DRL-Assisted Reoptimization of Network Slice Embedding on EON-enabled Transport Networks

Seyed Soheil Johari*, Sepehr Taeb*, Nashid Shahriar†, Shihabur R. Chowdhury*, Massimo Tornatore‡, Raouf Boutaba*, Jeebak Mitra§, and Mahdi Hemmati§
*David R. Cheriton School of Computer Science, University of Waterloo,
{ssjohari | staeb | sr2chowdhury | rboutaba}@uwaterloo.ca
†Department of Computer Science, University of Regina, Nashid.Shahriar@uregina.ca
‡Politecnico di Milano, massimo.tornatore@polimi.it
§Huawei Technologies Canada Research Center,
{jeebak.mitra | mahdi.hemmati}@huawei.com

***Abstract*—5G transport networks will support dynamic services with diverse requirements through network slicing. Elastic Optical Networks (EONs) facilitate transport network slicing by flexible spectrum allocation and tuning of transmission configurations. A major challenge in supporting dynamic services is the lack of priori knowledge of future slice requests. As a consequence, slice embedding can become sub-optimal over time, leading to spectrum fragmentation and skewed utilization. This in turn can block future slice requests, impacting operator revenue. To address this issue, operators can periodically re-optimize slice embedding for reducing fragmentation. In this paper, we address this problem of re-optimizing network slice embedding on EONs for minimizing fragmentation. The problem is solved in its splittable version, which significantly increases problem complexity, but also offers more opportunities for a larger set of re-configuration actions. We employ simulated annealing for systematically exploring the large solution space. We also propose a greedy algorithm to address the practical constraint of limiting the number of re-configuration steps. Moreover, we present a novel method based on Deep Reinforcement Learning (DRL) for determining when performing re-configuration is most effective. Our extensive simulations demonstrate that the greedy algorithm yields a solution very close to that obtained using simulated annealing while requiring orders of magnitude lesser re-configuration actions. Finally, we show that by applying the greedy algorithm periodically on the network according to the DRL-based time selection algorithm, a significant improvement in the total number of accepted slice requests can be achieved with only performing a limited number of re-configuration operations.**

***Index Terms*—Elastic Optical Network, Fragmentation, Deep Reinforcement Learning (DRL), Transport Network**

I. Introduction

Transport networks are evolving to support dynamic and short-lived services with diverse quality of service (QoS) requirements [1]–[3], such as enhanced mobile broadband and ultra-reliable low-latency communication [4] through 5G network slicing [5]. This evolution towards network slicing is being fueled by recent advances in Elastic Optical Network (EON) virtualization [5], [6], as EONs support finer-grained spectrum allocation and tuning of transmission configurations (such as modulation format, forward error correction (FEC) overhead, and baudrate) that allow to rightsize EON resource allocation to network slices [7], [8]. Hence, EON-enabled transport networks can leverage their flexible resource allocation capabilities to offer to customers tailored and dynamic network slices, typically in terms of a virtual networks (VN) consisting of virtual nodes connected by virtual links.

A key challenge in supporting dynamic services with short lifetime through network slicing is how to deal with the lack of a priori knowledge of slice arrivals and departures. Without such knowledge, it has been observed that slice resource allocation becomes sub-optimal over time, and, in the specific case of EONs, underlying spectrum resources become fragmented and capacity utilization becomes skewed [9], [10]. Such imbalanced and fragmented spectrum utilization can result in blocking future VN requests [9]. For these reasons, it is essential to re-optimize resources allocated to VNs to minimize fragmentation, ensuring optimal resource utilization, and making room for future VN requests.

In this paper, we address the problem of re-optimizing network slice or VN embedding on an EON. VN embedding on EON involves allocating spectrum resources to a VN by mapping the virtual nodes and virtual links on EON nodes and paths with appropriate transmission configuration [7]. Among the possible objectives, we focus on reducing spectrum fragmentation through re-optimization of VN embedding. Defragmentation is considered crucial for network operators and has attracted remarkable attention in research [9]. Since minimizing fragmentation alone may lead to unbounded increase in spectrum usage, we measure to limit additional spectrum usage during defragmentation, which adds additional challenge to the re-optimization problem. While solving the network slice embedding re-optimization problem, we only consider modifying virtual link embedding, as virtual nodes of a network slice typically have location constraints (e.g., they are mapped to an EON node equipped with compute resources located at a metro data center (MDC) or a Point-of-Presence (PoP) site).

Moreover, re-optimization of VN embedding needs to be invoked multiple times during network operation, and determining the most appropriate times for invoking the re-optimization is necessary for decreasing VN blocking with minimal operational complexity. In other words, not performing re-optimization can result in severe VN blocking, whereas invoking re-optimization too often will cause unnecessary operational complexity. We propose to utilize Deep Reinforcement Learning (DRL) for selecting the times for performing re-optimization. DRL has shown to be very successful for maximizing both current and future rewards simultaneously in many decision-making applications where there are no prior knowledge about the behavior of the environment [11], [12].

Even though Virtual Network Embedding (VNE) related problems have been thoroughly investigated in the past, especially in IP networks, investigation of the same problem in the optical layer is less extensive, and in particular, recent development of sliceable transponders [13] and splitting-enabled EON [14] to support massive bandwidth requests has increased the complexity of the problem [8]. This has motivated us to devise a new methodology to address the increased complexity of the VNE-related problems. So, we significantly differ from the research literature in optical network defragmentation on several aspects. First, we consider the possibility of splitting a virtual link demand on multiple continuous and contiguous spectrum allocations (called splits) over one or more substrate EON path(s) (similar to [8], [14]). Second, we do not assume the existence of any special technology (*e.g.*, push-pull [15] and hop-retuning [16]) for performing disruption free defragmentation. We assume the capabilities of commercial transponders and adopt *make-before-break* [17] approach whenever possible.

We quantify spectrum fragmentation using the root mean square fragmentation (RMSF) metric [18], which is non-linear. Due to the non-linearity of our objective function of RMSF, it becomes infeasible to use standard integer/mixed-integer programming modeling tools to obtain optimal solution even for very small problem instances. Therefore, we leverage Simulated Annealing (SA) search [19] to systematically explore a large solution space for obtaining an estimation of a loose lower bound of RMSF. One advantage of using SA is that the search process naturally yields the sequence of actions that must be taken to bring the network to the computed re-optimized state. However, it may arrive at the final defragmented state by applying a excessively large number of re-configuration actions on the network, which might be undesirable from a network operator's perspective. Hence, we also devise a greedy search heuristic that navigates through the solution space with a given budget on the number of possible actions. To the best of our knowledge, this aspect of defragmentation was not studied before in the context of splitting-enabled EON. Our extensive evaluation using real network topologies demonstrates that the greedy search process reduces network defragmentation nearly to the same extent as the SA search, while applying orders of magnitude lesser number of reconfiguration actions. For determining the best times to perform re-optimization, we train a Soft Q-Network model [20] as a DRL agent that in each considered time step, intelligently decides whether to invoke the re-optimization process or not.

This paper is an extended version of the work presented in [21]. With respect to a SA and a greedy algorithm in [21], we analyze the effect of applying re-optimization on increasing the EON's ability to accept more VN requests. Moreover, we propose a novel DRL-based algorithm for resolving the trade-off between increase in the number of accepted VNs and total number of performed re-optimization operations in an intelligent manner. Through extensive experiments, we demonstrate the superiority of our DRL-based method compared to other alternative strategies and also evaluate the generalization capability of the DRL method for VN request distributions that significantly differ from those seen during training. Finally, we expand our discussion of the related works and the experimental results.

The rest of the paper is organized as follows. We first discuss the related works in Section II. Then, we discuss our choice of fragmentation metric and re-configuration actions, followed by a formal problem definition in Section III. In Section IV, we present our SA algorithm for re-optimizing VN embedding over an EON. Our greedy search approach with a bounded number of re-configuration actions is presented in Section V, and the DRL-based time selection strategy for invoking the re-optimization operations is presented in Section VI. Evaluation results are presented in Section VII. We conclude with some future research directions in Section VIII.

II. Related Works

EON spectrum defragmentation strategies fall into two broad categories, namely, proactive and reactive [9]. Proactive approaches are executed periodically or when fragmentation level reaches a pre-defined threshold [22]. In contrast, reactive approaches defragment spectrum resources when the embedding of a new VN is blocked due to fragmentation and can make room for the newly arrived VN [23]. However, there is no guarantee that reactive defragmentation will be successful and it may result in sub-optimal embedding for the VNs that are migrated during the process. Another line of work for EON defragmentation is fragmentation-aware VN embedding [24], [25]. Fragmentation awareness is achieved by embedding new VNs in a way to reduce spectrum fragmentation. Fragmentation-aware embedding may not be sufficient to defragment an entire EON since such embedding only concerns fragmentation around the new VN. In contrast, proactive defragmentation accounts for all VNs and EON links, and offers more opportunities to re-optimize VN embeddings. Hence, in this paper, we focus on proactive defragmentation by changing the route and/or the spectrum allocation of connections. While doing so, existing approaches assume the availability of techniques, such as push-pull or hop re-tuning, that promises to reconfigure spectrum allocation without causing any traffic disruption [26], [27]. However, these technologies are still experimental and are not available in commercial equipment. Conversely, we do not assume any specific technology for eliminating disruption and adopt the make-before-break approach whenever possible [17].

Shakya *et al.*, proposed a defragmentation approach in [28] that minimizes the maximum used spectrum slot index on any EON link. This objective may not be useful since many fragmented spectrum blocks can exist on a link even when maximum slot index is low. Dávalos *et al.*, presented two metaheuristic-based algorithms for selecting lighpaths that need to be reconfigured for defragmentation [29]. This approach does not generate the order in which lightpaths should be re-configured that we address in this paper. Comellas *et* *al.*, [22] evaluated several ordering strategies for reconfiguring lightpaths for defragmentation. However, a deterministic order can get stuck at a local minima, mandating to include randomness in the deframengtation algorithm.

Zeng *et al.*, presented an SA-based defragmentation mechanism for EONs that modifies the route and/or spectrum assignment of a randomly chosen connection [30]. The cost function in [30] prefers a state where lightpaths have shorter lengths, and spectrum is allocated from a lower frequency. Conversely, we adopt a comprehensive cost function accounting for multiple factors such as number of fragments, size of fragments, and locations of fragments in the spectrum. We also employ several new re-configuration operations and additional optimizations not considered in [30]. One caveat of SA algorithm, both ours and that in [30], is that it requires a large number of steps to reach a re-optimized state, making it impractical. Therefore, we also propose an effective algorithm that reaches a defragmented state in a limited amount of steps, which makes our approach ready to be applied in practice.

