Hyper-parameter Optimization of Reinforcement Learning in Wireless Access Point Selection Problem

Yared Zerihun and Young-June Choi
Department of Artificial Intelligence
Ajou University
Suwon, Korea

***Abstract*—Beyond 5G (B5G) mobile nodes have several access options to choose from when they perform random access for the purpose of cell attachment. They can receive a pilot signal, (Received Signal Received Power (RSRP)) from several cells because of the proximal small and macro cell deployment scenarios in B5G networks. The available cells have varying levels of congestion. A reinforcement learning method allows the nodes to estimate the congestion level according to the delay experience in previous time slots, and select the appropriate access point. In this paper we propose an optimized algorithm for the reinforcement learning method, and compare its performance against the baseline approach.**

***Index Terms*—Machine Learning, Random-Access**

I. Introduction

In this paper, we focus on the scenario where coverage of cells overlap for mobile users at different spatial and time positions. The coverage areas of the access points (AP) are overlapping if not proximal to one other. For such a network, users are presented with an opportunity to route their requests through the best available access option. In 3GPP standards, the first procedure nodes should perform to request access with any selected AP is the Random Access (RA).

In the procedure to perform RA, nodes compete for seizing a RA preamble and the user that successfully win over the others is selected for sending/receiving data [1], [2]. The current random access point (RAP) selection on the standard is based on an RSRP measurement. Nodes greedily select the AP with strongest pilot signal power. They have no knowledge of the current congestion level. A higher RSRP AP, but a congested one leads to a RA delay that penalizes some of the stringent latency requirement traffic of B5G. Therefore powerful reinforcement learning algorithm such as *DQN* can be applied to solve the problem. Nodes should first rank the AP options according to a weight function. Then, the weight function is approximated by the deep-reinforcement learning algorithm. It is employed because of the dynamic network setup, where labeled data is not available to approximate the delay function as typically done in supervised learning algorithms [3]. The algorithm helps the node to leverage both the RSRP and predicted delay measurement to avoid congestion.

However, the RL algorithm, as with any typical supervised learning algorithm is very sensitive to its architecture and the hyper parameters configuration [4]. Neural Architecture Search (NAS) is a procedure in machine learning which deals with machine learning architectures and their performances [5]. However, most of the NAS methods cover the supervised learning algorithms. In light of this, the goal of this paper is to test which configurations from the neural architecture and hyper parameter space allow a performance increase for the RL algorithm that solves the RAP network selection problem.

II. Related-Work

Deep learning combined with reinforcement learning such as Q has revealed superior performance in an environment where transition probabilities are not easy to compute [6]. The RAPS environment is a stochastic one as such. Selecting a neural network architecture that can better approximate the Q as in the proposed DQN is not a trivial task. First, Neural Architecture Search (NAS) is a step towards finding that goal. However, NAS is a relatively new research topic. On top of that, most NAS studies revolve around finding good architectures and hyper parameter optimization (HPO) for tasks that have labeled data sets. For example, the CIFAR data set [7]. However interesting methodologies as preliminary studies can be learnt to propose a better neural architecture and HPO for the reinforcement learning task, which is dominant in wireless networks.

III. Problem Statement

For the optimization of the deep RL algorithm that solves the RAP problem, we define a neural architecture search space set, $N = \{N_1, N_2, ..., N_n\}$ consisting of the different architectures, and a hyper-parameter search space $H = \{H_1, H_2, ..., H_n\}$ considered for performance estimation test. Subset of H is associated with subset of N. We mathematically formulate the joint neural architecture search and hyper parameter optimization problem as two sub-problems. Let $n*$ represent the optimal neural architecture that is optimized. The problem of minimizing the neural architecture space with a set of initial hyper parameters, is formulated as:

$$n^* = \arg\max_{n \in N} R(D(n, h, \theta)), \tag{1}$$

where, h is an initial subset of H, D is the DQN algorithm, and R is a reward function detailed in section IV. The HPO optimization also follows mathematically as:

$$h^* = \arg\max_{h \in H} R(D(n^*(h)). \tag{2}$$

IV. Solution

The algorithm that solves the problem set defined in equations (1) and (2) is given in Algorithm 1 below. We first maintain a map that contains the architecture space with its selected set of hyper parameter configurations. An initial rank is assigned to the solution and updated later in the procedure. Then, we search for an appropriate neural architecture, and then we optimize H by applying the architecture selected in the former step. The appropriate solution is the one returning the highest rank. The higher the reward function the higher the rank of a specific configuration. The reward function R allows

