
CHALMERS
UNIVERSITY OF TECHNOLOGY

# Deep reinforcement learning for proactive spectrum defragmentation in elastic optical networks [Invited]



Citation for the original published paper (version of record):
Etezadi, E., Natalino Da Silva, C., Diaz, R. et al (2023). Deep reinforcement learning for proactive spectrum defragmentation in elastic optical networks
[Invited]. Journal of Optical Communications and Networking, 15(10): E86-E96.
http://dx.doi.org/10.1364/JOCN.489577



(article starts on next page)

# Deep reinforcement learning for proactive spectrum defragmentation in elastic optical networks [Invited]

EHSAN ETEZADI[1,*], CARLOS NATALINO[1], RENZO DIAZ[2], ANDERS LINDGREN[2], STEFAN MELIN[2], LENA WOSINSKA[1], PAOLO MONTI[1], AND MARIJA FURDEK[1]

[1] *Department of Electrical Engineering, Chalmers University of Technology, 412 96 Gothenburg, Sweden*
[2] *Telia Company, 169 94 Solna, Sweden*
[*] *ehsanet@chalmers.se*

*Compiled March 14, 2024*

**The immense growth of Internet traffic calls for advanced techniques to enable the dynamic operation of optical networks, efficient use of spectral resources, and automation. In this paper, we investigate the proactive spectrum defragmentation (SD) problem in elastic optical networks and propose a novel deep reinforcement learning-based framework *DeepDefrag* to increase spectral usage efficiency. Unlike the conventional, often threshold-based heuristic algorithms that address a subset of the defragmentation-related tasks and have limited automation capabilities, DeepDefrag jointly addresses the three main aspects of the SD process: determining when to perform defragmentation, which connections to reconfigure, and which part of the spectrum to reallocate them to. By considering services attributes, spectrum occupancy state expressed by several different fragmentation metrics, as well as reconfiguration cost, DeepDefrag is able to consistently select appropriate reconfiguration actions over the network lifetime and adapt to changing conditions. Extensive simulation results reveal superior performance of the proposed scheme over a scenario with exhaustive defragmentation and a well-known benchmark heuristic from the literature, achieving lower blocking probability at a smaller defragmentation overhead.**

https://doi.org/10.1364/JOCN.489577

## 1. INTRODUCTION

The tremendous growth of bandwidth-intensive applications that have dynamic behavior and high performance requirements (e.g., high-definition video on demand, cloud computing, Internet of Things, content delivery networks) puts a significant strain on the optical backbone networks. Dynamic, automated, and resource-efficient network operation is essential to fulfilling these requirements. Elastic optical networks (EONs) [1] enable both fine-grained spectrum slicing and high-capacity super-channels that match the spectrum requirements of service requests. However, EONs are prone to spectrum fragmentation (SF), where the requested service bandwidth exceeds the number of continuous and contiguous free spectrum slots. In dynamic traffic scenarios, the establishment and tear-down of optical connections often exacerbates SF by scattering relatively small unoccupied spectral gaps across the available fiber bandwidth [2, 3]. When these spectral gaps are insufficient to support incoming service requests, SF has a direct, detrimental impact on the blocking probability of service demands [4].

To alleviate the impact of fragmentation, spectrum allocation should be consolidated to leave as few unusable spectral gaps as possible. This process is called spectrum defragmentation (SD), and is known to improve spectrum grid utilization and reduce service blocking ratio (SBR) [5]. The goal of SD is to make the spectral gaps larger and better aligned across the network links. This enables for accommodating more services, thus maximizing the use of the spectrum.

(i) *When to reconfigure?* Deciding on the best time to perform SD among arbitrary service arrivals and departures.

(ii) *What to reconfigure?* Determining the number and the order of connections to be reallocated.

(iii) *Where to reallocate the connections to?* Finding new spectral resources for the reconfigured connections.

The problem of minimizing spectrum fragmentation by reconfiguring a minimum number of connections has been shown to be NP-complete in static traffic scenarios [6]. Traffic dynamicity further increases the problem complexity due to the constantly changing set of connections in the network. Hence, tractable optimization approaches are needed to solve the highly complex problem of dynamic SD.

SD approaches can be classified into two main schemes: reactive and proactive [2]. Reactive approaches are triggered by service blocking. Proactive approaches are executed without

waiting for the blocking to occur. They typically monitor network performance metrics to find the best time for SD or perform it periodically. These schemes are further classified into two types, namely with or without rerouting of connections [5]. The latter approaches only reallocate the spectrum of the connections, while the former may modify their routes as well. SD approaches that interrupt running services are referred to as non-hitless, while those that do not cause any traffic disruption are known as hitless [7]. Push-pull retuning is a hitless approach where the spectrum occupied by a connection is first expanded until it includes both the original and the targeted spectrum slots and then shrunk to include only the targeted slots [8]. Another hitless SD approach is make-before-break, where an additional connection is established over the target route and spectrum before tearing down the original one, allowing for a spectrum jump [5]. It should be noted that make-before-break is considered non-disruptive specifically for the optical layer, while interruptions may occur at the higher layers depending on the employed protocol and/or rerouting strategy.

While SD has been shown to decrease the SBR, it also imposes a reconfiguration overhead that is not desirable by network operators. Depending on the SD approach, the overhead may entail terminating, reallocating, and reestablishing selected connections. Consequently, performing SD too frequently or on an excessive number of connections may drastically increase the complexity of network control and management. In fact, the frequency of SD cycles and the number of connection reallocations within each cycle are used to measure the SD overhead [9]. This indicates that the potential SBR improvement and the corresponding overhead should be considered jointly and flexibly in the design and evaluation of SD approaches. Existing SD strategies (e.g., [10, 11]) handle only a subset of the aforementioned SD tasks and they do so by utilizing deterministic thresholds and policies, which makes them inapplicable to dynamic settings with changing network conditions.

Different from the deterministic, threshold-based policies, in reinforcement learning (RL), the algorithm makes decisions by learning from the environment, aiming at maximizing the long-term reward without being explicitly programmed. RL has recently been demonstrated as a promising technique for solving large-scale online control tasks, e.g., routing and resource assignment in EON [12, 13] and 5G network slicing [14]. The deep reinforcement learning (DRL) method combines RL with deep neural networks (DNNs), allowing complex systems to be analyzed for high-dimensional input data, including traffic matrices and images. One of the valuable capabilities of some DRL agents is to learn online and adapt to changing network conditions. Through the online learning process, the DRL agent continuously interacts with the environment, receives feedback, and updates its policy accordingly.

To utilize the merit of DRL in automating SD, we proposed DeepDefrag, a novel DRL-based framework that jointly addresses all of the tasks involved in the SD process: determining when to perform defragmentation, which connections to reconfigure, and which part of the spectrum to reallocate them to [15]. DeepDefrag considers the network state to select the most appropriate course of action and can take into account the priorities of a network operator, such as minimizing the number of SD cycles and connection reallocations. Our preliminary study in [15], considered only a subset of connections as eligible for reconfiguration and did not examine the spectrum occupancy state in the decision-making process. This paper extends and improves DeepDefrag by (i) considering all connections in the network as candidates for reconfiguration, (ii) considering full information about the spectrum occupancy, including different fragmentation metrics, and (iii) revising the reward function to allow for a more comprehensive evaluation of the impact of actions. An evaluation of the impact of different penalties modeling the SD overhead, and of changes in the traffic load is also included. The performance of the proposed DeepDefrag framework is evaluated through comparison with several heuristic algorithms from the literature. The simulation results reveal that DeepDefrag outperforms the well-known existing older-first first-fit (OF-FF) algorithm in different aspects. Moreover, it yields SBR values close to an approximated (heuristic) lower bound obtained through exhaustive spectrum defragmentation. We demonstrate that, unlike preconfigured algorithms like OF-FF, DeepDefrag can effectively handle changes in the traffic load by considering the new situation and learning the optimal policy for the updated circumstances. This adaptability allows DeepDefrag to continuously optimize its actions and make informed decisions that align with the current network conditions, resulting in improved performance and spectrum resource utilization.