There have been some works that try to form an analytical performance modeling of spectrum defragmentation in elastic optical links, such as [31], and [32]. The work in [31] presents an analytical study of the effect of defragmentation on the connections' blocking probability in an elastic optical link depending on whether defragmentation is proactively or reactively performed. The authors of [32] formulate a Markov Chain model for characterizing the fragmentation issue in a simplified two-service elastic optical link scenario. These works are limited to the case of a single fiber link, and extending their work to attain an analytic modeling of spectrum defragmentation in large scale EONs with splitting-enabled technology would be a very interesting but challenging research direction. Moreover, heuristics that require calculating a complex fragmentation metric (such as RMSF [18]) would significantly increase the complexity of the modeling. We consider such analytical modeling to be out of scope of this paper and leave it as a potential future work direction.

There are also several papers that analyze the effect of spectrum defragmentation on decreasing the blocking probability and improving the EON capacity for accepting more future traffic. Among them, [29] evaluates the effect of two proactive defragmentation algorithms based on Ant Colony optimization and Genetic metaheuristics on blocking probability over two different EON topologies. A survey paper [9] evaluates and analyzes state-of-the-art fragmentation management approaches in terms of blocking probability by implementing and applying those approaches in a unified simulation environment for fair comparison. As discussed before, an important question regarding proactive defragmentation is when to invoke the re-optimization process. However, the aforementioned papers simply trigger the re-optimization process pro-actively in a periodic manner, or when specific thresholds are exceeded (e.g., fragmentation is high). In contrast to these works, [33] proposed an intelligent algorithm for time selection of the re-optimization operations. The proposed algorithm periodically monitors the temporary blocking probability for the previous requests in a time window whose size is purposefully adjusted to cope with the changes in the traffic. However, as our experimental results will show, simply relying on a single metric for choosing the re-optimization times might lead to a very poor performance in terms of acceptance ratio.

III. Problem Definition

A. Problem Statement

We are given an EON G and a set of VNs $\mathcal{G}$ embedded on G. EON G (the substrate network) consists of a set of substrate optical nodes (SNodes) and substrate optical links (SLinks), respectively; and a SPath is a substrate path between two SNodes in the EON that can contain a sequence of SLinks. Each VN $\bar{G} \in \mathcal{G}$ consists of a set of virtual nodes (VNodes) $\bar{V}$ and virtual links (VLinks) $\bar{E}$ where each VLink $\bar{e} \in \bar{E}$ has a bandwidth demand $b_{\bar{e}}$. Each VN is embedded by mapping each of its VNodes to an EON node and each of its VLinks to a set of splits with a maximum of q splits. Fig. 1 shows an illustrative example of the embedding of a VLink over two splits. Each split represents a path in the EON where each path p is configured with a transmission configuration represented by a tuple $t = (d, b, m, f) \in \mathbb{T} = (\mathbb{D} \times \mathbb{B} \times \mathbb{M} \times \mathbb{F})$ to provide a data-rate so that the sum of data-rates is $b_{\bar{e}}$. Here, d, b, m, and f represent *data-rate*, *baud-rate*, *modulation format*, and *FEC* selected from the set of possible values $\mathbb{D}$, $\mathbb{B}$, $\mathbb{M}$, and $\mathbb{F}$, respectively. Each tuple t has a spectrum requirement and a maximum optical reach within which t can be used with satisfactory signal quality as defined by a reach table $\mathcal{R}$ [34]. The reach table $\mathcal{R}$ contains a set of tuples each of which has a specific spectrum requirement and a maximum optical reach. Note that the same SPath can be used multiple times as the splits of a VLink following the reasoning in [8]. Also note that $\chi_{\bar{e}i} = (p, t, s_b, s_t) | 1 \leq i \leq q$ represents the i-th split, where $\chi_{\bar{e}i}^{(p)}$ and $\chi_{\bar{e}i}^{(t)}$ denote the selected SPath and transmission configuration for the i-th split, respectively. The spectrum slot allocation for the i-th split begins at index $\chi_{\bar{e}i}^{(s_b)} \in S$ and ends at index $\chi_{\bar{e}i}^{(s_t)} \in S$ along each SLink in the SPath $\chi_{\bar{e}i}^{(p)}$ to satisfy the spectrum contiguity constraint [9].

The VN embedding re-optimization problem seeks to find a feasible sequence of re-configuration actions applied on the VN embeddings such that the resulting embeddings of the VNs optimize (*i.e.*, maximize or minimize) a certain objective function. In our problem, the objective is to minimize network-wide RMSF metric presented in (1). This metric leverages RMSF values of each EON link computed using (2), which is the most comprehensive fragmentation metric [18]. We assume VNode mapping to remain unchanged during the re-optimization process (typical assumption in optical network virtualization [35]).

$$F_{\text{net}}^{\text{RMSF}} = \frac{\sum_{e \in E} F_e^{\text{RMSF}}}{|E|} \frac{s^{\max}}{|S|} \tag{1}$$

$$F_e^{\text{RMSF}} = \frac{s_e^{\max} |I|}{\sqrt{\frac{\sum_{i \in I} f_i^2}{|I|}}} \tag{2}$$

During re-optimization, the embedding of a VLink of the existing VNs in $\mathcal{G}$ can be re-configured one or multiple times to reduce spectrum fragmentation. VLink re-configuration consists in changing the VLink's embedding *i.e.*, one or more

TABLE I: Different re-configuration actions and corresponding cost

ID	Re-optimization action	Disruption level	Extra Transponder?	Extra Spectrum?
R_1	Move a split on the same path with the optimal tuple in $\mathcal{R}$	Zero	No	No
R_2	Move a split to a different path with the optimal tuple in $\mathcal{R}$	Zero	No	No
R_3	Merge two splits of the same VLink into one such that spectrum allocation of the new split does not overlap with the spectrum allocations of previous two splits	Zero	No	No
R_4	Merge two splits of the same VLink into one such that spectrum allocation of the new split overlaps with the spectrum allocation of any of the previous two splits	High	No	No
R_5	Divide a split of a VLink into multiple splits on the same path	Zero	Yes	(Possibly) Yes

splits' SPath ($\chi^{(p)}_{\bar{e}i}$), tuple ($\chi^{(t)}_{\bar{e}i}$), and spectrum allocation ($\chi^{(s_b)}_{\bar{e}i}$ and/or $\chi^{(s_t)}_{\bar{e}i}$) or a combination thereof. A re-configured split still has to satisfy spectrum contiguity and optical reach constraints and total number of splits after re-optimization cannot be more than q. We use 5 different re-configuration actions with different levels of disruption and their impact on resource usage (*e.g.*, transponder or spectrum usage) shown in Table I. For instance, R_1 in Table I may change any or all of $\chi^{(t)}_{\bar{e}i}$, $\chi^{(s_b)}_{\bar{e}i}$, and $\chi^{(s_t)}_{\bar{e}i}$ of a split while keeping its $\chi^{(p)}_{\bar{e}i}$ fixed, whereas R_2 can change all of them. Splitting offers us new re-configuration actions (R_3, R_4, and R_5) that were not considered in prior work. For example, we can merge two splits into a larger one using R_3/R_4 or divide a large split into multiple smaller ones using R_5.

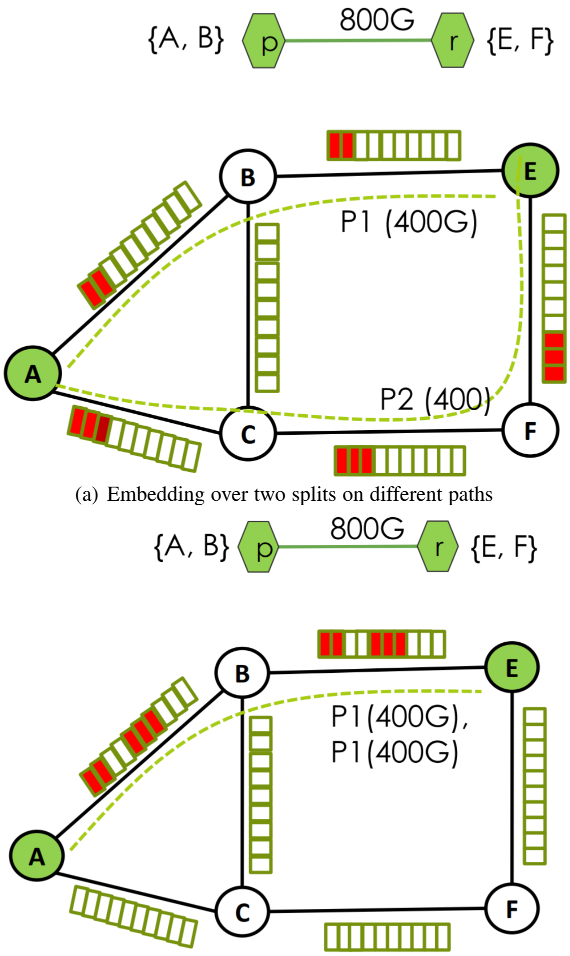
All actions except R_4 in Table I can be applied using *make-before-break* to avoid any disruption as long as the spectrum allocation after applying any of these actions does not overlap with the previous spectrum allocation. Note that *make-before-break* requires double committed resource for a short period of time when it is applied. There are scenarios when the spectrum allocation after re-configuration overlaps with the allocation present before applying the action such as for action R_4 in Table I. In such scenarios, it might not be possible to apply *make-before-break*, therefore, the disruption level can be significantly high. However, disruptive action such as R_4 may be useful to defragment a highly utilized EON and should be used only when necessary.

B. Pre-computations

For each VLink $\bar{e} \in \bar{E}$, we pre-compute $\mathcal{P}^k_{\bar{e}}$, a set of k shortest paths between each pair of SNodes that a VLink's end VNodes have been mapped to. For each SPath $p \in \mathcal{P}^k_{\bar{e}}$, we pre-compute the set of admissible transmission configurations, $\mathcal{T}_{\bar{e}p} \subset \mathcal{T}$, such that each configuration $t \in \mathcal{T}_{\bar{e}p}$ results in a reach $r_t \geq len(p)$ and has a data-rate $t^{(d)}$. $\mathcal{T}_{\bar{e}}$ contains all the distinct tuples suitable for $\bar{e}$ and is defined as $\bigcup_{\forall p \in \mathcal{P}^k_{\bar{e}}} \mathcal{T}_{\bar{e}p}$.

IV. Re-optimization with Unbounded Number of Actions

In this section, we present a Simulated Annealing [19] (SA) based algorithm for solving the VN embedding re-optimization problem. SA allows us to systematically search through a solution space, while maintaining a lineage of operations starting from the initial network state to reach a feasible re-optimized state. The advantage of SA is that it can avoid getting stuck at local minima by employing random moves, and converge to a global minimum if the SA process is run for a sufficiently long period of time.



(a) Embedding over two splits on different paths

(b) Embedding over two splits on the same path

Fig. 1: Embedding of a virtual link over two substrate splits. Red and white rectangles represent occupied and empty spectrum slots, respectively.

