Dynamic Optical Networks as Arcade Games: Lessons Learnt and Next Steps

Alejandra Beghelli[1,*] and Moshe Simon,[1]

[1] Optical Networks Group, Electronic & Electrical Engineering Department, University College London, Malet Place, London, WC1E 7JE, UK

[*]alejandra.beghelli@ucl.ac.uk

Abstract: We transformed the dynamic routing and spectrum allocation problem into an Arcade-like game and trained a reinforcement learning agent to play it. Preliminary results show the agent outperforms the K-SP-FF heuristic. © 2023 The Author(s)

1. Introduction

One of the most impressive achievements of artificial intelligence in the last years has been the super-human performance of reinforcement learning agents when playing Arcade games [1]. ‘Super-human’ means the agents found game strategies never tried by humans before to achieve high scores in those games. Such strategies could be identified thanks to the visual information provided by the game.

Inspired by these results, we wondered **a)** whether a reinforcement learning agent could also achieve super-human results (i.e. outperform human-designed heuristics) when solving dynamic spectrum allocation problems transformed into Arcade games and **b)** whether we could identify visually the agent strategy to transfer it into a networking environment. Although deep reinforcement learning (DRL) agents have been explored since 2018 to solve different resource allocation in optical networks [2], they have obtained mixed results - not always outperforming heuristics - and the way they are usually implemented (no graphic visualisation) does not lend itself to understanding their behaviour easily.

In this paper we present the first stages of our journey exploring the potential of transforming the operation of a dynamic elastic optical network into an Arcade-like game, to then train a reinforcement learning agent to play it. Preliminary results show the feasibility of the approach, with an agent getting good results when solving the RSA (routing and Spectrum Allocation) problem in a small network. These results might open the door for using Arcade-like games to solve a variety of resource allocation problems in optical networks. However, there is still much research needed to identify the best way of exploiting visual games for networking problems.

2. The Game (the DRL Environment)

The game was built with a specific aim: discovering a potential novel strategy to solve dynamic resource allocation problems in optical networks. Hence, the visual information presented to the player and the actions allowed to the player were simplified considering the game will be ultimately played by a DRL agent, not a human. The library PyGame was selected to implement the game.

2.1. Game mechanics

In each game round, connection establishment requests are presented to the agent. A round ends when X requests have been blocked (i.e. X unsuccessful agent actions). A request is defined by the triplet (src, dst, s), where src and dst are the source and the destination nodes, respectively, and s is the number of requested slots.

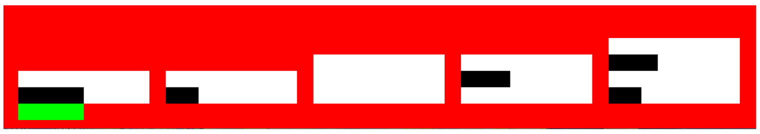

Fig. 1: Screenshot of the game screen.

For each request, K pre-computed routes between the source and destination nodes are displayed on the screen. Figure 1 shows the 920 x 150 screen used for the game, for K=5. Each route is visually represented as a matrix of $L_{src,dst,k} \times S$ squares, where $L_{src,dst,k}$ is the number of links composing the k-th pre-computed route between nodes src and dst, and S is the number of slots in each network link (equal to 8 in the example of Figure 1). Each spectrum slot is represented by a square. White and black squares represent available and used spectrum slots, respectively. The slots requested by the new connection are represented by the green squares at the bottom of the screen.

To select the route and a set of contiguous slots along the route, the player needs to move the green block by selecting the right and left arrows. This will cause the block of green squares to move along the screen to visualise the agent's strategy. Once the green squares are in the intended position, the player must press the key Enter.

2.2. *Game Score (Reward)*

Moving the arrows does not earn any points for the player. Score (reward) is given only after pressing the key Enter. If the position of the green squares (selected by the player by pressing left and right arrows) at the moment of pressing the Enter key is such that allows establishing the connection - that is, the block of green squares is aligned with a blocks of white squares in every link of the route - a positive score (reward) of 10 is given to the player. Otherwise, the player receives a negative score of -10.

2.3. *Observation*

A 3-dimensional array containing the RGB pixel information of the 920 x 150 screen is sent back to the agent once the action is processed. Each element is an integer number in the range [0,255].

3. The Player (the DRL Agent)

The player was implemented in Stable Baselines 3 as a Deep Q-Network agent using a convolutional neural network, the same agent used to achieve super-human performance in Atari games [1].

3.1. *Action trigger and action space*

The event that triggers an action by the agent is the arrival of a connection request (src, dst, s). The possible actions taken by the agent are 3, namely: "left arrow", "right arrow" and "Enter". The arrows allow the agent to select the position of the block of green squares (route and block of spectrum slots).

4. Preliminary Results

4.1. *Training the agent*

The agent was initially trained during 1 million timesteps in the 6-node topology shown in Figure 2, equipped with 8 spectrum slots per link.