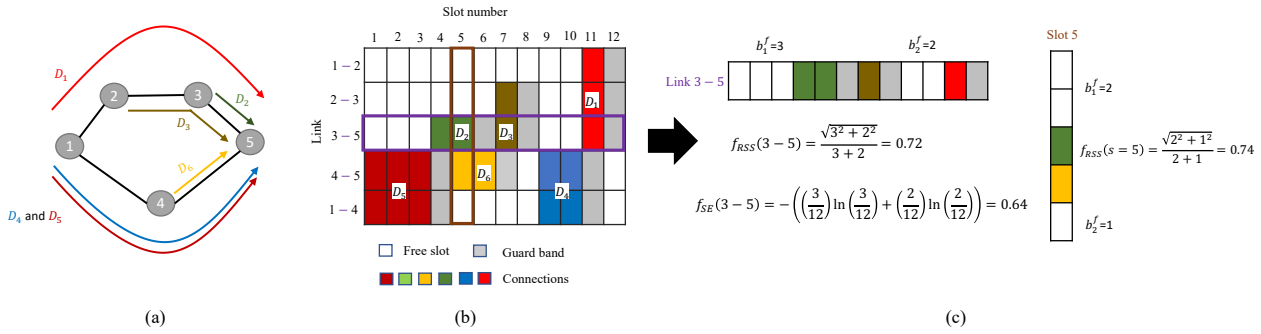
## 2. RELATED WORK AND BACKGROUND

### A. Spectrum fragmentation metrics

In general, spectrum fragmentation metrics in EONs measure the efficiency of spectral utilization. A better fragmentation metric value indicates that the occupied frequency slots are used more efficiently, with fewer unusable gaps between occupied slots. These metrics help network operators monitor and optimize the utilization of optical spectrum resources, ensuring high performance and efficient use of available resources. The SF issue in EONs has been widely analyzed in the literature and several fragmentation metrics have been introduced. Wang et al. [16] present the concept of utilization entropy to measure the level of optical spectrum fragmentation. Authors in [17] define an external fragmentation metric as a ratio of the largest free contiguous fragment of the spectrum and the sum of the size of all free spectral fragments. The spectrum compactness metric from [18] indicates the occupation of spectrum on a link or in the network by calculating the difference between the maximum and the minimum indices of occupied slots. Takita et al. [19] define the high slot mask metric as an indicator of the maximum number of occupied spectrum slots in the network.

In this paper, we incorporate the spectrum occupancy state into the DRL agent to enhance its understanding of the network state. However, considering a large number of SF metrics is impractical due to the increased complexity of computing the metrics at every step, and the potential increase in the training time of the DRL agent. As highlighted in [20], different metrics capture various aspects of SF, and the selection of metrics depends on the specific requirements and context. Therefore, we carefully chose three metrics to measure the fragmentation state of the network: the number of cuts [21], the Shannon entropy (SE) [17], and the root of sum of squares (RSS) [20]. In support of our choices, previous studies such as [12] have demonstrated the suitability of incorporating RSS into the reward function of DRL agents for routing and spectrum assignment in EONs. Furthermore, the effectiveness of the number of cuts and SE in enhancing network utilization has been highlighted in [21] and [22], respectively.

In Fig. 1, we exemplify the parameters and calculation of these metrics with a snapshot of a simple network example with




(a) (b) (c)

**Fig. 1.** A simple network example serving six connections (a). The spectrum occupancy state (b). Shannon entropy and root of sum of squares metrics (c).

five nodes and four links, each with 12 spectrum slots. The considered network state comprises six connections established in the network, denoted by $D_1$ to $D_6$. The connection routes are depicted in Fig. 1(a), while the spectrum assignment state for each link is shown in Fig. 1(b). We assume one spectrum slot is used as guardband between adjacent connections on a link. The notation includes the following parameters: $e$ is the index of a link, $E$ is the total number of links, $s$ is the index identifying a spectrum slot on a link, $S$ is the total number of slots of a link, $b_i^f$ is the size of free spectrum block $i$, and $N$ is the number of free spectrum blocks.

The number of cuts denotes the number of links with free adjacent spectrum slots on the path selected for a connection. The SE values for a link and the entire network are formulated by (1) and (2), respectively. Equation (3) defines the RSS metric for a link $e$, while it can be calculated for a slot $s$ analogously. The two metrics are referred to as spectral and spatial fragmentation, respectively [12]. Finally, the RSS metric for the network is calculated as the average of spectral and spatial RSS metrics over all the links and slots in (4). A higher SE value implies higher fragmentation, while a higher RSS value implies lower fragmentation.

$$f_{SE}(e) = -\sum_{i=1}^{N} \frac{b_i^f}{S} \ln \frac{b_i^f}{S} \quad \textbf{(1)}$$

$$F_{SE} = \frac{\sum_e^E f_{SE}(e)}{E} \quad \textbf{(2)}$$

$$f_{RSS}(e) = \frac{\sqrt{\sum_i^N (b_i^f)^2}}{\sum_i^N b_i^f} \quad \textbf{(3)}$$

$$F_{RSS} = \frac{\sum_s^S f_{RSS}(s)}{S} + \frac{\sum_e^E f_{RSS}(e)}{E} \quad \textbf{(4)}$$

Figure 1(c) shows how the values of the SE and RSS metrics for link $(3-5)$ and slot number $(5)$ (highlighted with frames) are calculated. In the example, connection $D_1$ occupies slot 11, so slot 10 is checked to calculate the number of cuts. Slot 10 is free on all three links included in the path of $D_1$, so the number of cuts is equal to 3 for this connection. The number of cuts for $D_3$ is equal to 1 since slot 6 is occupied on link $(3-5)$ and free on link $(2-3)$.

### B. Spectrum defragmentation techniques

In recent years, extensive research has examined spectrum fragmentation and its mitigation, relying on integer linear programming (ILP) formulations, (meta)heuristics and machine learning techniques. The work in [10] models the proactive parallel connection reconfiguration in EONs mathematically as an ILP formulation and studies the complexity of the problem. ILP models for three defragmentation techniques denoted as Push-Pull, Hop-Tuning, and Replanning are proposed in [11]. The authors in [23] delve deeper into the trade-off between SD gain in terms of fragmentation ratio and the extent of connection disruptions in terms of reconfiguration delays. They create a mathematical model to optimize high-slot marks as the fragmentation metric across all links.

Heuristic and metaheuristic algorithms are also widely used to tackle the fragmentation problem and decrease the SBR. The authors in [24] investigate heuristic algorithms for hitless bandwidth defragmentation, including spectrum sweeping and hop tuning. In [9], SD is performed periodically, and connections are selected for reallocation based on service attributes. Older-first (OF) selects the longest-lasting connections, longer-lasting-first (LLF) selects those with the longest remaining holding time, bigger-first (BF) selects the connections with the biggest size, and longer-path-first (LPF) selects those with the longest path for reconfiguration. A first-fit (FF) spectrum assignment policy is employed to reallocate spectrum slots. Simulation results indicate that the algorithms exhibit similar performance, with the OF algorithm demonstrating the best performance within a marginal difference of one percent.

In [25], different SD heuristic algorithms, including lowest-slot-index-first, holding-time-aware, and proactive-reactive defragmentation, are compared based on their blocking probability, entropy, and bandwidth fragmentation ratio. The authors in [26] propose a reactive disruptive scheme and a proactive non-disruptive scheme. Both schemes utilize the holding time information of existing connections to minimize the SBR. The authors in [27] use a meta-heuristic nature-inspired optimization technique called jellyfish search optimization to solve spectrum defragmentation and show performance improvement compared to the state-of-the-art heuristic algorithms.