A. Simulated Annealing (SA) Algorithm

SA systematically explores the neighborhood of an initial set of embeddings of all VLinks in $\mathcal{G}$ (*i.e.*, $current\_state$) and keeps reducing the fragmentation of the EON G. A *neighbor* to the $current\_state$ is another set of embeddings where the embedding of only one VLink from the $current\_state$ is re-configured using one of the actions presented in Table I. SA then evaluates quality of the neighbor using our objective function of $F^{\text{RMSF}}_{\text{net}}$ in Eqn. (1) and moves to a neighbor that improves $F^{\text{RMSF}}_{\text{net}}$. It also probabilistically accepts a worse neighbor (a re-optimized embedding with a higher value of $F^{\text{RMSF}}_{\text{net}}$ than the $current\_state$) from the search neighborhood.

Typically, a temperature parameter (T) and energy function controls the probability of accepting a worse neighbor. Parameter T is set to a higher value during the initial iterations of the search, resulting in a higher probability of accepting a worse neighbor. A cooling schedule attenuates the temperature and eventually decreases the probability of accepting a worse neighbor towards the end of the SA search. By accepting worse solutions, SA tries to avoid being stuck at a local minimum. We run multiple iterations of the neighborhood exploration procedure for each temperature to cover a wider search space. To better explain the SA algorithm, we define the following:

- A neighborhood generation function, which generates a set of neighbors to the $current\_state$ and returns a neighbor according to a policy.
- An energy function, which determines the fitness of a neighbor. It regulates the probability of accepting or rejecting a neighbor during an iteration of SA.
- A cooling schedule, which determines how the temperature is attenuated during the search process.

1) Neighborhood Generation: We devise an algorithm (Algorithm 1) for generating a set of neighbors to the $current\_state$ and returning one of the neighbors. Algorithm 1 first chooses a random split Y from the uniformly distributed set of embeddings of the VLinks of $current\_state$. Then, Algorithm 1 generates all feasible neighbors, $all\_neighbors$, by applying each action from Table I on split Y. However, if the chosen random split belongs to a Vlink that does not allow service disruption according to its QoS requirement, then no neighbors from applying R_4 on the current state will be generated in this process. Moreover, note that the number of feasible neighbors for a single action can be large due to the many possible feasible spectrum re-allocation of a split Y. Since there can be a substantial number of feasible neighbors in $all\_neighbors$ and a significant portion of them are not helpful for re-optimization, Algorithm 1 adopts two heuristics for keeping the most promising neighbors in $neighbor\_pool$. First, Algorithm 1 excludes those neighbors from $all\_neighbors$ that use $\delta\%$ additional spectrum slots compared to that used in split Y's current embedding (line 4). We call δ as *Slot Usage Limit*. The rationale for doing so is that a network operator may not want to increase spectrum usage during re-optimization and may want to move to neighbors whose spectrum usage remains within a specified bound. Second, Algorithm 1 populates a sorted list of the remaining neighbors in $all\_neighbors$ in increasing order of the neighbors' $F_{\text{net}}^{\text{RMSF}}$ values ($sorted\_neighbors$) and adds only the first $\Gamma\%$ (called as *Neighborhood Limit*) neighbors from $sorted\_neighbors$ to $neighbor\_pool$ (line 6). Here, both δ and Γ parameters can be tuned based on network operator policies. Moreover, as action R_4 has the drawback of causing service disruption, to make sure that this action is only chosen when it leads to a significant decrease in the network's fragmentation metric, we introduce a new parameter Γ' called *Disruption Limit*. We only keep neighbors generated by action R_4 that lie in the first $\Gamma'\%$ of neighbors in $neighbor\_pool$ (which is also sorted based on the $F_{\text{net}}^{\text{RMSF}}$ values) and remove all the other neighbors generated by applying R_4 from $neighbor\_pool$. In our experiments, we observed that even setting $\Gamma' = 100\%$ led to choosing very limited number of action R_4 (see Fig. 6). However, if the operators desire to further limit the number of times that action R_4 is chosen in the defragmentation process, they can choose a lower value for Γ'.

Algorithm 1: Select-Neighbor

```
1  function Select-Neighbor($current\_state, \delta, \Gamma, \Gamma'$)
2      Choose a random split $Y$ from $current\_state$
3      $all\_neighbors$ ← Generate all feasible neighbor states
         by applying actions from Table I on split $Y$
4      Exclude neighbor states from $all\_neighbors$ that uses
         $\geq \delta\%$ additional slots compared to $current\_state$
5      $sorted\_neighbors$ ← Sort the neighbors in
         $all\_neighbor$ in increasing order of their RMSF
6      $neighbor\_pool$ ← Select the first $\Gamma\%$ neighbors from
         $sorted\_neighbors$
7      Remove neighbors generated by $R_4$ that don't lie in the
         first $\Gamma'\%$ of $neighbor\_pool$
8      $max\_RMSF$ ← Maximum value of RMSF among the
         neighbors in $neighbor\_pool$
9      for $neighbor \in neighbor\_pool$ do
10         $Gain_{neighbor} \leftarrow$
             $(RMSF_{neighbor} - max\_RMSF)^2$
11     $< neighbor, action\_spec > \leftarrow$ A neighbor from
         $neighbors\_pool$ with a probability proportional to
         $Gain_{neighbor}$ and corresponding action
12     return $< neighbor, action\_spec >$
```

After generating $neighbor\_pool$, Algorithm 1 finds the maximum value of $F_{\text{net}}^{\text{RMSF}}$ among the neighbors in $neighbor\_pool$. This value is then used to compute the relative RMSF gain ($Gain_{neighbor}$) of each of the other neighbors in $neighbor\_pool$ (line 8 – 9). $Gain_{neighbor}$ is defined as the square of the difference between the neighbor's $F_{\text{net}}^{\text{RMSF}}$ and the maximum value of $F_{\text{net}}^{\text{RMSF}}$. Finally, the algorithm returns a neighbor with the probability proportional to its RMSF gain and $action\_spec$, the action that generated the neighbor.

2) Energy Function and Cooling Schedule: We use our fragmentation metric $F_{\text{net}}^{\text{RMSF}}$ defined in Eqn. (1) as the energy function. During iteration k of SA search, the probability of moving to a neighbor is a function of energy and temperature [19]. This probability is defined as follows:

$$\mathcal{P}(Energy, T_k) = \begin{cases} 1 & \text{if } \Delta_{Energy} < 0 \\ e^{-\Delta_{Energy}/T_k} & \text{otherwise.} \end{cases}$$

Here, T_k is the temperature at the k-th iteration and $\Delta_{Energy} = F_{\text{net}}^{\text{RMSF}}(current\_state) - F_{\text{net}}^{\text{RMSF}}(new\_state)$, where $F_{\text{net}}^{\text{RMSF}}(current\_state)$ and $F_{\text{net}^{\text{RMSF}}(new\_state)}$ are energies of $current\_state$ and the neighbor returned by Algorithm 1, respectively. We use a linear cooling schedule [36] and set the temperature T at iteration $k+1$ as: $T_{k+1} = \rho * T_k$, where $0 < \rho < 1$ is the cooling rate (Line 21).

The SA algorithm, as outlined in Algorithm 2, takes as input an initial state, *i.e.*, current set of VN embedding ($\mathcal{C}$), maximum number of iterations to perform (it_{max}), the number of iterations to perform per temperature value (it_{temp}), an initial temperature (T_0) and the cooling rate (ρ). During each iteration for a particular temperature value, new_state

Algorithm 2: SA-Re-Optimize

```
1  function SA-Re-Optimize($\mathcal{C}$, $it_{max}$, $it_{temp}$, $T_0$, $\rho$)
2      $iterations \leftarrow 0$, $T \leftarrow T_0$
3      $action\_sequence \leftarrow best\_sequence \leftarrow \phi$
4      $best\_state \leftarrow current\_state \leftarrow \mathcal{C}$
5      while $iterations < it_{max}$ do
6          while $iterations_t < it_{temp}$ do
7              $current\_cost \leftarrow F^{\text{RMSF}}_{\text{net}}(current\_state)$
8              $< new\_state, action\_spec > \leftarrow$
                 Select-Neighbor $(current\_state, \delta, \Gamma)$
9              $new\_cost \leftarrow F^{\text{RMSF}}_{\text{net}}(new\_state)$
10             $\Delta_{energy} = current\_cost - new\_cost$
11             $p \leftarrow$ rand(0, 1)
12             if $\Delta_{energy} < 0$ or $p < e^{\Delta_{energy}/T}$ then
13                 $current\_state \leftarrow new\_state$
14                 $current\_cost \leftarrow new\_cost$
15                 $action\_sequence.append(action\_spec)$
16             if $current\_cost < best\_cost$ then
17                 $best\_cost \leftarrow current\_cost$
18                 $best\_state \leftarrow current\_state$
19                 $best\_sequence \leftarrow action\_sequence$
20             Increment $iterations_t$
21         $T = \rho * T$, Increment $iterations$
22     return $< best\_state, best\_sequence >$
```

and its $action\_spec$ are generated by invoking the Select-Neighbor procedure from Algorithm 1 (line 8). Based on the energy of the new_state and the current temperature, the SA search moves to the new_state or not (line 11 – 15). During the search, SA keeps track of the best state ($best\_state$) and corresponding action sequence ($best\_sequence$) according to the objective function $F^{\text{RMSF}}_{\text{net}}$. Finally, $best\_state$ and $best\_sequence$ generated during all the iterations are returned.

V. Re-optimization with Bounded Number of Actions

Although the SA algorithm from Section IV is capable of systematically exploring a large solution space, it may require an excessively large number of reconfiguration actions to reach the re-optimized state. A long sequence of reconfiguration steps can cause a long period of network instability, rendering the SA algorithm impractical in realistic settings. Another drawback of SA algorithm is the lack of fairness among the involved VLinks, *i.e.*, some VLinks may be subject to a significantly more number of reconfiguration actions than others, resulting in longer period of instability for those VLinks. To resolve these two limitations of the SA algorithm, we propose a greedy algorithm that can provide a satisfactory re-optimization performance using a bounded number of total actions and per-VLink actions. In this section, we first discuss the two constraints to bound the number of actions, and then describe how the greedy algorithm enforces these constraints.

Maximum number of actions: This bound defines the maximum number of actions (M_{max}) the re-optimization process can take, similar to the bound proposed in [37]. The solution to the re-optimization problem will generate an ordered sequence of at most M_{max} actions that lead to the best possible re-optimized state.

Maximum number of actions per-VLink: This bound enforces a limit on the maximum number of actions (A_{max}) the re-optimization process can apply on each VLink. Similar bounds have been applied to ensure fairness during defragmentation of wavelength division multiplexed optical networks [37]. Note that one could easily impose a different bound on different VLinks to facilitate a differentiated treatment of VLinks during re-optimization. Such differential treatment can enable a variety of service level agreements where a VLink from the highest priority class is not impacted during re-optimization, while a VLink from best-effort class goes through a substantially large number of re-configurations.

