Technical University of Crete
School of Electrical and Computer Engineering
MSc Program: **MLDS**
Course: **Reinforcement Learning**
Instructor: T. Spyropoulos
Student **Chivintar Amenty**

Reducing Opinion Polarization over a Social Network
Phase II

Report Delivery Date: July 3, 2025

This document describes the second and final part of the Reducing Opinion Polarization over a Social Network project. It is shown that for the toy networks of Phase I, the DQN Agent is successful in under a minute. Next, considering the same environment, multiple graphs (Echo Chambers and Random) are examined with DQN. Next, some further experiments are being discussed. Finally, the report is concluded with some final remarks.

Environment Setting

The influence model of Phase I is slightly adapted to facilitate investigation for graphs with opinions [-1, 1] with a step of 0.01 . The changes are with regards to the terminal conditions and the reward structure. The stochasticity rules remained the same. In addition, the code was refactored to include Python classes, methods and functions, instead of only Python functions as in Phase I. This enabled even more freedom in the various experiments.

Environment Class: For graph with N nodes, the state space is the N-dimensional vector of individual opinions. The action space has a size equal to the size of the bidirectional edges, that is, two times the number of undirected edges. The reward structure was changed to be the difference of polarization after each step. As a bonus, a small reward is added when this difference is negative. Terminal conditions were relaxed slightly for the continuous case only, including states that have total polarization ≤ 0.1 (various values were examined).

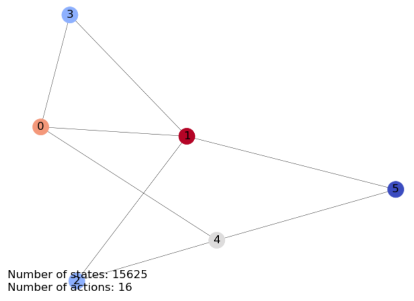
Agent Class: The vanilla DQN algorithm was chosen[1], with a replay buffer of memory capacity of 2000 records. The neural network was chosen to be a two-layer fully connected,

[1]Mnih et al., Playing Atari with Deep Reinforcement Learning, arXiv:1312.5602, 2013. `https://arxiv.org/abs/1312.5602`

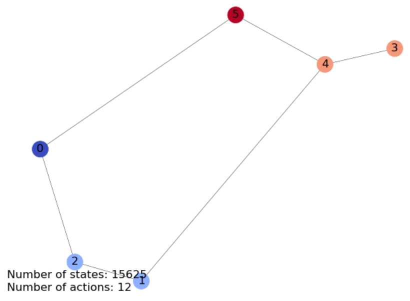
with input size the size of the state vector, and output size equal to the `action_size` (as the agent would see many different graphs during training). Of course the output is the Q value, where the agent is choosing the `argmax`, while doing so with a factor of exploration ϵ which is initially set to 1 and decays exponentially/non linearly??? to the minimum value 0.01.

Toy Networks of Phase I

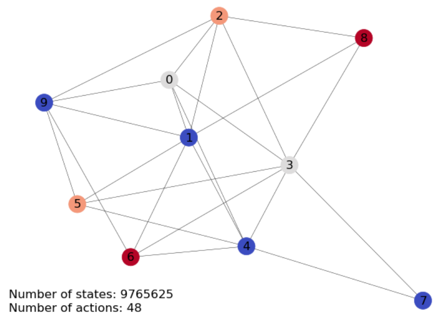
We begin by revisiting the toy networks of Phase I (Figure 1). Value Iteration in Phase I needed ≈1.2 seconds for 4-node networks, ≈20s for 6-node networks, ≈45 minutes for 8-node networks, while needed more than 90 minutes for 10-node networks. In Figure 2, a comparison of the two algorithms for an 8-node echo-chamber network is shown. Notably, DQN was able to solve it in under 30 seconds for 100% of the evaluation episodes.



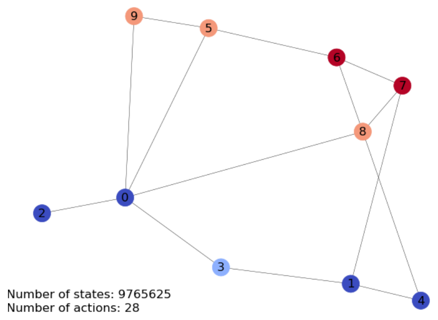
(a) graph G1-6



(b) graph G2-6



(c) graph G1-10



(d) graph G2-10

Figure 1: Toy networks of Phase I. Left: random nets. Right: Echo Chamber nets. V.I. for $N \leq 4$ needed a couple of seconds, while DQN for the same nets needed ≈13 seconds (more plots for Phase I can be found in the `.ipynb` file).

