are the numbers of FSs and paths, respectively. The option of *Action* is *Do Nothing*; if assignable resources do not exist in KSP, take the *Action* of *Doing Nothing*, which leads to blocking. For example, when selected *Action* no. is 120, (1) do-nothing if $120 > S \times K$, or otherwise (2) the

selected path is $\lfloor 120/S \rfloor$ and start index of used FSs is $120 \bmod S$. This formulation makes *Agent* to select a routing path and FSs concurrently.

Mask RSA: Inference

Fig. 1 overviews Mask RSA inference that features the concurrent decision of RSA. Key elements are (1) feature vector for convolution, (2) RSA assignable mask, and (3) *Action* selection.

(1) Feature vectors: A feature vector is a vector of characteristics of the current *state* that an *Agent* uses as a reference when deciding on an *Action*. Past studies employ partial features, i.e., features of selected routes instead of whole network features and artificial features such as the number of available FSs in routing path. Since the states of the whole network would be more helpful in improving performance, Mask RSA sets the *Action*-decision function to use both entire FS utilization tensor and convolutional neural network (CNN). Let $SU \in \{0,1\}^{L \times S}$ be the slot utilization matrix (0 indicates occupied, and 1 indicates available); $SU_{l,s}$ indicates slot status of the *l*-th link at the *s*-th slot. Feature vectors are made by concatenating the slot utilization status vector $\overrightarrow{SV}_l = [SU_{l,0}, ..., SU_{l,S}] \in \{0,1\}^S$ of all links; $feature\_vector \in \{0,1\}^{L \times 1 \times S}$. This data structure makes it easy to extract features of slot utilization of the whole network by convolution.

(2) RSA Assignable Mask (RSA2M): To deal with dynamic changes of the number of assignable candidates, the masking approach is employed. First, for each routing path, assignable boundary slot mask (ABSM) is generated as a vector whose assignable position at a boundary is 1; otherwise 0. ABSM can avoid spectral fragmentation since it considers only boundaries. Next, *Do Nothing mask* is created; 0 when assignable candidates exist among KSP, otherwise 1. Finally, both *K* ABSMs and *Do Nothing mask* are concatenated, to generate RSA assignable mask (RSA2M) $\overrightarrow{RSA2M} \in \{0,1\}^{S \cdot K+1}$.

(3) *Action* Selection: Let $softmax(\cdot)$ and $argmax(\cdot)$ be softmax and argmax functions, respectively. Mapping function from CNN output to *Action* is written as

$$Action = argmax(softmax(\overrightarrow{out} \circ \overrightarrow{RSA2M})),$$

where $\overrightarrow{out}$ is a CNN output vector, and $\circ$ is Hadamard product. This mask avoids the examination of unassignable choices.

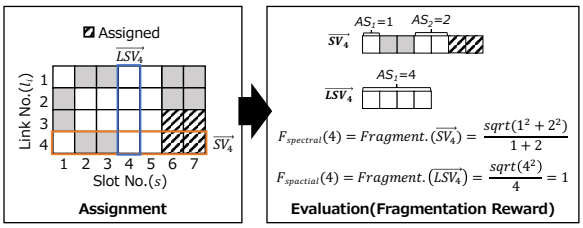

Fig. 2: Fragmentation metric.

Mask RSA: Training

Key elements of the training mechanism in Mask RSA are (1) exploration among assignable choices and (2) multi-metric reward function.

(1) Exploration Storategy: Balancing exploration against exploitation is one of the important issues in achieving high performance. Mask RSA employs Boltzmann exploration, which selects *Actions* based on a probability from a Boltzmann distribution. *Action* selection is performed after RSA2M, which means that exploration runs among only assignable choices. Related work[4] forces RL to learn not only whether slots are assignable, but also which assignable one has the best efficiency. However, we consider that determining whether slots are assignable can be calculated easily, making the use of RL for this unnecessary.

(2) Reward Function: To provide useful information for training efficient RSA algorithms, Mask RSA uses a reward function with dual metrics: fragmentation metric and utilization metric.

$$R = \begin{cases} R_{frag} + R_{util} & \text{if assignable} \\ -1 & \text{otherwise,} \end{cases}$$

The fragmentation metric combines spatial fragmentation and spectral fragmentation as shown in Fig. 2. Let $\overrightarrow{LSV}_s = [SU_{0,s}, ..., SU_{L,s}] \in \{0,1\}^L$ be a vector at the *s*-th slot. Fragmentation metric is defined as follows:

$$Fragmentation(\overrightarrow{v}) = \frac{\sum_i sqrt(AS_i^2)}{\sum_i AS_i},$$
$$F_{spatial}(s) = Fragmentation(\overrightarrow{LSV}_s),$$
$$F_{spectral}(l) = Fragmentation(\overrightarrow{SV}_l),$$

where $sqrt(\cdot)$ is the square root function, and AS_i is the *i*-th number of adjacent-available-blocks. From these metrics, fragmentation reward function is defined as follows:

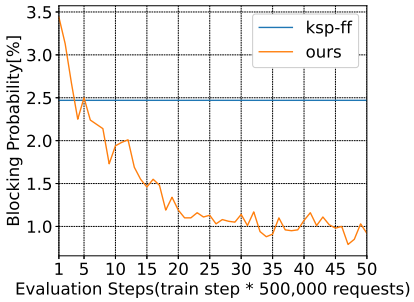

Fig. 3: BPs during training in NSF.