A. Greedy Algorithm

The greedy algorithm presented in Algorithm 3 takes the current set of VN embeddings ($\mathcal{C}$), maximum number of iterations to perform (it_{max}), number of inner iterations (it_{inn}), M_{max}, and A_{max} as inputs. The greedy algorithm follows a similar flow of SA algorithm from Algorithm 2 with three major differences, while ensuring the constraints imposed by the two bounds we discussed. First, Algorithm 3 selects the best neighbor in terms of $F^{\text{RMSF}}_{\text{net}}$ as opposed to selecting a neighbor chosen based on a probability distribution in Algorithm 2. To do so, Algorithm 3 invokes Best-Neighbor procedure that returns the neighbor with the lowest value of $F^{\text{RMSF}}_{\text{net}}$ among the neighbors generated by applying the actions from Table I on a randomly selected split Y. Best-Neighbor procedure is a modified version of Algorithm 1 that first picks a random split Y which has been subjected to *Slot Usage Limit* and has not exceeded A_{max} actions, and returns the neighbor with the highest RMSF gain by applying the actions from Table I on split Y. We do not present Best-Neighbor procedure for the sake of brevity. If the neighbor returned by Best-Neighbor procedure improves $F^{\text{RMSF}}_{\text{net}}$ from the $current\_state$, Algorithm 3 takes the corresponding action and moves to the neighbor state. Since Algorithm 3 selects neighbor in a greedy fashion, it can get stuck to a local minima. To circumvent this issue, Algorithm 2 has a provision to move to a worse neighbor than the $current\_state$ with a low probability as discussed next.

The second difference of Algorithm 3 with Algorithm 2 is that Algorithm 3 moves to a worse neighbor than the $current\_state$ only when it is necessary as opposed to stochastically moving to a worse neighbor in Algorithm 2. This move is triggered when the neighbors of all the splits have worse $F^{\text{RMSF}}_{\text{net}}$ values than $F^{\text{RMSF}}_{\text{net}}$ of $current\_state$. This is achieved by incrementing a counter ($it\_counter$) when a worse neighbor is found and comparing $it\_counter$ with total number of splits (no_of_splits) in line 24 – 25. When a better neighbor is found in a particular iteration, $it\_counter$ is reset to start over (Line 12). Finally, Algorithm 3 has a terminating condition when the number of applied actions reaches the threshold M_{max} in line 21. When such condition is reached, Algorithm 3 returns the best neighbor found during the search. Note that we also tried a naive greedy algorithm that chooses a split Y on the EON link with the maximum fragmentation and does not take a worse neighbor at any time. Such a naive greedy algorithm gets stuck to a local minima very soon during the search. Hence, our proposed greedy algorithm

Algorithm 3: Greedy-Re-Optimize

```
function Gr-Re-Optimize($\mathcal{C}$, $it_{max}$, $it_{inn}$, $M_{max}$, $A_{max}$)
    $iterations \leftarrow 0$, $move\_counter \leftarrow 0$
    $action\_sequence \leftarrow best\_sequence \leftarrow \phi$
    $best\_state \leftarrow current\_state \leftarrow \mathcal{C}$
    while $iterations < it_{max}$ do
        $it\_counter \leftarrow 0$
        while $iterations_t < it_{inn}$ do
            $current\_cost \leftarrow F_{\text{net}}^{\text{RMSF}}(current\_state)$
            $< new\_state, action\_spec > \leftarrow$ Best-Neighbor
              $(current\_state, \delta, \Gamma)$
            $new\_cost \leftarrow F_{\text{net}}^{\text{RMSF}}(new\_state)$
            if $new\_cost < current\_cost$ then
                $it\_counter \leftarrow 0$
                Increment $move\_counter$
                $current\_state \leftarrow new\_state$
                $current\_cost \leftarrow new\_cost$
                $action\_sequence.append(action\_spec)$
                if $current\_cost < best\_cost$ then
                    $best\_cost \leftarrow current\_cost$
                    $best\_state \leftarrow current\_state$
                    $best\_sequence \leftarrow action\_sequence$
                if $move\_counter = M_{max}$ then
                    return $< best\_state, best\_sequence >$
            else
                Increment $it\_counter$
                if $it\_counter > no\_of\_splits$ then
                    $current\_state \leftarrow new\_state$
                    $current\_cost \leftarrow new\_cost$
                    $action\_sequence.append(action\_spec)$
            Increment $iterations_t$
        Increment $iterations$
    return $< best\_state, best\_sequence >$
```

blends greedy selection of neighbors with random choice of a split at the beginning and has the provision of selecting a worse neighbor when needed.

VI. DRL for Balancing the Trade-off Between Number of Performed Re-Optimization Operations and Acceptance Ratio

In the reoptimization process, we have a trade-off between the performance improvement and the increase in operational complexity. In other words, invoking the re-optimization operations too frequently might cause unnecessary operational complexity since the re-optimization process may involve spectrum re-configuration or lightpath provisioning/termination, while not performing the re-optimization operation in the some crucial situations can lead to a severe decrease in the number accepted VN requests.

For this purpose, we propose a DRL-based algorithm to determine the timing of the re-optimization operations intelligently so that we can achieve a significant improvement in acceptance ratio while keeping the number of re-optimization operations at a reasonable level.

In the proposed algorithm, after every K incoming VN requests (e.g., $K = 40$), the DRL agent decides based on the status of the substrate network whether to apply the re-optimization operation or not. The components of the DRL formulation are as the following:

RL steps: step $l = i \in \mathbb{N}$ is when $K \times i$ VN requests have arrived, so each step represents arrival of K VN requests.

RL state space: the state $s_l \in \{-1, +1\}^{|E| \times |S|}$ describes whether each spectrum slot in each SLink of the EON is occupied $(+1)$ or not (-1). $|E|$ is the number of SLinks and $|S|$ is the number of spectrum slots in each SLink.

RL action space: binary action $a_l \in \{0, 1\}$ determines whether in step l the re-optimization operation should be performed $(a_l = 1)$ or not $(a_l = 0)$.

RL reward: we define the reward of the DRL agent to be the acceptance ratio of the next K incoming VN requests, plus a negative penalty for choosing to apply the re-optimization operation. In this way, the agent is encouraged to balance the trade-off between the number of performed re-optimization operations and the overall acceptance ratio of the VN requests. So, we have $r_l = \delta_l - c \times a_l$, where δ_l is the ratio of accepted VN request between steps l and $l + 1$, and c is the chosen penalty as the cost of performing a re-optimization operation.

RL episode: each training episode should be a certain duration of EON operation, possibly starting with an empty EON and then allocating/deallocating VN requests that dynamically arrive and depart according to some specific life times. We have described the training episode in our experiments specifically in Section VII.

As the DRL algorithm, we choose the Soft Q-Network (SQN) method [20], which is a modification of the well-known Deep Q-Network (DQN) method that adds an entropy regularizer to the standard Markov Decision Process (MDP) formulation, and is shown to be more robust, less likely to overfit, more sample efficient, and handles the exploit-explore balance better than DQN [20], [38], [39]. Since we are dealing with a discrete (binary) action space, the SQN would be a very suitable RL algorithm for our problem.

A high-level overview of the SQN's interaction with the EON environment is depicted in Fig. 2, and its training procedure is described in Algorithm 4. At each training step, the agent observes the status of the EON's SLinks as the current state s, and decides whether to invoke re-optimization or not (binary action a). Then, after K incoming VN requests, the status of the substrate network is considered as the next state s', and the reward value can be calculated as described before. The experience (s, a, s', r) is stored in a memory referred to as the replay (or experience) buffer. For updating the SQN according to the (soft) Bellmam equation, instead of only using the latest experience, a mini-batch is sampled from the experience buffer and used for updating the neural network to prevent oscillations in the action values. This technique of using a large buffer of past experience and sampling training data from it is called experience replay [40], and allows to learn from individual experience multiple times and in general make a better use of experience. This procedure is repeated for the next states and for multiple iterations (episodes) until the convergence of the DRL agent.

VII. Evaluation

A. Simulation Setup

We implement the algorithms presented in Section VII-B using C++. We consider a fully-flexible EON using Nobel

Algorithm 4: SQN Training

```
initialize the SQN weights
for each training episode do
    wait for the first $K$ VNs to arrive
    observe current state $s$
    while episode is not finished do
        select action $a$ and execute it
        wait for the next $K$ VNs to arrive
        calculate reward $r$
        observe new state $s'$
        add $(s, a, s', r)$ to experience buffer
        sample mini-batch of experiences from buffer
        for each experience in mini-batch, perform gradient
          update for the SQN by the soft Bellman equation
        update state $s \leftarrow s'$
    end of episode
end of training
```

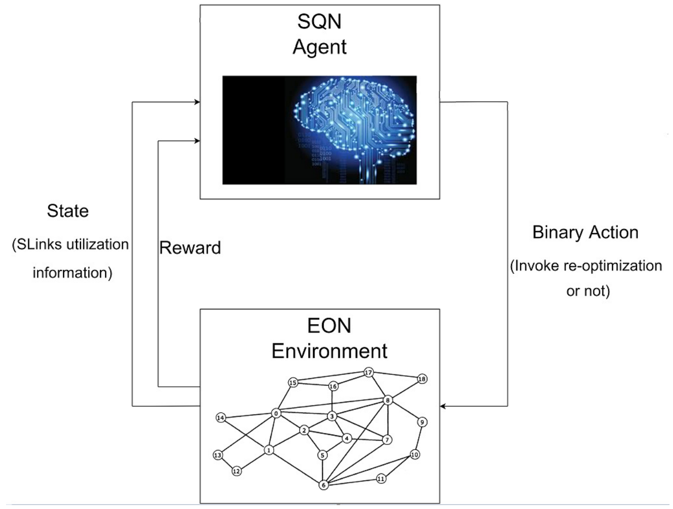


Fig. 2: Overview of the DRL agent's interaction with the EON environment

Germany topology (17 nodes and 26 links) from the SNDlib Repository (available at http://sndlib.zib.de). Each EON link has 4THz spectrum bandwidth divided into 160 slots of 25GHz. To emulate a live EON, we develop a discrete event simulator. The simulator loads the EON with VNs by simulating VN arrival and departure events and allocating and releasing spectrum slots to virtual links accordingly. In our simulator, VN arrival rate (VNR) follows a Poisson distribution with varying means (4-20 VNs per 100 time units) and VN life time is exponentially distributed with a mean of 500 time units. The number of VNodes vary from 2 to 6 and number of VLinks vary from 3 to 15 (randomly chosen). When a VN arrives, the simulator embeds the VN using the algorithm proposed in [8]. This simulator generates snapshots of the EON's current occupation at different time instances in which a varying number of VNs are embedded. To analyse the performance of the compared algorithms in terms of RMSF reduction, we take different snapshots of the EON at different utilizations (varying from 30% to 60%). For each network utilization, we pick five random snapshots. For each snapshot, we independently run the compared variants five times and take the output of the run that achieves the best defragmentation. Finally, we report the average (with maximum and minimum as errorbars) of the best performances of each five different snapshots.