Algorithm 1

```
Get the NA list
Get the HP list
rank = (NAS, HPO)
while (NA space is not explored) do
    while (HP space is not explored) do
        Calculate the reward according to equation 3
        Update rank according to the reward obtained
    end while
end while
return max_NAS (rank)
```

for a performance estimation function. It helps to relatively compare the baseline algorithm and the algorithm we proposed in this section. It is given by the following equation

$$R = 1/(D_e - D_s), \tag{3}$$

where D_e and D_s are the node's start time and end time of the random access procedure, respectively. Then a cumulative expectation of the reward from the *DQN* algorithm is given by,

$$E[R] = \sum_{i=0}^{\infty} (\gamma_i S_i | S = S_0), \tag{4}$$

where $\gamma < 1$, and S_i, the i^th state that will return a reward R.

V. Experiment and Result Analysis

We ran a simulation to test the performance of the algorithm. The simulation is run for a number of episodes. Iteration values are plotted for every other episode. The reward graph is plotted for a number of training iterations as Figure 1 illustrates. We can observe that Algorithm 1 reveals a reward increase after a number of iterations. There is a performance penalty during the first few iterations, and even though the reward at some points approaches the baseline *DQN* algorithm, we can report a promising average increase in performance. With regard to the convergence, we can observe that both algorithms have a comparable stability.

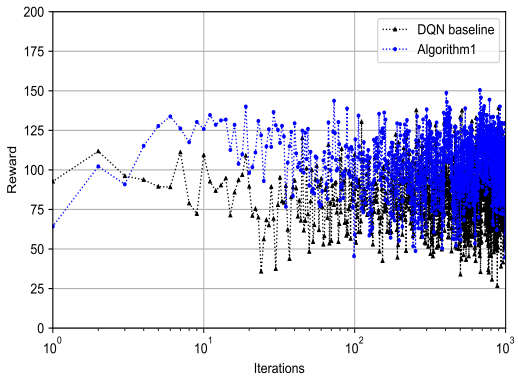

Fig. 1. Reward increases after initial training penalty for the optimized DQN.

VI. Conclusion

This paper presents an algorithm which maximizes the reward function of the reinforcement learning *DQN* algorithm that solves the wireless RAP selection problem. Since the environments that such learning algorithms solve is highly dynamic, and stochastic in nature, an even better optimal algorithm which reduces the time complexity of finding the hyper parameter configurations is still left as our future work.

References

[1] 3GPP Technical Specification, TS 36.321. Evolved Universal Terrestrial Radio Access (E-UTRA); Medium Access Control (MAC) Protocol Specification.

[2] 3GPP TS 36.211. Evolved Universal Terrestrial Radio Access (E-UTRA); Physical Channels and Modulation.

[3] Caruana R, Niculescu-Mizil A. An empirical comparison of supervised learning algorithms.

[4] InProceedings of the 23rd international conference on Machine learning 2006 Jun 25 (pp. 161-168).

[5] Falkner S, Klein A, Hutter F. BOHB: Robust and efficient hyperparameter optimization at scale. arXiv preprint arXiv:1807.01774. 2018 Jul 4. Wistuba M, Rawat A, Pedapati T. A survey on neural architecture search. arXiv preprint arXiv:1905.01392. 2019 May 4.

[6] Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G, Petersen S. Human-level control through deep reinforcement learning. nature. 2015 Feb;518(7540):529-33.

[7] Domhan T, Springenberg JT, Hutter F. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. InTwenty-fourth international joint conference on artificial intelligence 2015 Jun 27.