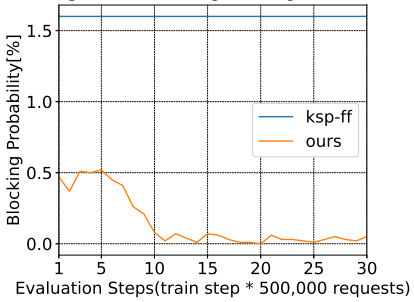

Fig. 4: BPs during training in JPN48.

$$R_{frag} = \frac{\sum_s^S F_{spatial}(s)}{S} + \frac{\sum_{P_i} 2 \cdot F_{spectral}(P_i) - 1}{P},$$

where P is the number of links in the selected routes, and P_i is the *i*-th link No. of the selected routes.

To create an efficient routing algorithm, training uses the following utilization metric:

$$R_{util} = \frac{\sum_l^L \sum_s^S SU_{l,s}}{L \times S}.$$

After every *Action* is conducted, our reward function evaluates the status of FS in detail.

Simulation Settings

We conducted two experiments for evaluating the blocking probability (BP) performance of Mask RSA by comparing it with the heuristic algorithm, KSP first-fit (KSP-FF). The first experiment examined training, and the second one challenged trained *Agent* to various traffic loads. Simulations involved a dynamic traffic scenario, where requests were generated based on a Poisson process following a uniform traffic distribution. For the first experiment, the average arrival rate and service duration were 10 and 12, respectively. The requested slot width was 1-4 slots, and the transmission reaches of BPSK, QPSK, 8-QAM, and 16-QAM signals were determined based on published experiments[6]. RSA-RL framework[7] was used for establishing the simulation environment. Tested networks were NSF Network (14 nodes and 13 links) and JPN48 Network (48 nodes and 82 links). Spectral slot was set to 50 GHz, and each link had 80 slots. DRL algorithm used was PPO (Proximal Policy Optimization)[8], and the optimization algorithm was Adam[9] (learning rate $1e-7$, epsilon $1e-7$, and weight decay $1e-4$).

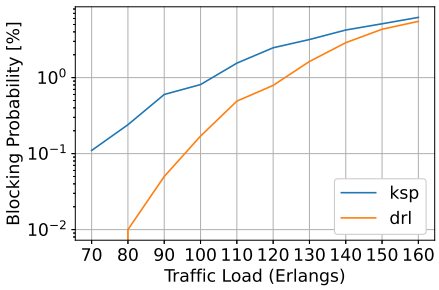

Fig. 5: BPs versus various traffic loads in NSF.

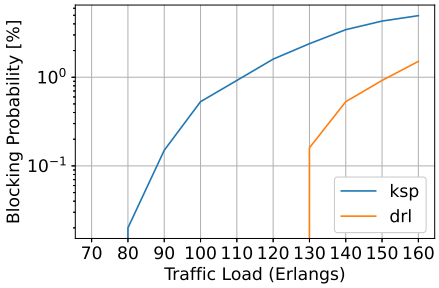

Fig. 6: BPs versus various traffic loads in JPN48.

Simulation Results

The first experiment evaluated the performance with 10,000 requests after training with 500,000 requests. Fig. 3 and Fig. 4 show BPs versus evaluation steps until training convergence. In NSF, our minimum BP was 0.79% at the 47-th step, while that of KSP-FF was 2.47%. As for JPN48, our minimum BP was 0.0% at the 19-th step, while that of KSP-FF was 1.6%.

Next, trained *Agent* from the previous experiment was subjected to various traffic loads. Fig. 5 and Fig. 6 show request BPs versus traffic loads from 70 to 160. In both networks, Mask RSA outperformed KSP-FF even when the traffic load differed from that used in training.

From two experiments show that the masking approach enabled *Agent* to attain efficient RSA regardless of the network size or the traffic load used for training; accordingly, Mask RSA outperformed KSP-FF.

Conclusion

This paper proposed Mask RSA. Its concurrent decision approach with mask outperformed KSP-FF, and enabled efficient RSA algorithms to be created for dynamic network operations in EONs.

References

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning", *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[2] X. Chen, B. Li, R. Proietti, H. Lu, Z. Zhu, and S. J. B. Yoo, "Deeprmsa: A deep reinforcement learning framework for routing, modulation and spectrum assignment in elastic optical networks", *Journal of Lightwave Technology*, vol. 37, no. 16, pp. 4155–4163, 2019.

[3] O. Gerstel, M. Jinno, A. Lord, and S. J. B. Yoo, "Elastic optical networking: A new dawn for the optical layer?", *IEEE Communications Magazine*, vol. 50, no. 2, s12–s20, 2012.

[4] R. Shiraki, Y. Mori, H. Hasegawa, and K. Sato, "Dynamically controlled flexible-grid networks based on semi-flexible spectrum assignment and network-state-value evaluation", in *2020 Optical Fiber Communications Conference and Exhibition (OFC)*, 2020, pp. 1–3.

[5] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning", English (US), *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015, ISSN: 0028-0836. DOI: 10.1038/nature14539.

[6] A. Bocoi, M. Schuster, F. Rambach, M. Kiese, C.-A. Bunge, and B. Spinnler, "Reach-dependent capacity in optical networks enabled by ofdm", in *2009 Conference on Optical Fiber Communication*, 2009, pp. 1–3. DOI: 10.1364/OFC.2009.OMQ4.

[7] *RSA-RL*, https://github.com/Optical-Networks-Group/rsa-rl, 2020.

[8] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms", *arXiv preprint arXiv:1707.06347*, 2017.

[9] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization", *arXiv preprint arXiv:1412.6980*, 2014.