Then, to analyse the performance of applying re-optimization in terms of acceptance ratio, we run a similar simulation and after every $K = 40$ incoming VN requests, according to the chosen time selection strategy, we decide whether to perform our re-optimization operation ($Gr-Re-Optimize$) on the EON or not. We have reported the acceptance ratio and number of re-optimization operations of different time selection strategies in Section VII-F.

We trained the DRL agent for around 6000 episodes of only simulations with $VNR = 12$ (other VNR values are not used during training). Each training episode was running the simulation for $5,000$ time units; since we have $VNR = 12$ and $K = 40$, around 600 VNs will arrive during the simulation and the training episode will have around 15 RL steps (as defined in Section VI). Fig. 3 shows the reward of the DRL agent during the training episodes. We can see in the figure that after a certain number of training episodes, the aggregated reward of the DRL agent in an episode has significantly improved compared to the beginning where the DRL's actions are mostly chosen at random. This shows that the agent gradually learns the most effective times to invoke re-optimization and also not to perform too many re-optimizations due to the negative penalty we have considered for each re-optimization operation.

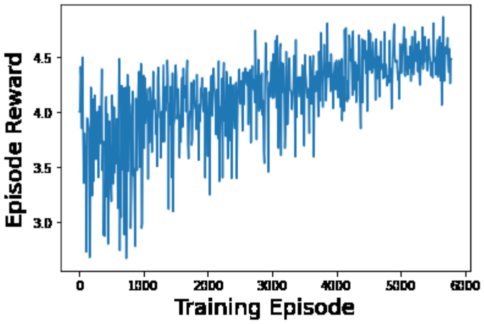


Fig. 3: Episode rewards of the DRL agent trained with the VNR12 scenario

B. Compared Variants for Slice Embedding Re-optimization

1) SA-baseline: This variant represents SA-Re-Optimize algorithm (Algorithm 2) with a huge number of iterations for $it_{\max}$ and it_{temp}. The values of $T_0 = 100$ and $\rho = 0.99$ for Algorithm 2 are chosen as best performing ones from multiple trials. This variant also uses *Slot Usage Limit* as $\delta = 10\%$ and *Neighborhood Limit* as $\Gamma = 10\%$ for Algorithm 1 chosen based on trials. *SA-baseline* is used to show convergence of SA-Re-Optimize algorithm, and it provides a feasible lower bound for the re-optimization problem.

2) SA-Re-Optimize: This variant shows the performance of SA-Re-Optimize algorithm (Algorithm 2) with a fixed number of total iterations *i.e.*, $it_{\max} = 1000$, $it_{\text{temp}} = 1000$ and $T_0 = 100$ and $\rho = 0.99$.

3) Gr-bounded: This variant represents Gr-Re-Optimize algorithm (Algorithm 3). In this case, we set $M_{\max} = 500$ and $A_{\max} = \infty$ in Algorithm 3. We also vary $M_{\max}$ and $A_{\max}$ to show sensitivity of Algorithm 3.

4) SA-SOA: Simulated Annealing State-of-the-art (SA-SOA) is a variant that represents the scenario where we minimize our cost function $F_{\text{net}}^{\text{RMSF}}$ (*i.e.*, Eqn. (2)) by using the SA-based defragmentation mechanism presented in [30]. To the best of our knowledge the work in [30] is the closest to SA-Re-Optimize algorithm (Algorithm 2). The mechanism presented in [30] is not specific to any cost function, therefore, we use our cost function instead of theirs for a fair comparison.

C. Compared Approaches for Time Selection of the Re-optimization Operations

1) Freq-Re-Optimize: In this approach, the re-optimization operation is frequently performed on the EON after every $K = 40$ incoming VN requests. This approach yields the highest number of re-optimization operations in our experiments.

2) Less-Freq-Re-Optimize: This approach is a less frequent version of *Freq-Re-Optimize*, where the re-optimization operation is performed on the EON after every $K_2 > K$ incoming VN requests (e.g., $K_2 = 4 \times K$).

3) Thr-Re-Optimize: In this approach, the re-optimization operation is invoked whenever the RMSF metric exceeds a predefined threshold value.

4) Window-Re-Optimize: In this approach, the times for performing the re-optimization operations are chosen based on the algorithm presented by [33]. In this algorithm, after each time interval Δt, the temporary blocking probability during the latest Δt is calculated and if it exceeds a pre-defined threshold, the re-optimization process is invoked. Moreover, after performing a re-optimization operation, based on the behavior of the calculated blocking probabilities in the previous time intervals, Δt will be increased or decreased to adopt to the current status of the network.

5) DRL-Re-Optimize: In this approach, after every $K = 40$ incoming VN requests, our trained DRL agent decides whether to invoke the re-optimization process or not based on the current status of the EON.

D. Performance Metrics

- RMSF Reduction: It is the reduction of fragmentation achieved by an algorithm in terms of RMSF ($F_{\text{net}}^{\text{RMSF}}$) from the RMSF of the given snapshot before re-optimization. It is defined as (1 - (the ratio of $F_{\text{net}}^{\text{RMSF}}$ of the EON after re-optimization to $F_{\text{net}}^{\text{RMSF}}$ of the EON before re-optimization)). The desired value of RMSF Reduction is 1 but that is not practically achievable. However, the closer RMSF Reduction is to 1, the more defragmentation is achieved through re-optimization (*e.g.*, roughly, RMSF reduction of 0.95 means that 95% fragmentation has been reduced compared to the fragmentation before re-optimization).
- Slot Ratio: It is the ratio between the total spectrum slot usage by the VNs after re-optimization and the total spectrum slot usage by the VNs before re-optimization. Slot Ratio smaller than 1 means that spectrum occupation has decreased after re-optimization.
- Action Count: It is the total number of sequential actions adopted by an algorithm to reach its re-optimized state.
- Number of Actions per VLink: It is the average number of actions applied on each VLink by an algorithm to reach its re-optimized state.
- Acceptance Ratio: Ratio of the number of accepted VN requests to the total number of requests during an episode of the simulation.
- Number of Re-optimization Operations: Total number of performed re-optimization operations during an episode of the simulation.

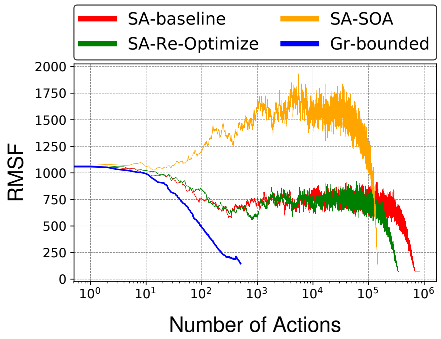
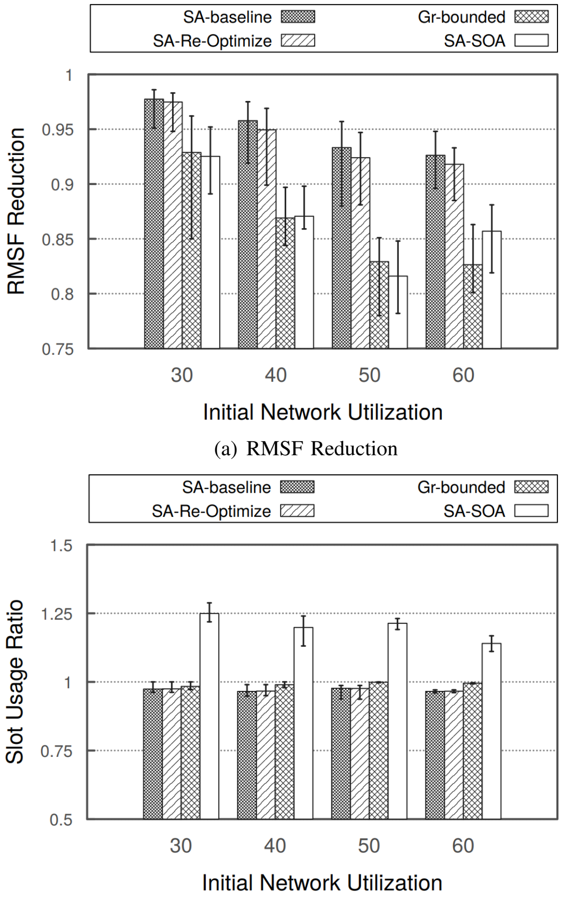


Fig. 4: Convergence of Algorithms

E. Discussion of Results on RMSF Reduction

1) Comparison of Different Algorithms: Fig. 4 shows that convergence of the compared variants in terms of RMSF ($F_{\text{net}}^{\text{RMSF}}$) values of the EON against number of actions for a specific snapshot with 60% utilization. This figure shows that *SA-baseline* converges to a feasible lowerbound if it is allowed to run for a sufficiently long time. On the other hand, *SA-Re-Optimize* does not converge to a steady value but achieves an RMSF very close to *SA-baseline*'s lowerbound despite being run for a fixed number of iterations. Similarly, *SA-SOA* diverges initially but reaches to an RMSF value closer to the lowerbound through a large number of steps. In comparison to SA based approaches, *Gr-bounded* decreases RMSF of the EON sharply and achieves an RMSF close to *SA-baseline*'s lowerbound even by applying a limited number of actions.