SD has recently benefited from adopting machine learning techniques. An application from [28] uses unsupervised machine learning to rearrange the fragmented spectrum based on connection clustering. In [29], Elman neural networks (NNs) are employed to predict traffic demands, and a two-dimensional rectangular packing model is used to allocate spectrum in a way that minimizes fragmentation. A machine learning-assisted signal-quality-aware proactive defragmentation scheme for the C + L band system is proposed in [30]. The scheme prioritizes

minimizing the fragmentation index and quality of transmission (QoT) maintenance for the defragmentation algorithms.

### C. Reinforcement learning in optical networks

Multiple studies have explored the efficacy of RL for solving resource allocation problems in EONs, such as the DRL-based routing, modulation, and spectrum assignment (RMSA) algorithm in [31], which performs joint routing and spectrum assignment by masking infeasible options to improve the blocking probability performance. In [31], the connection admission control and routing and spectrum assignment (RSA) problems are modeled as a Markov decision process (MDP), and the concept of deterministic policy for the RSA problem in the policy iteration algorithm is introduced. The work in [32] demonstrates that DRL is an effective alternative to established and well-known solutions for optical network optimization problems, including routing and wavelength assignment (RWA). A DRL approach for resource provisioning in a dynamic multi-band EON is studied in [33] and compared to a heuristic algorithm. The authors in [34] investigate the problem of global optimization of network performance in a survivable EON use case and propose a DRL-based algorithm with the objective of improving the overall network performance in terms of cost value and survivability, where two RL agents are utilized to provide working and protection paths. In [35], DRL is used to tackle the on-demand, reactive hitless SD problem. Upon a failure of an incoming service request, the DRL agent selects one of the pre-defined schemes that increase the size of the fragmented spectrum to accommodate blocked services. To the best of our knowledge, the merit of DRL in solving proactive SD has not been evaluated yet in spite of its strong potential to solve complex problems. Therefore, in the next sections, we propose a DRL-based framework for proactive SD and evaluate its performance against heuristic algorithms.

## 3. PROBLEM FORMULATION

We consider a network topology represented by a graph $G(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ represent the set of nodes and fiber links, respectively. We model a service request from node $s$ to $d$ ($s, d \in \mathcal{V}$) as $D_i(s_i, d_i, b_i, a_i)$, with $b_i$ and $a_i$ denoting the requested bit rate, arrival time, respectively. To provision service requests, the network must solve the RMSA problem of finding an end-to-end physical route, determining the modulation format, and allocating the required spectral resources. We adopt the model from [13] to decide on the modulation format limited by the length of the selected path. The number of required spectrum slots, denoted by $n_i$, is determined by $\lceil b_i/(12.5 \times m) \rceil$, where 12.5 Gbit/s is the data rate that a spectrum slot of BPSK signal can support, and $m$ is spectral efficiency of the selected modulation format. A connection is established if a path with $n_i + 1$ continuous and contiguous spectrum slots is found, where the extra slot accounts for the guardband. If these spectrum resources are not found, the service request is blocked.

We consider a dynamic EON scenario in which service requests arrive and depart throughout the network operation. At any given time, the spectrum grid state information about the existing connections is known. In the considered proactive SD scenario, only spectrum reallocation is performed, without connection rerouting. The goal is to reallocate the spectrum of a subset of connections to consolidate the free available spectrum for future use. We consider a hitless, make-before-break scenario. The first challenge of proactive SD is to find the best time to perform a defragmentation operation. The second challenge is to determine the set of connections and the order in which they should be reconfigured. Finally, the new spectrum slots must be identified for the services.

## 4. THE DEEPDEFRAG SCHEME

### A. System model

Figure 2 illustrates the DeepDefrag scheme under dynamic traffic, where SD cycles are triggered in response to connection departures. When a connection departs, DeepDefrag assesses whether to initiate a defragmentation cycle or not. If the decision is to start a new SD cycle, DeepDefrag selects a connection to reconfigure and identifies the target spectrum. This process is repeated until DeepDefrag decides to conclude the cycle. The left-hand inset at the top of the figure provides an example of an SD cycle that includes three connection reallocations. The DeepDefrag scheme uses two variables to model the SD process. $\theta \in \{0, 1\}$ is a network control flag with a value of 1 when an SD cycle is in progress, and 0 otherwise. The selected action is denoted as $\alpha$, with value equal to the index of the connection selected for reconfiguration, or $\alpha{=}\emptyset$ to represent the *stop* action.

As shown in Fig. 2, $\theta{=}0$ and $\alpha \neq \emptyset$ when DeepDefrag starts an SD cycle and reallocates the first connection. At this point, DeepDefrag has the option to either continue the ongoing SD cycle by reallocating another connection or to terminate it by returning $\alpha = \emptyset$. In this particular example, DeepDefrag decides to reallocate two other connections ($\theta{=}1$, $\alpha \neq \emptyset$), and then stops the SD cycle ($\theta{=}1$, $\alpha{=}\emptyset$). Note that only sequential reconfiguration of individual connections is considered (i.e., two or more connections are not reconfigured jointly). The time between two sequential SD cycles is referred to as the SD period. DeepDefrag can also decide not to trigger an SD cycle upon a connection departure. Fig. 2 illustrates this scenario after the departure of the third connection, where the actions and variables involved in the decision-making process are presented in the inset on the right hand side. Here, the SD cycle is not currently in progress ($\theta{=}0$), and the scheme chooses to take no action ($\alpha{=}\emptyset$).

DeepDefrag considers all connections as candidates for reallocation and examines several options to reassign the spectrum. All available spectrum blocks that can accommodate the connection are enumerated, and each option represents reallocating a connection to the beginning of every available free block along its path. Let us consider the same example as in Fig. 1 and analyze the reallocation options for connections $D_1$ and $D_4$, shown in Fig. 3. For connection $D_1$, which is currently using slot 11, two free blocks along links 1–2, 2–3, and 3–5 can be considered as alternatives: slots 1–3 and 9–12. Therefore, connection $D_1$ has two alternative spectrum options, which are at the beginning of the two candidate blocks, denoted as $o_1^1$ and $o_1^2$. The alternative for connection $D_4$ is at the beginning of the only free block on links 1–4 and 4–5, i.e., slots 8–12, denoted as $o_4^1$ in the figure. It should be noted that one option is available for connection $D_2$ and one for connection $D_3$, which are not shown in the figure. By combining the event model from Fig. 2 and the intuition developed in Fig. 3, a DRL agent can be designed to solve the SD problem.