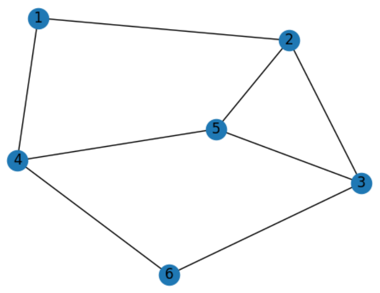

Fig. 2: 6-node topology

Hyperparameter	Value
gamma	0.999
learning rate	$2.5 \cdot 10^{-4}$
batch size	32
buffer size	$1.0 \cdot 10^{5}$
exploration fraction	0.5
eps end	0.1
target update interval	$1.0 \cdot 10^{4}$
train frequency	1
learning starts	$5.0 \cdot 10^{3}$
timesteps	$5.00 \cdot 10^{5}$

Table 1: Optimised hyperparameters

Weights & Biases [3] was used to optimise the parameters of the agent. The final parameters used are shown in Table 1. The number of rejected requests considered to end a round was set to X=3 and the maximum number of pre-calculated routes was set to K=5. Figure 3 shows the score achieved by the optimised agent over 100 game rounds against the performance of the 5-SP-FF heuristic. The agent achieved an average score of 31.9, slightly outperforming the heuristic that achieved an average score of 30.9. The agent outperformed the heuristic in 41 out of 100 game rounds, 9 of which were by a margin of 50 points or higher.

Although promising, these results are far from conclusive. First, although we could visualise the movements of the agent while attempting to allocate resources, the strategy applied was not clear and many times the agent would

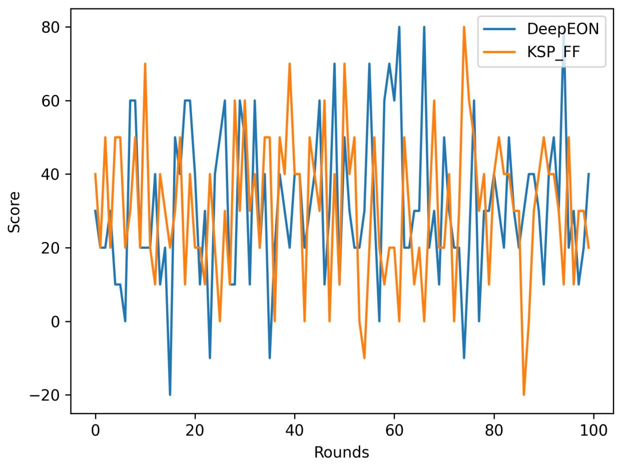

Fig. 3: Screenshot of the game screen.

remain oscillating, moving between left and right. A more detailed analysis of the actions taken by the agent is required. Second, the topology studied is very simple to draw general conclusions. Further study is needed with different topologies and larger capacities. Finally, from these preliminary results we have learnt to build simple games to represent the operation of an optical network and have an agent successfully playing such a game. This work has also allowed us to identify several lines of future research:

- **Impact of the visual information.** We have not investigated how to optimise the way the visual information is presented on screen. Aspects such as screen size, as well as size, location and colour of screen's elements have not been studied.
- **Action space.** Using arrows to select the position of the new request might hamper learning for routes on the rightmost part of the screen. For example, to get to the 5-th route, the agent should select the action "move right" at least 33 times. Using arrows is useful for us - humans- to distinguish the way the agent makes decisions. However, that very strategy might also hamper agent's performance. We will explore performance improvement by changing the action space. For example, by letting the agent to directly select the final position of the block of green squares on the screen.
- **Divide to conquer**. In our current implementation, two different goals (route selection and spectrum selection) must be achieved using the same set of actions. Most Arcade games - the inspiration for this research - have a unique goal. Hence, some aspects of the game could be modified to reflect this double-goal nature. This could be done either by designing a new reward system, re-organising the elements on the screen, by having the player to solve only one problem and have a heuristic to solve the other one or by using a two-agent system (one in charge of the routing problem and another in charge of the spectrum allocation problem) where the RSA problem is solved by agent's cooperation.
- **Train on adversity, test on prosperity**. Some research has shown that agents trained in adverse conditions (e.g. limited network capacity), achieve better performance than agents trained in the same "prosperity" test conditions (increased network capacity) [4]. We will explore whether this strategy will help improve the player's performance.

Acknowledgements. Financial support from EPSRC Programme Grant TRANSNET (EP/R035342/1) is gratefully acknowledged.

References

1. V. Mnih, K. Kavukcuoglu, D. Silver et al. "Human-level control through deep reinforcement learning, " Nature **518**, 529 –533 (2015).
2. T. Panayiotou, M. Michalopoulou and G. Ellinas, "Survey on machine learning for traffic-driven service provisioning in optical networks", arXiv: 2209.05080v1, https://arxiv.org/pdf/2209.05080.pdf (2022)
3. Weights & Biases https://wandb.ai/site
4. J. W. Nevin et al., "Techniques for applying reinforcement learning to routing and wavelength assignment problems in optical fiber communication networks," J. Opt. Commun. Netw. **14**, 733–748 (2022)