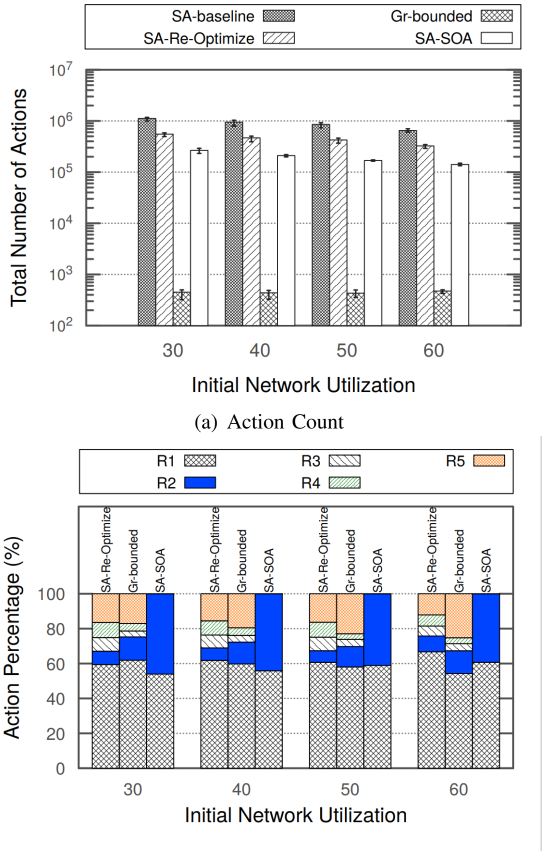
Fig. 5 compares different algorithms in terms of the performance metrics introduced before. In Fig. 5(a), RMSF reduction decreases for larger initial network utilization. This is rational since a highly utilized EON offers less room to re-optimize. Fig. 5(a) also shows that *SA-Re-Optimize* closely approximates *SA-baseline*, and both achieve great performance by reducing more than 90% fragmentation in the given snapshots. On the other hand, *Gr-bounded* and *SA-SOA* achieve lower reductions, yet reductions consistently remain within 10% of *SA-baseline*. *Gr-bounded*, despite using limited number of actions, achieves higher or equal RMSF reduction than *SA-SOA* in all cases except for the snapshots with the highest utilization (60%). Even in the constrained snapshots of high utilization, *Gr-bounded* reduces more than 80% fragmentation in the EON. If not properly designed, defragmentation algorithms might lead to an increase in spectrum slot usage. Our proposed approaches (*SA-baseline*, *SA-Re-Optimize*, and *Gr-bounded*) succeed instead in decreasing slot usage compared to the slot requirement of given snapshots (see Fig. 5(b)) thanks to *Slot Usage Limit* (δ) in Algorithm 1. Note instead that, in the



(a) RMSF Reduction

(b) Slot Usage Ratio

Fig. 5: Performance of different algorithms



(a) Action Count

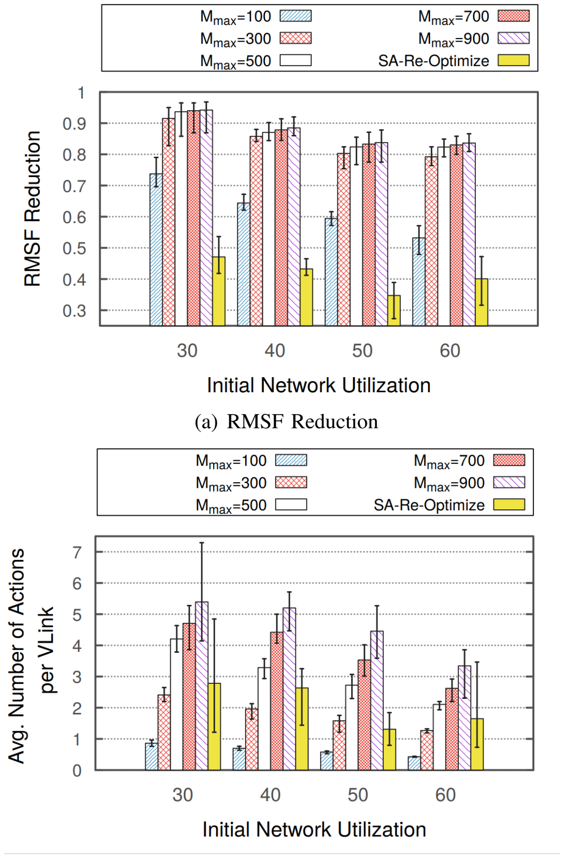
(b) Action Distribution

Fig. 6: Actions taken by different algorithms

case of *SA-SOA* that does not employ any *Slot Usage Limit*, defragmentation requires more than 20% additional spectrum slots (while this increase is avoided by our algorithms).

Note that RMSF reductions of SA based approaches (*SA-baseline*, *SA-Re-Optimize*, and *SA-SOA*) in Fig. 5(a) require a huge number of re-configuration actions to converge to the most optimized final state (see Fig. 6(a)), in the order of 100's of thousands, which makes their application impractical in a realistic operational settings. In contrast, *Gr-bounded* achieves more than 80% reduction on fragmentation with no additional spectrum usage by employing only up to 500 re-configuration actions (on average, 2 to 3 actions per VLink), making *Gr-bounded* a more practical solution to be leveraged by a network operator. Finally, Fig. 6(b) shows the distribution of different actions from Table I adopted by *SA-Re-Optimize*, *SA-SOA*, and *Gr-bounded*. According to this figure, usage of action R_1 comprises the major percentages, while the proposed re-configuration actions ($R_3 - R_5$) have been used in more than 20% cases of *SA-Re-Optimize* and *Gr-bounded*. We also observe that the action R_4 which causes disruption has been used very rarely in case of *Gr-bounded*.

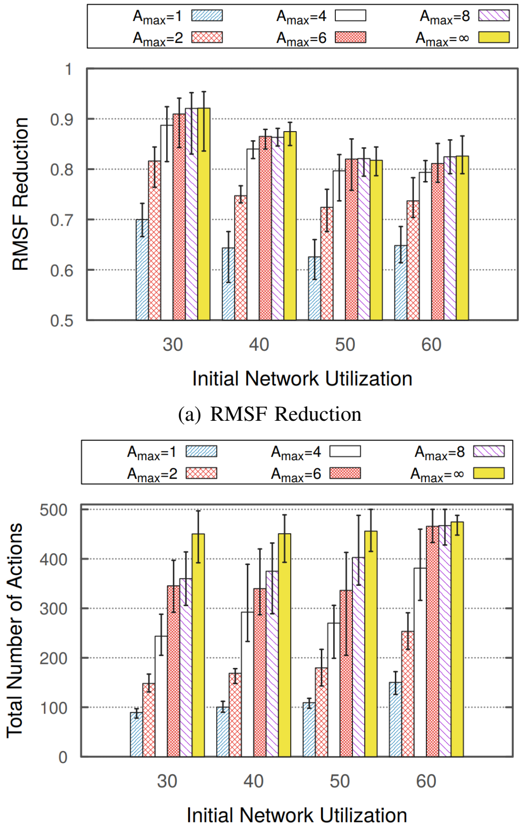
2) Analysis of Gr-bounded: Fig. 7 shows the impact of varying $M_{\max}$ (Maximum number of actions) on the performance of *Gr-bounded* with $A_{\max} = \infty$ and for different values of initial network utilization. Fig. 7 also compares the performance of *Gr-bounded* having different $M_{\max}$ bounds with the best possible re-optimized state that could be achieved by *SA-Re-Optimize* in limited number of steps ($M_{\max} = 1000$). Fig. 7(a) shows that *Gr-bounded*, even with the smallest $M_{\max}$ ($M_{\max} = 100$) achieves 20-30% more defragmentation than the *SA-Re-Optimize* with $M_{\max} = 1000$. We also observe that *SA-Re-Optimize* with $M_{\max} = 1000$ could not even reduce RMSF more than 50% in any case. The reason for such poor initial performance of *SA-Re-Optimize* is that it allows to move to a worse neighbor with a high probability during the initial stages of SA search that diverges *SA-Re-Optimize* within $M_{\max} = 1000$ actions. Such behavior is fundamental to the SA algorithm and, for this reason, *SA-Re-Optimize* cannot be used in a practical setting where the number of actions is limited. Another takeaway from Fig. 7 is that RMSF reduction of *Gr-bounded* increases drastically when the value of $M_{\max}$ goes from 100 to 300 and RMSF reduction stabilizes for $M_{\max} \geq 500$. This justifies our choice of $M_{\max} = 500$ for the other evaluations of *Gr-bounded*. Finally, Fig. 7(b) shows that *Gr-bounded* applies on average one action to each VLink when $M_{\max}$ is low ($M_{\max} = 100$). For *SA-Re-Optimize*, we keep the best re-optimized state within $M_{\max} = 1000$ that can be reached with lower number of actions per VLink. After reaching the best re-optimized state, *SA-Re-Optimize* diverges to worse solutions, and hence we ignore those actions justifying the lower number of per-VLink actions. This figure also shows that average number of actions per VLink increases with the increase in $M_{\max}$ and decreases when network uti-



(a) RMSF Reduction

(b) Action Count per VLink

Fig. 7: Performance of *Gr-bounded* by varying $M_{\max}$



(a) RMSF Reduction

(b) Action Count

Fig. 8: Performance of *Gr-bounded* by varying $A_{\max}$

lization increases. This is due to the fact that, as utilization increases, snapshots have more VLinks, and the same total number of actions are distributed among more VLinks thus reducing number of actions per VLink.

Let us now discuss the impact of varying $A_{\max}$ (Maximum number of actions per-VLink) on the performance of *Gr-bounded* with $M_{\max} = 500$ and for different values of initial network utilization. Recall from Section V, $A_{\max} = x$ means that a VLink can go through at most x number of re-configuration actions during the whole re-optimization process. Fig. 8(a) shows that *Gr-bounded* with $A_{\max} = 1$ achieves the lowest RMSF reduction as it allows a single re-configuration action for all the splits of a VLink. Such a conservative version of *Gr-bounded* still reduces more than 60% fragmentation using the smallest number of actions as shown in Fig. 8(b). As we loosen the constraint of $A_{\max}$ (by increasing its value), a VLink can be re-configured more than once offering more opportunities to re-optimize. Hence, RMSF reduction increases for increasing $A_{\max}$. Such gain is more prominent for smaller values of $A_{\max}$ while smaller increases in RMSF reduction are observed for $A_{\max} \geq 6$. This is due to the fact that the other limit of Maximum number of actions ($M_{\max} = 500$) dominates in *Gr-bounded* for $A_{\max} \geq 6$. Similar observations apply to the total number of actions in Fig. 8(b). Another takeaway from Fig. 8(b) is that action count increases with the increase in utilization of snapshots. This is expected as a snapshot with higher utilization has more VLinks that requires more actions to be applied.

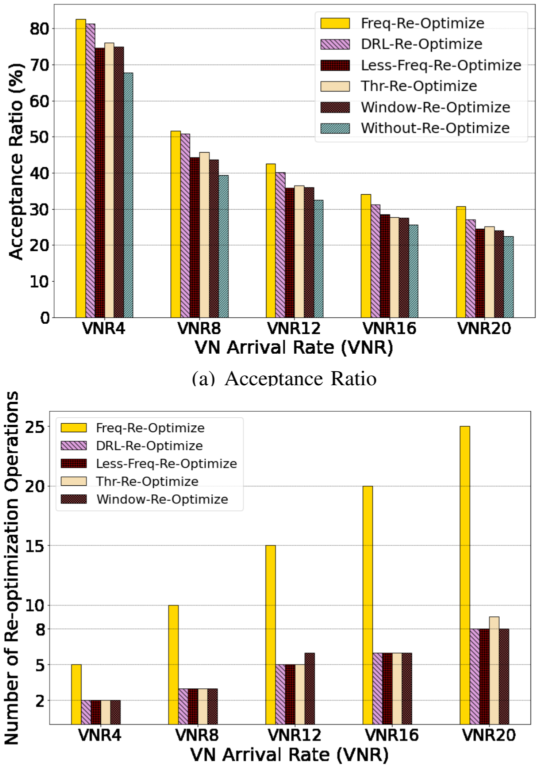
F. Discussion of Results on Different Re-optimization Time Selection Strategies

Fig. 9 shows the acceptance ratio and number of re-optimization operations of different time selection approaches for different values of VNR. As shown in Fig. 9(a), applying the greedy re-optimization operation on the network (*Freq-Re-Optimize*) leads to considerable improvement in the acceptance ratio compared to the case with no re-optimization, and this improvement is more significant for lower VN arrival rates (8% for $VNR = 20$ and 15% for $VNR = 4$). This shows that there is a high correlation between the fragmentation level (RMSF) of the EON and its capability to accept more traffic, and reducing RMSF is a very suitable indirect approach for increasing the acceptance ratio when there is no priori knowledge of the future VN requests.