### B. Markov decision process modeling

The DeepDefrag scheme uses DRL to solve the proactive SD problem discussed in the previous section. DRL is a machine learning technique focused on solving control problems, where

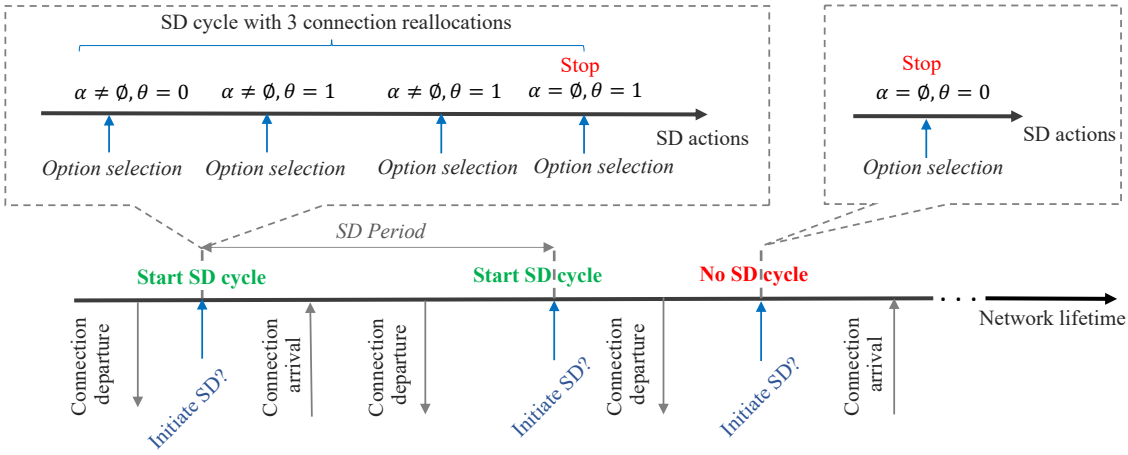



**Fig. 2.** The DeepDefrag scheme decisions taken and implemented during network operation.

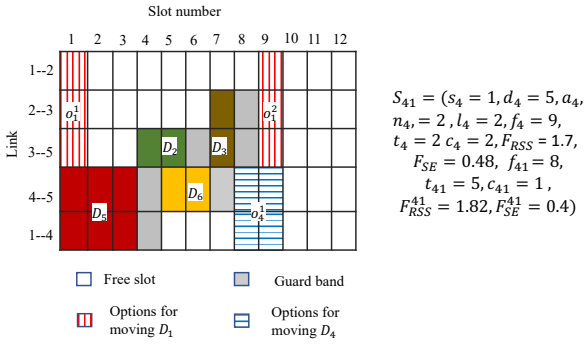



**Fig. 3.** Different options for spectrum reallocation of the connections, and the state representation

a DRL agent interacts with the environment and has the objective of maximizing a notion of cumulative reward. Such control problems are commonly modeled as MDPs. The following section outlines the MDP model of DeepDefrag, which covers the definitions of the observation space, action spaces, and reward function.

### B.1. Observation space

The observation space should provide the DRL agent with enough information to characterize the current state of the environment (i.e., the optical network in our case). The observation space of DeepDefrag consists of several components. The state representation for reallocation option $j$ of connection $D_i$ is denoted as $S_{ij}$, and defined as follows:

$$S_{ij} =< s_i, d_i, a_i, n_i, l_i, f_i, t_i, c_i, F_{RSS}, F_{SE}, f_{ij}, t_{ij}, c_{ij}, F_{RSS}^{ij}, F_{SE}^{ij} > ,$$

where $l_i$ is the number of links along the path allocated to the connection, $f_i$ is the currently assigned starting spectrum slot, $t_i$ is the total number of available slots along path, and $c_i$ is the number of cuts (as defined in Sec. 2.A) along the current path. The RSS and SE metrics for the current state of the network are represented by $F_{RSS}$ and $F_{SE}$, respectively. $f_{ij}$, $c_{ij}$, and $t_{ij}$ represent the new candidate starting slot, the number of cuts, and the size of the free spectrum block used by option $j$ for reallocating connection $D_i$, respectively. Finally, $F_{RSS}^{ij}$, $F_{SE}^{ij}$ are the RSS and SE metrics of the network if $D_i$ is chosen to be reallocated to option $j$. The example of the state representation for option $o_4^1$ is represented in Fig 3.

### B.2. Action space

The action space represents the set of all actions the agent can perform in a specific environment. As shown in Fig 2, for our environment, the agent can select one of the available options in each decision step. In the DeepDefrag environment, we denote the set of possible actions as $\mathbb{J}$. Each action is characterized by the tuple <$D_i$, $f_{ij}$>, which represent the connection and the new starting slot of the selected option, respectively. The set $\mathbb{J}$ also contains the $\emptyset$ action, which denotes termination of an SD cycle in progress, or the absence of initiating a new one.

### B.3. Reward function

The reward function is a function that provides a numerical score based on the state of the environment and the action taken by the agent. The critical challenge of using RL is to find the appropriate reward function that reflects the behavior of the environment and steers the agent towards the most suitable policy. The reward value $r_i$ for DeepDefrag is defined by (5).

$$r_i = \begin{cases} -\frac{\log_{10} SBR}{3} & \theta \in \{0,1\} \wedge \alpha{=}\emptyset \\ -\frac{\log_{10} SBR}{3} - Ps - Pe & \theta = 0 \wedge \alpha \neq \emptyset \\ 1 + \frac{\log_{10}(F_{RSS}^{ij} - F_{RSS})}{3} - Pe & \theta = 1 \wedge \alpha \neq \emptyset \end{cases} , \quad (5)$$

The SBR is the main term of the reward function due to its direct representation of the objective of performing SD. The value of SBR is defined as the ratio between the blocked and the total number of processed service requests. The design of the reward function aims to strongly penalize even a slight increase of the SBR. Therefore, the logarithm of the SBR is used in the reward function to amplify the small changes of SBR when the agent chooses not to start an SD cycle ($\alpha{=}\emptyset$, i.e., the first term of

(5)). To limit the SD overhead, each new SD cycle and each connection reallocation are associated with a penalty, denoted by $Ps$ and $Pe$, respectively. Both penalties are considered in the reward function whenever the agent initiates a new SD cycle by reallocating a connection, i.e., the second term in (5). The third term in (5) refers to the reward for connection reallocation within an SD cycle in progress. As mentioned in Sec. 2.A, a higher value of the RSS metric implies lower fragmentation. Hence, the agent uses the difference between the RSS metric before and after reconfiguration to evaluate the benefit of the connection reallocation. The logarithmic function is employed to guarantee that a small increment of the RSS metric yields a significant increase of the reward value. The penalty for connection reallocation is also considered. The logarithm addends in the reward function are normalized using a factor of 3 to conform to the range between zero and one. This normalization process facilitates setting the values of the penalties relative to the other components of the reward. It also helps the DRL agent to learn more efficiently by balancing the magnitudes of the reward values and preventing them from becoming too large or too small.

The penalty values in the reward function (i.e., $P_s$ and $P_e$) are determined by the network operator based on the costs associated with each proactive SD cycle and reallocation, respectively. In this work, the values of the penalties are selected based on the target resulting SD overhead.

### C. Learning Process using Deep Q-Networks

We utilize the deep Q-Networks (DQN) algorithm [36] to determine the policy for the proposed SD approach. The objective of the DQN algorithm is to learn a policy that maximizes the long-term reward by estimating the state-action values, also known as $Q$-values, using a DNN. These $Q$-values represent the expected long-term reward for each state-action pair. To approximate the $Q$-values, we employ an NN, which takes the network state $S_t$ as input. The output of the NN provides the predicted state-action values for all possible actions given the input state. For training, we utilized two NNs with the same architecture. One network, called the $Q$-Value-Network, uses the parameter $\theta$ to estimate the state-action values $Q(S_t, A_t, \theta)$ for a given state-action pair $(S_t, A_t)$, where $S_t$ represents the network state at time $t$, and $A_t$ represents the action taken by the agent at time $t$. The other network, called the $Q$-Target-Network, employs the parameter $\theta^-$ to determine the target $Q$-value. Algorithm 1 illustrates the DeepDefrag training and operation, which combines DQN training with proactive SD. In this algorithm, $M$ represents the number of episodes, $T$ denotes the length of each episode, $\gamma$ represents the discount factor, $\epsilon$ represents the exploration rate, and $C$ signifies the frequency of updating the target network. A detailed description of all the hyperparameters can be found in the original DQN paper [36]. The algorithm begins with the initialization step (lines 1-3). Then, for each episode of the training process, the environment is reset, and the loop for time steps begins (lines 4-6). During the training process, the agent adopts the $\epsilon$-greedy policy to balance exploration and exploitation. This means that the agent selects the action with the maximum $Q$-value with a probability of 1-$\epsilon$, and chooses a random action with a probability of $\epsilon$ (line 7). If the agent decides to perform a reallocation, it moves the connection $D_i$ to the starting slot related to the selected action $f_{ij}$ (lines 8-9). Otherwise, it continues the network operation while observing the reward and the next state (lines 10-11). The agent stores the transition samples in the replay memory for training purposes (line 12). The training step takes place at the end of each episode. Samples are randomly selected from the replay memory to train the NN (line 13). The target values for each transition in the mini-batch are calculated (line 14). If the next state is terminal, the target value is set to the immediate reward $r_i$. Otherwise, it is calculated as the sum of the immediate reward $r_i$ and the discounted maximum expected reward. The NN parameter $\theta$ is updated using Mini-batch Gradient Descent (line 15), while the $Q$-Target-Network parameter $\theta^-$ is updated with the current $Q$-Value-Network parameter $\theta$ every $C$ iterations

