7 APPENDIX

Sections 7.1-7.4 elaborate on further improvements, and section 7.5 goes into detail about initial choice of analyzed sets of hyperparameter values.

7.1 ALTERNATIVE VERSION OF ϵ-GREEDY POLICY

The alternative version of ϵ-greedy suggests assigning $1-\epsilon$ probability to the best-estimated action, and distributes probability of ϵ among all the actions, including the best one (unlike considered version of the policy). Thus, in formula it is expressed as the following:

$$\pi_{\epsilon-greedy-alt}(a) = \begin{cases} 1-\epsilon+\frac{\epsilon}{|A|}, & \text{if } a = \arg\max_{b \in A} Q(b) \\ \frac{\epsilon}{|A|}, & \text{otherwise} \end{cases} \tag{7}$$

Even though previous version might have been more understandable (as now probability of choosing best action is not so intuitively clear), the alternative policy is more mathematically logical. This is because usual choice of epsilon and the algorithm itself imply clear preference to exploitation. In other words, we expect best action so far to have higher probability of being chosen compared to all other actions.
We see that it is true for $\pi_{\epsilon-greedy-alt}(a)$, because $1-\epsilon \geq 0$. However, this feature does not always occur in $\pi_{\epsilon-greedy}(a)$ version (especially if ϵ is comparatively large). For instance, this can be seen from the following counterexample: $\epsilon = 0.95$ (and $|A| = 10$, as we considered before): $1-0.95 = 0.05 < 0.11 \approx \frac{0.95}{9}$. Although, ϵ can be adjusted for that, so the original variant is still valid. Nonetheless, this property makes the alternative provided policy more mathematically strict.

7.2 ALTERNATIVE COMPARISON MEASUREMENTS

Average total reward given by equation (6) in section 5.1 is a solid measurement of overall performance of algorithm with a certain fixed parameter, however it counts initial exploration leap. And it might be the case that we are not interested in how long exploration takes at the beginning, but our goal is higher intermediate rewards in the end of the curve. Thus, we propose 2 methods to consider end rewards more than starting ones:

1. Final average reward. By taking last reward, or an average over last period of rewards (e.g. over 5% of time steps), we neglect the initial exploration period completely, and are interested only in the latter rewards;
2. Discounted total average reward. We take discounted cumulative reward and average it over repetitions. Thus, we value latter rewards more than initial ones, making fast leap at the beginning less necessary to get better performance measure.

7.3 BETTER PARAMETER VALUES (DENSER DISTRIBUTION)

With more values for hyperparameters in between their minimum and maximum options (described in more details in Appendix 7.5), it would be possible to obtain better lower and higher boundaries. Besides that, further inference on relations between parameter value and correspondent curves could be conducted.
It would also improve comparison plots between the algorithms. For instance, on Figure 4.1 it seems that UCB broken line is significantly better than of OI (because UCB has high rewards for most of it parameter settings and it's lowest point is 0.82, while, in contrast OI's lowest point is 0.73 - lowest overall). Although, this can be as a result of the chosen parameter settings. Firstly, we considered more experimental values of c than init_val. And secondly, this depends on how extreme the boundaries are - it is likely that if minimum init_val was higher, the OI broken line could be comparable with UCB.

7.4 PERCENTAGE OF OPTIMAL ACTION PLOT

Last suggested improvement included in Appendix is a learning plot with response variable being percentage of optimal action (theretical best), instead of reward. This would allow to see the true

performance of each algorithm and hyperparameter value. Thus, discovering the magnitude of the difference between the algorithms. Meaning, if the true best immediate reward, averaged over repetitions, would be much higher, this would indicate that both algorithm perform very poorly and there is no significant advantage of choosing one over another. This, fortunately, is not the case, as maximum theoretical immediate reward is 1.0 and all algorithms approach it quite closely. However, averaging the reward over repetitions and showing it on the plot would provide some useful additional information.

7.5 Choices of tested hyperparameter values

In this section we show why taken boundaries of paramater values are sufficient for our conclusions in the above sections.

7.5.1 Epsilon

From Figure 1 we can already see confirmation of good choice of the parameter settings. Recall 2 types of ignored ϵ values from section 2.1. 1) With $\epsilon = 0.01$ we clearly see that exploration is needed, because the curve increases too slowly, and greedy algorithm would act even more poorly. 2) And now let us consider $\epsilon = 0.25$. Even though at the beginning it achieves fastest leap (because it explores the best action faster), later it chooses worse actions and at the end performs the worst of all settings. Going for even higher value of ϵ would decrease the asymptote of the curve, because percentage of exploitation of the best action would be even lower.

7.5.2 Initial values

Regarding the minimum taken value, we can already see that it is underestimated, as on Figure 2.0 its curves do not reach the rewards achieved by other settings, throughout entire timeline. Therefore, the starting point is already below the one the algorithm moves towards. For the current upper boundary, we see that it takes too many time steps for the algorithm to explore until it finds best action. Increasing it further would only make this number of steps bigger, thus decreasing performance (because the final rewards will be similar to those init_val used already).

7.5.3 Exploration constant

As can be seen on Figure 3.1, $c = 0.01$ and $c = 0.05$ have the same leap at the start of the curve, as median values of c, but later achieve worse highest rewards on average. As explained in section 4.2, this is because these algorithms in some of repetitions started exploiting wrong action, thus exploration is indeed needed. While bigger c values, as 1.0, explore too much and do not exploit enough for the true best action be figured out in 1000 time steps, which is accompanied by overall lower curves on Figures 3.1 and 3.2.