Moreover, we can see that the acceptance ratio of our proposed *DRL-Re-Optimize* is very close to the *Freq-Re-Optimize* case, while performing much fewer re-optimization operations (one-third on average, as shown in Fig. 9(b)). This means that our RL algorithm has successfully detected the most crucial moments for performing the re-optimization process to achieve a high degree of performance improvement with limited number of re-optimization operations. Moreover, the figure shows that other time selection strategies with similar number of re-optimization operations as our RL algorithm (namely, *Less-Freq-Re-Optimize*, *Thr-Re-Optimize*,

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2022.3230381

and *Window-Re-Optimize*) have a much lower acceptance ratio than our RL-based algorithm. Among the compared approaches, *Window-Re-Optimize* has the worst performance, since it considers the blocking probability in the latest time window as the metric for estimating the level of fragmentation in the network, while blocking probability depends also on the overall link utilization of the EON. As a result, this method can lead to only invoking the re-optimization when the overall link utilization is high, which might not be the optimum strategy. Furthermore, *Thr-Re-Optimize* has a slightly better performance than *Less-Freq-Re-Optimize*, because it takes into account the RMSF metric for determining the right times for performing re-optimization, as opposed to *Less-Freq-Re-Optimize* that simply invokes the operations in fixed intervals. As we mentioned earlier, the DRL agent for this evaluation is only trained in simulations with $VNR = 12$, however, we can see in Fig. 9 that it performs well for other VNR cases that have not been seen during training as well. We will evaluate the generalization capability of our DRL algorithm in the next section with more details and experiments.



(a) Acceptance Ratio

(b) Number of Re-optimization Operations

Fig. 9: Performance of different time selection strategies for different VNRs

G. Generalization Capability of the DRL Algorithm

Since real-life application scenarios might be different than our simulation environment, we would desire a generalizable DRL agent that does not overfit on the specific characteristics of our training environment, and is applicable to different scenarios that differ in distribution from those generated in our training setup. To analyze this aspect of our proposed algorithm, we evaluate the DRL agent on scenarios that were unseen during training, both in terms of VNR value and life-time distribution, as described below:

TABLE II: Acceptance ratio (A.R.) and number of re-optimization operations (#Oper.) of RL agents trained with different VNR scenarios evaluated in different VNR cases (e.g., $RL4$ is an agent that is only trained with $VNR4$ scenario).

VNR/ RL Agent	RL4	RL12	RL20
VNR4	A.R. = 81.5% #Oper. = 2	A.R. = 81.3% #Oprations = 2	A.R. = 80.8% #Operations = 2
VNR12	A.R. = 39.4% #Oper. = 6	A.R. = 40.2% #Oper. = 5	A.R. = 39.7% #Oper. = 5
VNR20	A.R. = 27.2% #Oper. = 9	A.R. = 27.1% #Oper. = 8	A.R. = 27.5% #Oper. = 8

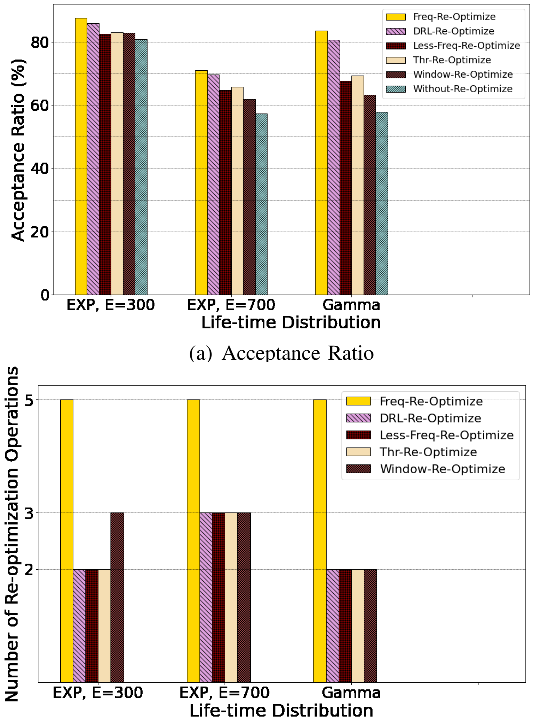
1) Different VNR Values: We already showed in Section VII.C that the DRL agent trained only with the $VNR12$ case performed well on other VNR scenarios. For further evaluation, here, we have trained two other DRL agents trained with $VNR4$ and $VNR20$ scenarios (named $RL4$ and $RL20$) and reported their performance on different VNR cases in Table II. We can see that for each VNR case, the agent that is trained specifically for that VNR case has the highest acceptance ratio (e.g., $RL12$ and $VNR12$). However, the degradation in their performance for VNR cases that differ from the one they were trained for seems to be insignificant, specially for $RL12$ and $RL20$ agents. For example, the difference in acceptance ratio of $RL12$ and $RL20$ in different cases never exceeds 0.5%. However, we can see that the $RL4$ agent has a slightly worse performance compared to the other two agents, since the number of steps in each episode of training $RL4$ is much smaller than the other two agents. Therefore, we can argue that agents trained with high VNR values are more generalizable in comparison.

2) Different life-time distributions: In Fig. 10, we have trained a DRL agent with $VNR12$ scenario and exponential life-time distribution with a mean of 500 time units, and we are evaluating the agent in scenarios with $VNR4$ and different life-time distribution, namely an exponential distribution with a mean of 300 time units, an exponential distribution with a mean of 700 time units, and a Gamma distribution with shape argument of 0.2 and scale argument of 1600, that simulates a scenario where most of the life times are concentrated around a low value while a few outliers are significantly higher than this value. So, in these experiments, both the VNR value and life-time distribution are different from the training environment. We can see in the figure that the DRL method still has a higher acceptance ratio compared to other approaches that have the same number of re-optimization operations.

VIII. Conclusion

In this paper, we have addressed the re-optimization of network slice embedding over EON with the objective to minimize spectrum fragmentation. Given an EON with a set of embedded VNs, the problem is to re-optimize the existing VN embeddings by employing a series of different re-configuration actions. We advance the state-of-the-art by addressing, for the first time, the spectrum defragmentation problem in the context of splitting-enabled EON and by proposing novel

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2022.3230381



(a) Acceptance Ratio

(b) Number of Re-optimization Operations

Fig. 10: Performance of different time selection strategies for $VNR4$ and different life-time distributions

re-configuration actions that offer more opportunities to re-optimize. Given the complex and non-linear nature of this problem, we have proposed simulated annealing based algorithm for systematically exploring its large solution space. We also propose a greedy search mechanism to address the practical constraint to limit the number of re-configuration steps taken to reach a final defragmentated state. For improving the number of accepted VN request through a reasonable number of re-optimization operations, we proposed a DRL-based algorithm to determine the most effective moments for invoking re-optimization.

Our extensive simulation results demonstrate that the simulated annealing based algorithm reduces more than 90% fragmentation at the expense of employing a huge number re-configuration actions. Our results also show that the proposed greedy search algorithm, if we consider a scenario with limited number of re-configuration actions, achieves better defragmentation (reduces more than 80% fragmentation) than a prior approach that employs simulated annealing. The greedy search algorithm can impose different level of bounds on the number of actions to be taken, enabling a fair or prioritized treatment of VLinks during re-optimization. Moreover, our results show that, when the number of admissible reconfiguration action is limited (*e.g.*, below 500), having the possibility to exploit a comprehensive set of reconfiguration actions (involving merging and dividing splits of a virtual link) allows to achieve noticeable improvement in defragmentation. We also evaluated the effect of our proposed greedy re-optimization algorithm on increasing the network ability to accept more traffic where the times of the re-optimization operations were chosen according to different strategies. We showed that our proposed DRL-based method outperformed the existing time selection methods with the same number of re-optimization operations up to 23% in acceptance ratio.

References

[1] B. Yan *et al.*, “Tidal-traffic-aware routing and spectrum allocation in elastic optical networks,” *IEEE/OSA Journal of Optical Communications and Networking*, vol. 10, no. 11, pp. 832–842, 2018.

[2] M. Hadi, M. R. Pakravan, and E. Agrell, “Dynamic resource allocation in metro elastic optical networks using lyapunov drift optimization,” *J. of Opt. Commn. and Net.*, vol. 11, no. 6, pp. 250–259, 2019.

[3] Z. Zhong *et al.*, “Provisioning short-term traffic fluctuations in elastic optical networks,” *IEEE/ACM TNet.*, vol. 27, no. 4, pp. 1460–1473, 2019.

[4] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina, “Network slicing in 5g: Survey and challenges,” *IEEE Communications Magazine*, vol. 55, no. 5, pp. 94–100, 2017.

[5] S. Aleksic, “Towards fifth-generation (5g) optical transport networks,” in *Proceedings of ICTON*, 2015, pp. 1–4.

[6] R. Boutaba, N. Shahriar, and S. Fathi, “Elastic optical networking for 5G transport,” *Springer JNSM*, vol. 25, no. 4, pp. 819–847, 2017.

[7] L. Gong and Z. Zhu, “Virtual optical network embedding (VONE) over elastic optical networks,” *IEEE/OSA Journal of Lightwave Technology*, vol. 32, no. 3, pp. 450–460, 2014.

[8] N. Shahriar *et al.*, “Achieving a fully-flexible virtual network embedding in elastic optical networks,” in *IEEE INFOCOM*, 2019, pp. 1756–1764.

[9] B. C. Chatterjee, S. Ba, and E. Oki, “Fragmentation problems and management approaches in elastic optical networks: A survey,” *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 183–210, 2017.

[10] H. Beyranvand *et al.*, “An analytical framework for the performance evaluation of node-and network-wise operation scenarios in elastic optical networks,” *IEEE TComm*, vol. 62, no. 5, pp. 1621–1633, 2014.

[11] 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.

[12] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, “Asynchronous methods for deep reinforcement learning,” in *International conference on machine learning*. PMLR, 2016, pp. 1928–1937.