**Algorithm 1.** DeepDefrag Algorithm: Combination of DQN algorithm and proactive spectrum defragmentation.

1: Initialize replay memory $D$ with size $N$
2: Create action-value function $Q$ with random weights $\theta$
3: Create target action-value function $\hat{Q}$ with weights $\theta^- = \theta$
4: **for** episode $= 1$ to $M$ **do**
5: Reset environment to initial state $s_0$
6: **for** time step $t = 1$ to $T$ **do**
7: Choose action $a_t$ using $\epsilon$-greedy policy based on $Q$
8: **if** $a_t$ is $<D_i, f_{ij}>$ **then**
9: Reallocate $D_i$ to slot $f_{ij}$ , observe reward $r_t$ and next state $s_{t+1}$
10: **else if** $a_t$ is $\emptyset$ **then**
11: Serve the next incoming service request, observe reward $r_t$ and next state $s_{t+1}$
12: Store transition $(s_t, a_t, r_t, s_{t+1})$ in $D$
13: Randomly sample a minibatch of transitions $(s_i, a_i, r_i, s_{i+1})$ from $D$
14: $y_i = \begin{cases} r_i & \text{if } t = T-1 \\ r_i + \gamma \max_{a'} \hat{Q}(s_{i+1}, a', \theta^-) & \text{otherwise} \end{cases}$
15: Perform gradient descent on loss $\mathcal{L}(\theta) = \frac{1}{B}\sum_i (y_i - Q(s_i, a_i, \theta))^2$
16: Every $C$ steps, update target network weights: $\theta^- = \theta$

The training phase of the agent (lines 13–16 of Alg. 1) can be executed offline, meaning that it will not interfere with the network operation. In the predicting phase of the DQN (line 7 of Alg. 1), the trained model is utilized to predict the action for a given state. This phase is composed only of a simple DNN inference. Consequently, the time required for performing an inference becomes negligible compared to other events taking place in the network. Ideally, new experiences collected during operation are included in the memory and used to further improve the agent.

## 5. SIMULATION SETTINGS

We conduct simulations on a dynamic traffic scenario to evaluate the performance of DeepDefrag. We use the value of SBR, frequency, and volume of reconfiguration actions as performance metrics. Two network topologies are used to evaluate the DeepDefrag model: the NSFNET topology [13] with 14 nodes and 22 links, and the German topology [37] with 50 nodes and 88 links. In both topologies, we assume that each link supports 320 spectrum slots. To generate service requests, we use a Poisson process and tune the traffic load to 80 and 340 Erlang for the NSFNET and German topology, respectively. These values ensure a SBR of approximately 2% for the scenario without SD. 80% of service requests are long-lived with an average holding time of 25 time units, while the remaining 20% have an average

holding time of 12.5 time units. The holding time of the connections follows an exponential distribution. The considered bit rate is 100 Gbit/s for 50%, 200 Gbit/s for 30%, and 400 Gbit/s for the remaining 20% of the service requests. The choice of having an 80-20 split between long short-lived traffic aims at recreating a realistic traffic scenario experienced by a network operator in the Nordic countries where, within the optical layer, there exist connections that support the packet network and carry the majority of the traffic load. These connections are typically bound by long-term contracts and demonstrate consistent and stable behavior at the optical layer within the network. We adopt BPSK, QPSK, 8-QAM, and 16-QAM modulation formats, with a spectral efficiency $m$ of 1, 2, 3, and 4 b/Hz/s, respectively, as described in [13]. Modulation formats with higher spectrum efficiency are preferred, as long as the distance of the path is supported by the chosen modulation format. Specifically, the reach for the different modulation formats are as follows: 625 km for 16-QAM, 1250 km for 8-QAM, and 2000 km for QPSK. BPSK can be used for any path length in the adopted topologies [38]. For each request for all considered scenarios, the RMSA solution is obtained by choosing the shortest available path among five pre-computed shortest paths, and assigning the first available slots (first fit).

The performance of DeepDefrag is assessed through comparison with three heuristic algorithms denoted as older-first first-fit (OF-FF), exhaustive spectrum defragmentation (X-SD), and no spectrum defragmentation (No-SD). In the OF-FF strategy, the connections are selected according to their age, where the longest-running connections are reconfigured first, and the new spectrum is decided using the first-fit spectrum allocation scheme. This strategy is used for benchmarking purposes as it has shown excellent performance in terms of SBR [9]. OF-FF has two parameters: the SD period, which defines the number of request arrivals between two defragmentation cycles, and the number of connection reallocations per SD cycle. We evaluate the performance of OF-FF under different configurations and report on two representative settings to enable a fair comparison with DeepDefrag. The first setting has the same defragmentation overhead as DeepDefrag, enabling us to compare their performance in terms of SBR. In the second configuration, we ensure that the OF-FF has comparable levels of SBR as DeepDefrag. This enables a direct comparison of their performance in terms of SD overhead, namely the number of connection reallocations and SD cycles.

We also simulate the X-SD approach to find a (heuristic) lower bound on the SBR by reallocating an unlimited number of connections upon each connection departure and applying FF spectrum assignment to find the new slots. Note that the service blocking in this strategy occurs due to lack of resources, which cannot be avoided by any proactive defragmentation scheme. Moreover, achieving the absolute minimum SBR would require the use of optimal techniques (e.g., ILP). However, these techniques are not practical for this problem due to their complexity and scalability issues. Finally, the *No-SD* approach represents the network performance without defragmentation.

To implement the DeepDefrag scheme, we extended the Optical RL-Gym framework, which models optical networking problems related to resource management and reconfiguration as RL environments [39]. The DRL agent was trained using Stable-Baselines3 [40], an open-source implementation of DRL algorithms in Python. We trained the DRL agent using the DQN algorithm with a learning rate of $5 * 10^{-6}$, exploration rate 0.2, and a discount factor of 0.96. The NN has 5 layers with 384 neurons each. The values of DRL hyperparameters were defined through a hyperparameter analysis performed offline. Ten possible options for the oldest connections are introduced to the DRL agent.

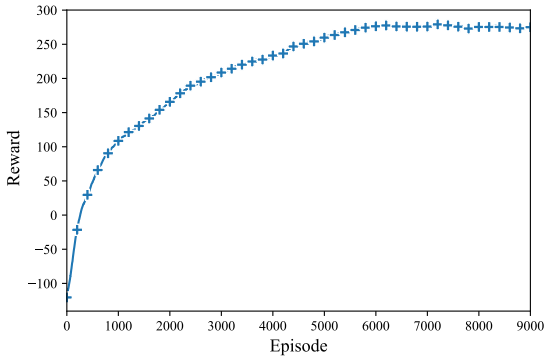



**Fig. 4.** Episodic sum of reward values for NSFNET.

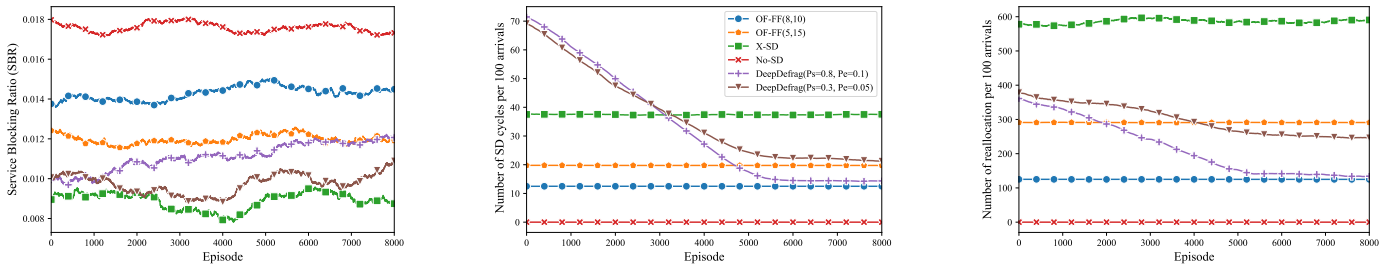
To assess the impact of defragmentation penalties on the performance of DeepDefrag, we conduct experiments using two sets of penalty factors. The first one consists of $Ps$=0.8 and $Pe$=0.1, while the second set has $Ps$=0.3 and $Pe$=0.05. In both sets, the value of $Ps$ is higher than $Pe$, reflecting the higher cost associated with initiating an SD cycle compared to a connection reallocation. The use of these two penalty sets allows us to understand the impact of SD penalties on the behavior of DeepDefrag, showing that network operators can fine-tune the penalty values based on their specific requirements, costs, and priorities.

Finally, we assess the performance of the proposed DeepDefrag approach under varying traffic load. To this end, we initiate the network operation with a load of 80 Erlang for the NSFNET topology. Subsequently, we change the load to new values: a higher load of 90 Erlang, and a lower load of 70 Erlang. This allows us to assess the agent's ability to adjust and converge to effective solutions under changing load conditions.

To train the agent, we set the episode length to 400 decision steps and perform training over 8000 episodes, which includes approximately 2 million service arrivals. It is important to note that fluctuations in the results are expected due to the inherent stochastic nature of the Poisson process. Hence, we conduct simulations of the DeepDefrag agent using 10 different seeds for the random number generator of the network environment to ensure robustness of the numerical results. We assess the performance of the DeepDefrag agent as it is trained and average the results over the last 1000 episodes for statistical purposes, followed by a calculation of the confidence interval to quantify the level of uncertainty in the results.

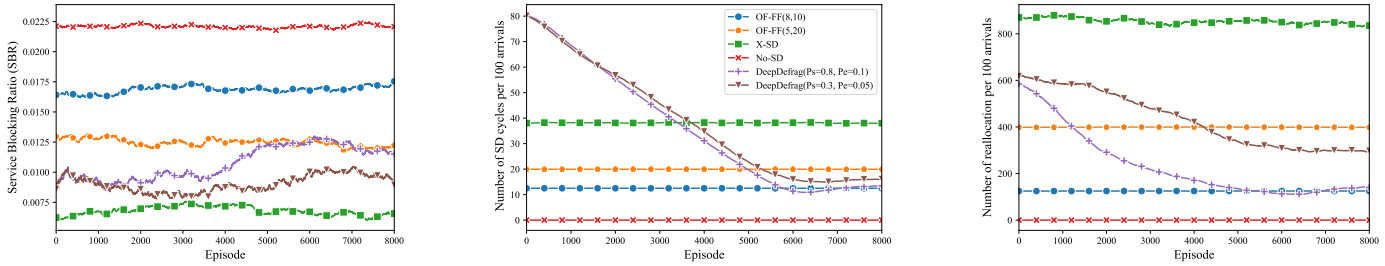
## 6. NUMERICAL RESULTS

Figure 4 depicts the progression of the episodic sum of reward values for DeepDefrag with the penalty set (0.8, 0.1) in the NSFNET topology. The plot shows the sum of the rewards of all actions taken within an episode. The result demonstrates how DeepDefrag optimizes its policy over time, leading to higher reward values. Eventually, around episode 6,000, the agent converges to a stable value. Naturally, as discussed later in this section, in normal operating conditions, the agent will continue to be trained in order to reflect the latest network conditions.




**(a)** Service blocking ratio (SBR) **(b)** Number of SD cycles per 100 arrivals **(c)** Number of reallocations per 100 arrivals

**Fig. 5.** Performance of the considered spectrum defragmentation schemes for the NSFNET network topology.




**(a)** Service blocking ratio (SBR) **(b)** Number of SD cycles per 100 arrivals **(c)** Number of reallocations per 100 arrivals

**Fig. 6.** Performance of the considered spectrum defragmentation schemes for the German network topology.

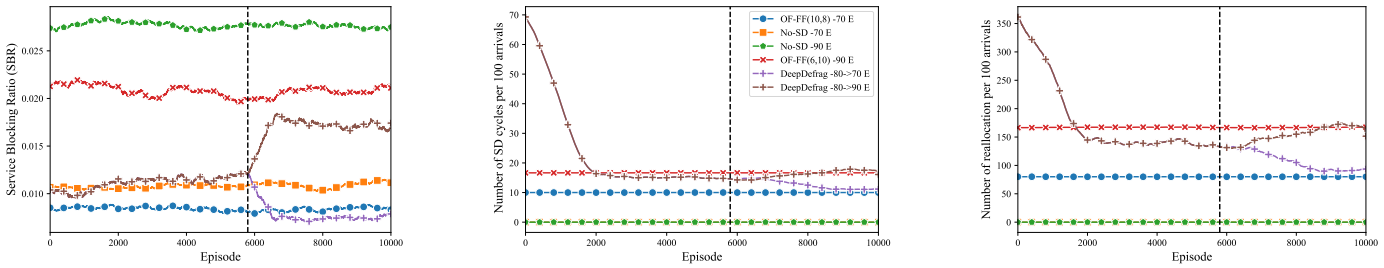
Figure 5 shows the performance of the considered schemes for the NSFNET topology, indicating the advantages of DeepDefrag. As shown in Fig. 5a, the two approaches performing the best and the worst in terms of the SBR are *X-SD* and *No-SD*, respectively. *X-SD* achieves 49% lower SBR than *No-SD*, which indicates the potential gain that can be achieved by sequential proactive SD algorithms. Figures 5b and 5c depict the number of SD cycles and connection reallocations per 100 arrivals for the different strategies.

Upon convergence of the DRL agent, DeepDefrag leads to a notable SBR reduction compared to the *No-SD* scenario. With the penalty set (0.8, 0.1), DeepDefrag achieves a 32% lower SBR than *No-SD*. For the penalty set (0.3, 0.05), DeepDefrag reduces the SBR by 38.6%. The confidence interval of the results for DeepDefrag is 1.6% with a 95% confidence level. For the sake of simplicity, we select penalty configuration (0.8, 0.1) for the rest of the paper. Two different configurations are evaluated for OF-FF. The first configuration is denoted by OF-FF (5, 15), with the SD period equal to 5 connection departures, and allowing up to 15 connection reallocations per SD cycle. OF-FF (5, 15) achieves approximately the same SBR as DeepDefrag in the NSFNET topology, allowing for a comparison of their defragmentation overheads. The second configuration is denoted by OF-FF (8, 10), with the SD period equal to 8 request departures and 10 reallocations per cycle. This results in almost the same defragmentation overhead as DeepDefrag, enabling an examination of their SBR. On average, the OF-FF (8, 10) and OF-FF (5, 15) schemes yield a 20.2% and 29.4% lower SBR than *No-SD*, respectively, which aligns with the result reported by [9]. As shown in these figures, DeepDefrag has almost the same defragmentation overhead as OF-FF (8, 10), while it reduces SBR by 15.8%. The X-SD achieves 23.3% lower SBR than and DeepDefrag, but at the cost of a higher defragmentation overhead. This confirms the effectiveness of DeepDefrag in reducing the SBR by selecting appropriate actions. Next, we move our attention to the configuration when DeepDefrag and OF-FF have close SBR performance, i.e., OF-FF (5, 15). DeepDefrag triggers 14.1 SD cycles per 100 arrivals on average as depicted in Fig. 5b. This is 29.5% lower than the number of SD cycles triggered by OF-FF. As shown in Fig. 5c, DeepDefrag reallocates 132 connections per 100 request arrivals on average, which is a 56% reduction compared to OF-FF (5, 15).