[13] Y. Ujjwal and J. Thangaraj, “Review and analysis of elastic optical network and sliceable bandwidth variable transponder architecture,” *Optical Engineering*, vol. 57, no. 11, p. 110802, 2018.

[14] A. Pagès, J. Perelló, S. Spadaro, and J. Comellas, “Optimal route, spectrum, and modulation level assignment in split-spectrum-enabled dynamic elastic optical networks,” *IEEE/OSA Journal of Optical Communications and Networking*, vol. 6, no. 2, pp. 114–126, 2014.

[15] F. Cugini, F. Paolucci, G. Meloni, G. Berrettini, M. Secondini, F. Fresi, N. Sambo, L. Poti, and P. Castoldi, “Push-pull defragmentation without traffic disruption in flexible grid optical networks,” *IEEE/OSA Journal of Lightwave Technology*, vol. 31, no. 1, pp. 125–133, 2012.

[16] R. Proietti *et al.*, “Rapid and complete hitless defragmentation method using a coherent rx lo with fast wavelength tracking in elastic optical networks,” *Optics express*, vol. 20, no. 24, pp. 26 958–26 968, 2012.

[17] T. Takagi *et al.*, “Disruption minimized spectrum defragmentation in elastic optical path networks that adopt distance adaptive modulation,” in *ECOC*, 2011, pp. Mo–2.

[18] P. Lechowicz *et al.*, “Fragmentation-aware algorithm with bordering super-channels in spectrally/spatially-flexible optical networks,” *IEEE/OSA J. of Opt. Commn. and Net.*, 2020.

[19] E. Aarts and J. Korst, “Simulated annealing and boltzmann machines,” 1988.

[20] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine, “Reinforcement learning with deep energy-based policies,” in *International conference on machine learning*. PMLR, 2017, pp. 1352–1361.

[21] S. Taeb, N. Shahriar, S. R. Chowdhury, M. Tornatore, R. Boutaba, J. Mitra, and M. Hemmati, “Reoptimizing network slice embedding on eon-enabled transport networks,” in *2021 17th International Conference on Network and Service Management (CNSM)*. IEEE, 2021, pp. 292–300.

[22] J. Comellas, L. Vicario, and G. Junyent, “Proactive defragmentation in elastic optical networks under dynamic load conditions,” *Photonic Network Communications*, vol. 36, no. 1, pp. 26–34, 2018.

[23] S. Shakya *et al.*, “Virtual network embedding and reconfiguration in elastic optical networks,” in *IEEE GLOBECOM*, 2014, pp. 2160–2165.

[24] M. Zhu *et al.*, "Fragmentation-aware vone in elastic optical networks," *J. of Opt. Commn. and Net.*, vol. 10, no. 9, pp. 809–822, 2018.

[25] S. Fernández-Martínez, B. Barán, and D. P. Pinto-Roa, "Spectrum defragmentation algorithms in elastic optical networks," *Optical Switching and Networking*, vol. 34, pp. 10–22, 2019.

[26] R. Wang and B. Mukherjee, "Provisioning in elastic optical networks with non-disruptive defragmentation," *IEEE/OSA Journal of Lightwave Technology*, vol. 31, no. 15, pp. 2491–2500, 2013.

[27] M. Zhang *et al.*, "Dynamic and adaptive bandwidth defragmentation in spectrum-sliced elastic optical networks with time-varying traffic," *J. of Light. Tech.*, vol. 32, no. 5, pp. 1014–1023, 2014.

[28] S. Shakya and X. Cao, "Spectral defragmentation in elastic optical path networks using independent sets," in *National Fiber Optic Engineers Conference*. Optical Society of America, 2013, pp. NTh1I–4.

[29] E. J. Dávalos *et al.*, "Spectrum defragmentation in elastic optical networks: Two approaches with metaheuristics," *IEEE Access*, vol. 7, pp. 119 835–119 843, 2019.

[30] Y. Zeng *et al.*, "Defragmentation of flexible optical networks based on simulated annealing," in *IEEE ACP*, 2012, pp. 1–3.

[31] S. K. Singh, W. Bziuk, and A. Jukan, "Analytical performance modeling of spectrum defragmentation in elastic optical link networks," *Optical Switching and Networking*, vol. 24, pp. 25–38, 2017.

[32] J. Kim, S. Yan, A. Fumagalli, E. Oki, and N. Yamanaka, "An analytical model of spectrum fragmentation in a two-service elastic optical link," in *2015 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2015, pp. 1–6.

[33] M. Zhang, C. You, H. Jiang, and Z. Zhu, "Dynamic and adaptive bandwidth defragmentation in spectrum-sliced elastic optical networks with time-varying traffic," *Journal of Lightwave Technology*, vol. 32, no. 5, pp. 1014–1023, 2014.

[34] T. Ahmed *et al.*, "Dynamic routing, spectrum, and modulation-format allocation in mixed-grid optical networks," *IEEE/OSA J. of Opt. Commn. and Net.*, vol. 12, no. 5, pp. 79–88, 2020.

[35] G. Zhang *et al.*, "A survey on ofdm-based elastic core optical networking," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 1, pp. 65–87, 2012.

[36] S. B. Masti and S. V. Raghavan, "Simulated annealing algorithm for virtual network reconfiguration," in *IEEE Euro-NF Conf. on Next Gen. Internet*, 2012, pp. 95–102.

[37] H. Duong *et al.*, "Efficient make before break capacity defragmentation," in *IEEE HPSR*, 2018, pp. 1–6.

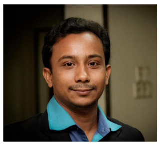
[38] N. Vieillard, O. Pietquin, and M. Geist, "Munchausen reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 4235–4246, 2020.

[39] T. Haarnoja, V. Pong, A. Zhou, M. Dalal, P. Abbeel, and S. Levine, "Composable deep reinforcement learning for robotic manipulation," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 6244–6251.

[40] L.-J. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," *Machine learning*, vol. 8, no. 3, pp. 293–321, 1992.

Seyed Soheil Johari is a graduate researcher at the David R. Cheriton School of Computer Science, University of Waterloo. He received his B.Sc. in electrical engineering from Sharif University of Technology in 2020. He was a recipient of the best student paper award at IEEE/IFIP NOMS 2022. His research focuses on the application of machine learning techniques for data-driven management and orchestration of 5G network slices.

Sepehr Taeb is a research assistant at the David R. Cheriton School of Computer Science, University of Waterloo. He received his B.Sc. in computer engineering from Sharif University of Technology in 2016, and the M.Math. degree in computer science from the University of Waterloo in 2019. His research interest includes network virtualization, optical networks, and Internet of drones.



Nashid Shahriar is an assistant professor in the Department of Computer Science at the University of Regina. He received his PhD from the School of Computer Science, University of Waterloo in 2020. He was a recipient of 2020 PhD Alumni Gold Medal, 2021 Mathematics Doctoral prize, Ontario Graduate Scholarship, President's Graduate Scholarship, and David R. Cheriton Graduate Scholarship with the University of Waterloo. His research received several recognitions, including the IEEE/IFIP NOMS 2022 Best Student Paper Award, IFIP/IEEE IM 2021 Best PhD Dissertation Award, the IEEE/ACM/IFIP CNSM 2019 Best Paper Award, IEEE NetSoft 2019 Best Student Paper Award, and the IEEE/ACM/IFIP CNSM 2017 Best Paper Award. His research interests include network virtualization, 5G network slicing, and network reliability.

Shihabur Rahman Chowdhury (S'13) is currently a Senior Software Engineer with the Apple Knowledge Platform. He received PhD in Computer Science from the University of Waterloo in 2021. His PhD dissertation "Resource Management in Softwarized Networks" was recognized by the 2021 University of Waterloo Alumni Gold Medal (PhD) and the 2022 IEEE/IFIP NOMS 2022 Best Dissertation Award. Shihabur's research interests include various aspects of building and managing networks and large-scale distributed systems. His contributions to the network management research has received several recognitions including, the 2021 IEEE CNOM Young Professional Award, and the Best Paper Awards at the IEEE/ACM/IFIP CNSM 2019, IEEE NetSoft 2019, and the IEEE/ACM/IFIP CNSM 2017 conferences.

Massimo Tornatore is an Associate Professor at Politecnico di Milano, Italy. He also held an appointment as Adjunct Professor at University of California, Davis, USA and as visiting professor at University of Waterloo, Canada. His research interests include performance evaluation and design of communication networks (with an emphasis on optical networking), and machine learning application for network management. He co-authored more than 400 conference and journal papers (with 19 best-paper awards) and of the recent Springer "Handbook of Optical Networks". He is member of the Editorial Board of IEEE Communication Surveys and Tutorials, IEEE Communication Letters, IEEE Transactions on Network and Service Management and IEEE/ACM Transactions on Networking.

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2022.3230381

Raouf Boutaba (F'12) M.Sc. and Ph.D. degrees in computer science from Sorbonne University in 1990 and 1994, respectively. He is currently a University Chair Professor and the Director of the David R. Cheriton School of Computer science at the University of Waterloo (Canada). He also holds an INRIA International Chair in France. He is the founding Editor-in-Chief of the IEEE Transactions on Network and Service Management (2007- 2010) and the current Editor-in-Chief of the IEEE Journal on Selected Areas in Communications. He is a fellow of the IEEE, the Engineering Institute of Canada, the Canadian Academy of Engineering, and the Royal Society of Canada. His research interests include resource and service management in networks and distributed systems.

Jeebak Mitra received the M.A.Sc. and Ph.D. degrees in electrical engineering from The University of British Columbia in 2005 and 2010, respectively. From 2010 to 2011, he was a Senior System Engineer with Riot Micro, leading the system level design for a local thermal equilibrium baseband. From 2011 to 2012, he was a Team Leader for physical layer DSP design with BLINQ Networks, Ottawa, focusing on small cell backhaul products. Since 2013, he has been a Senior Staff Engineer with the Huawei Technologies Canada Research Center, Ottawa, in the areas of algorithm design and implementation for coherent high-speed optical transceivers and flexible optical networks. His research interests lie in the area of high-performance communication systems design focusing on optical and wireless networks. He received the Best Student Paper Award at the IEEE Canadian Conference in Electrical and Computer Engineering 2009. He was a co-recipient of the Best Paper Award at IEEE/ACM/IFIP CNSM 2017 and 2019.

Mahdi Hemmati (M) received his MSc degree in Electrical Engineering - Systems & Control from Sharif University of Technology, Iran, in 2003 and the PhD degree in Electrical and Computer Engineering from the University of Ottawa in 2017. He was a recipient of the NSERC Postgraduate Scholarship during his doctoral studies. He is currently a Senior Staff Engineer at Huawei Technologies Canada Research Center, Ottawa. His research interests include distributed control and optimization, fair resource allocation, and reinforcement learning applications in network automation. He was a co-recipient of the Best Paper Award at IEEE/ACM/IFIP CNSM 2019.