The observed results illustrate how, during the training phase, the lower SBR values can be attributed to the agent's frequent execution of SD cycles and reallocation of a significant number of connections. As the agent progresses and learns to make better decisions, it finds a beneficial trade-off between the SBR and extensive reallocation, i.e., reduces the number of SD cycles and connection reallocations, while slightly increasing the SBR.

When comparing the two penalty sets, DeepDefrag with the (0.3, 0.05) configuration achieves a 10.2% lower SBR than the (0.8, 0.1) configuration. This advantage comes at the expense of a 44.2% higher number of connection reallocations and a 30% higher number of SD cycles. These results highlight the trade-offs involved in selecting penalty values for DeepDefrag. By adjusting the penalties, operators can effectively balance the reduction in SBR with the associated costs of connection reallocations and SD cycles.

Figure 6 depicts the SD performance when the considered schemes are applied in the German topology. Also for this topology, DeepDefrag, after convergence, outperforms all the benchmark SD heuristics. In this case, X-SD reduces the SBR by 69.5% compared to *No-SD* (Fig. 6a). DeepDefrag with the penalty set (0.8, 0.1) achieves 50% lower SBR than *No-SD*. Moreover, it decreases the SBR by 34.8% in comparison with OF-FF (8, 10), which has an equivalent defragmentation overhead to DeepDefrag. In addition, DeepDefrag has comparable SBR as OF-FF (5, 20), while reducing the number of SD cycles and connection




**(a)** Service blocking ratio (SBR) **(b)** Number of SD cycles per 100 arrivals **(c)** Number of reallocations per 100 arrivals

**Fig. 7.** Performance of DeepDefrag for the NSFNET network topology with changing load conditions. The black dashed vertical line indicates the moment when the load changes.

reallocations by 34.1% and 75%, respectively (Figs. 6b and 6c). Similar trends for the different sets of DeepDefrag penalties are observed as in the case of the NSFNET topology, trading-off the frequency and volume of reallocations for the SBR. Examining the learning aspects depicted in the figures indicates the ability of DeepDefrag to reduce the SD overhead in terms of connection reallocations and defragmentation cycles upon 5,500 and 6000 training episodes for the NSFNET and the German topology, respectively.

The gap between *X-SD* and *No-SD* in terms of SBR is 49% and 69.5% for the German and the NSFNET topology, respectively, indicating a more prominent effect of SF in the German network under the considered traffic scenario. Hence, the ability of DeepDefrag to select appropriate actions becomes more substantial, resulting in a better overall performance in the German topology compared to the NSFNET. In summary, DeepDefrag outperforms the considered SD heuristic algorithms evaluated across all of the examined metrics. It also achieves an acceptable performance in terms of SBR compared to X-SD.

Figure 7 illustrates the results for the scenario with changing load conditions in the NSFNET topology. The agent is initially trained when the network is experiencing a load of 80 Erlang. Around episode number 5,800, indicated by the black dashed vertical line, the load changes to 90 Erlang and 70 Erlang, respectively. The results demonstrate that the agent successfully adapts to both an increase and a decrease of the traffic load. To ensure a fair comparison, in this case we report the results for the configurations of the OF-FF scheme that have equivalent SD overhead as DeepDefrag. For the load of 90 Erlang, DeepDefrag outperforms the No-SD scheme by 36.7% in terms of SBR. Additionally, it exhibits an 18.2% improvement over the OF-FF (6,10) configuration. Similarly, for the load of 70 Erlang, DeepDefrag demonstrates a 28.9% performance advantage over the No-SD scheme, and a 9.5% improvement compared to the OF-FF (10,8) configuration. These findings highlight DeepDefrag's ability to adapt to acceptable solutions across varying load levels, demonstrating its effectiveness in managing spectrum resources throughout the network lifetime under different operating conditions.

## 7. CONCLUSION

In this paper, we propose a deep reinforcement learning (DRL)-based framework called DeepDefrag. The framework jointly addresses different aspects of the spectrum defragmentation (SD) problem. DeepDefrag determines when to perform SD, which connections to reallocate and in what order, and which target spectrum slots to be utilized by the reconfigured connections. DeepDefrag considers spectrum occupancy information, including three fragmentation metrics (i.e., number of cuts, Shannon entropy (SE), and root of sum of squaress (RSSs)), as input to the decision process. Simulation results show that DeepDefrag can effectively reduce the service blocking ratio (SBR) while requiring fewer SD cycles and connection reallocations compared to heuristic methods from the literature. In some cases, the SBR achieved by DeepDefrag approaches that of an exhaustive method, while incurring substantially lower overhead. Finally, simulations with varying load conditions demonstrate that DeepDefrag is able to effectively adjust to changing network conditions.

## FUNDING

Work partially supported by Sweden's innovation agency VINNOVA, within the framework of the EUREKA cluster CELTIC-NEXT project AI-NET-PROTECT (2020-03506).

## REFERENCES

1. M. Jinno, H. Takara, B. Kozicki, Y. Tsukishima, Y. Sone, and S. Matsuoka, "Spectrum-efficient and scalable elastic optical path network: architecture, benefits, and enabling technologies," IEEE Commun. Mag. **47**, 66–73 (2009). DOI: 10.1109/MCOM.2009.5307468.
2. R. Wang and B. Mukherjee, "Provisioning in elastic optical networks with non-disruptive defragmentation," J. Light. Technol. **31**, 2491–2500 (2013).
3. Y. Yin, H. Zhang, M. Zhang, M. Xia, Z. Zhu, S. Dahlfort, and S. B. Yoo, "Spectral and spatial 2D fragmentation-aware routing and spectrum assignment algorithms in elastic optical networks," J. Opt. Commun. Netw. **5**, A100–A106 (2013). DOI: 10.1109/LCOMM.2021.3053279.
4. W. Shi, Z. Zhu, M. Zhang, and N. Ansari, "On the effect of bandwidth fragmentation on blocking probability in elastic optical networks," IEEE Transactions on Commun. **61**, 2970–2978 (2013).
5. B. C. Chatterjee, S. Ba, and E. Oki, "Fragmentation problems and management approaches in elastic optical networks: A survey," IEEE Commun. Surv. & Tutorials **20**, 183–210 (2017).
6. S. Ba, B. C. Chatterjee, and E. Oki, "Defragmentation scheme based on exchanging primary and backup paths in 1+1 path protected elastic optical networks," IEEE/ACM Transactions on Netw. **25**, 1717–1731 (2017).
7. M. Zhang, Y. Yin, R. Proietti, Z. Zhu, and S. J. B. Yoo, "Spectrum defragmentation algorithms for elastic optical networks using hitless spectrum retuning techniques," in *Optical Fiber Communication Conference and Exposition and the National Fiber Optic Engineers Conference (OFC/NFOEC)*, (2013), pp. 1–3.
8. Y. Aoki, X. Wang, P. Palacharla, K. Sone, S. Oda, T. Hoshida, M. Sekiya, and J. C. Rasmussen, "Dynamic and flexible photonic node architecture with shared universal transceivers supporting hitless defragmentation,"

in *European Conference and Exhibition on Optical Communications (ECOC),* (2012), pp. 1–3.
9. J. Comellas, L. Vicario, and G. Junyent, "Proactive defragmentation in elastic optical networks under dynamic load conditions," Photonic Netw. Commun. **36**, 26–34 (2018).
10. M. Zhang, C. You, and Z. Zhu, "On the parallelization of spectrum defragmentation reconfigurations in elastic optical networks," IEEE/ACM Transactions on Netw. **24**, 2819–2833 (2016).
11. D. Moniz, A. Eira, A. de Sousa, and J. Pires, "On the comparative efficiency of non-disruptive defragmentation techniques in flexible-grid optical networks," Opt. Switch. Netw. **25**, 149–159 (2017).
12. M. Shimoda and T. Tanaka, "Mask RSA: End-to-end reinforcement learning-based routing and spectrum assignment in elastic optical networks," in *European Conference on Optical Communication (ECOC),* (2021), p. Th1E.4.
13. 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," J. Light. Technol. **37**, 4155–4163 (2019).
14. M. R. Raza, C. Natalino, P. Öhlen, L. Wosinska, and P. Monti, "Reinforcement learning for slicing in a 5G flexible RAN," J. Light. Technol. **37**, 5161–5169 (2019).
15. E. Etezadi, C. Natalino, R. Diaz, A. Lindgren, S. Melin, L. Wosinska, P. Monti, and M. Furdek, "DeepDefrag: A deep reinforcement learning framework for spectrum defragmentation," in *IEEE Global Communications Conference (GLOBECOM),* (2022), pp. 3694–3699.
16. X. Wang, Q. Zhang, I. Kim, P. Palacharla, and M. Sekiya, "Utilization entropy for assessing resource fragmentation in optical networks," in *Optical Fiber Communication Conference (OFC),* (2012), pp. OTh1A–2.
17. D. Amar, E. Le Rouzic, N. Brochier, J.-L. Auge, C. Lepers, and N. Perrot, "Spectrum fragmentation issue in flexible optical networks: analysis and good practices," Photonic Netw. Commun. **29**, 230–243 (2015).
18. X. Yu, J. Zhang, Y. Zhao, T. Peng, D. Wang, and X. Lin, "Spectrum compactness based defragmentation in flexible bandwidth optical networks," in *National Fiber Optic Engineers Conference,* (2012), p. JTh2A.35.
19. Y. Takita, K. Tajima, T. Hashiguchi, and T. Katagiri, "Wavelength defragmentation with minimum optical path disruptions for seamless service migration," in *Optical Fiber Communications Conference and Exhibition (OFC),* (2016), p. M2J.3.
20. P. Lechowicz, M. Tornatore, A. Włodarczyk, and K. Walkowiak, "Fragmentation metrics in spectrally-spatially flexible optical networks," in *International Conference on Optical Network Design and Modeling (ONDM),* (Springer, 2020), pp. 235–247.
21. Y. Yin, M. Zhang, Z. Zhu, and S. B. Yoo, "Fragmentation-aware routing, modulation and spectrum assignment algorithms in elastic optical networks," in *Optical Fiber Communication Conference and Exposition and the National Fiber Optic Engineers Conference (OFC/NFOEC),* (2013), p. OW3A.5.
22. P. Wright, M. C. Parker, and A. Lord, "Minimum- and maximum-entropy routing and spectrum assignment for flexgrid elastic optical networking [invited]," J. Opt. Commun. Netw. **7**, A66–A72 (2015).
23. N. T. Khai, R. Romero Reyes, and T. Bauschert, "Spectrum defragmentation with improved lightpath migration scheme in flex-grid networks," in *International Conference on Optical Network Design and Modeling (ONDM),* (2021), pp. 1–6.
24. M. Zhang, Y. Yin, R. Proietti, Z. Zhu, and S. B. Yoo, "Spectrum defragmentation algorithms for elastic optical networks using hitless spectrum retuning techniques," in *Optical Fiber Communication Conference (OFC),* (2013), pp. OW3A–4.
25. S. Fernández-Martínez, B. Baran, and D. P. Pinto-Roa, "Spectrum defragmentation algorithms in elastic optical networks," Opt. Switch. Netw. **34**, 10–22 (2019).
26. S. K. Singh and A. Jukan, "Efficient spectrum defragmentation with holding-time awareness in elastic optical networks," J. Opt. Commun. Netw. **9**, B78–B89 (2017).
27. S. Selvakumar and S. Manivannan, "A spectrum defragmentation algorithm using jellyfish optimization technique in elastic optical network (eon)," Wirel. Pers. Commun. pp. 1–19 (2021).
28. S. Trindade and N. L. da Fonseca, "Machine learning for spectrum defragmentation in space-division multiplexing elastic optical networks," IEEE Netw. **35**, 326–332 (2021).
29. Y. Xiong, Y. Yang, Y. Ye, and G. N. Rouskas, "A machine learning approach to mitigating fragmentation and crosstalk in space division multiplexing elastic optical networks," Opt. Fiber Technol. **50**, 99–107 (2019).
30. R. K. Jana, B. C. Chatterjee, A. P. Singh, A. Srivastava, B. Mukherjee, A. Lord, and A. Mitra, "Machine learning-assisted nonlinear-impairment-aware proactive defragmentation for c+l band elastic optical networks," J. Opt. Commun. Netw. **14**, 56–68 (2022).
31. R. Romero Reyes and T. Bauschert, "Towards DRL-based routing and spectrum assignment in optical networks: Lessons to be learned from Markov decision processes," in *IEEE Latin-American Conference on Communications (LATINCOM),* (2021), pp. 1–6.
32. N. D. Cicco, E. F. Mercan, O. Karandin, O. Ayoub, S. Troia, F. Musumeci, and M. Tornatore, "On deep reinforcement learning for static routing and wavelength assignment," IEEE J. Sel. Top. Quantum Electron. **28**, 1–12 (2022).
33. N. E. D. El Sheikh, E. Paz, J. Pinto, and A. Beghelli, "Multi-band provisioning in dynamic elastic optical networks: a comparative study of a heuristic and a deep reinforcement learning approach," in *International Conference on Optical Network Design and Modeling (ONDM),* (IEEE, 2021), pp. 1–3.
34. X. Luo, C. Shi, L. Wang, X. Chen, Y. Li, and T. Yang, "Leveraging double-agent-based deep reinforcement learning to global optimization of elastic optical networks with enhanced survivability," Opt. express **27**, 7896–7911 (2019).
35. R. Li, R. Gu, W. Jin, and Y. Ji, "Learning-based cognitive hitless spectrum defragmentation for dynamic provisioning in elastic optical networks," IEEE Commun. Lett. **25**, 1600–1604 (2021).
36. V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," arXiv preprint arXiv:1312.5602 (2013).
37. R. Carpa, M. D. de ASSUNÇÃO, O. Glück, L. Lefèvre, and J.-C. Mignot, "Responsive algorithms for handling load surges and switching links on in green networks," in *IEEE International Conference on Communications (ICC),* (IEEE, 2016), pp. 1–7.
38. B. Kozicki, H. Takara, Y. Sone, A. Watanabe, and M. Jinno, "Distance-adaptive spectrum allocation in elastic optical path network (slice) with bit per symbol adjustment," in *Conference on Optical Fiber Communication (OFC/NFOEC), collocated National Fiber Optic Engineers Conference,* (2010), pp. 1–3.
39. C. Natalino and P. Monti, "The Optical RL-Gym: An open-source toolkit for applying reinforcement learning in optical networks," in *International Conference on Transparent Optical Networks (ICTON),* (2020).
40. A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-Baselines3: Reliable reinforcement learning implementations," J. Mach. Learn. Res. **22**, 1–8 (2021).