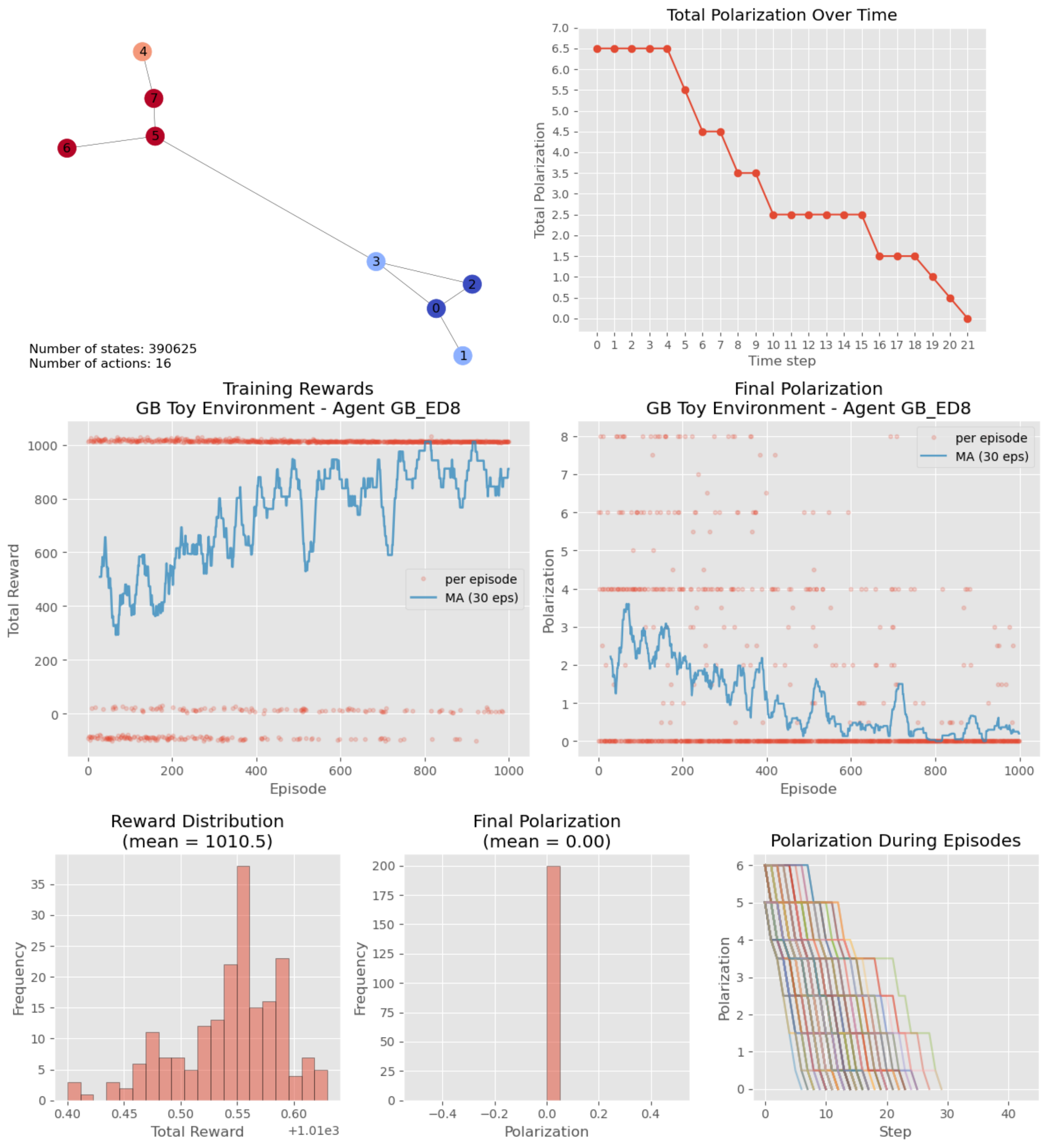
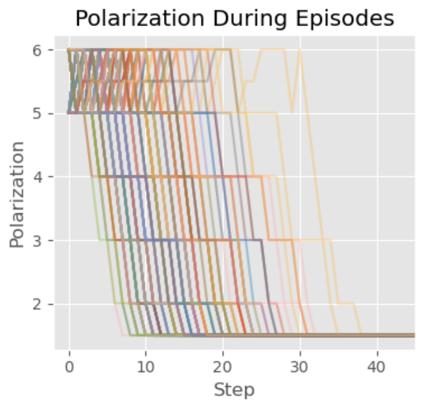


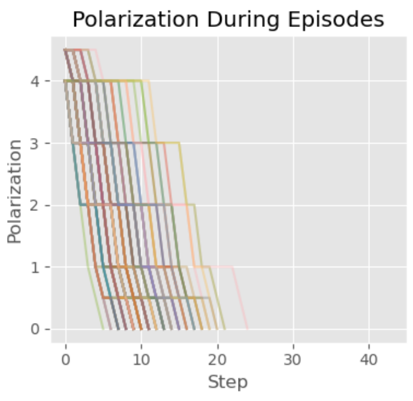
Figure 2: A (difficult) echo chamber toy graph from phase I. V.I. (top right) needed at least 45 minutes. DQN (middle: training time, bottom: evaluation episodes) solved it in 30 seconds.

DQN in Partial Knowledge

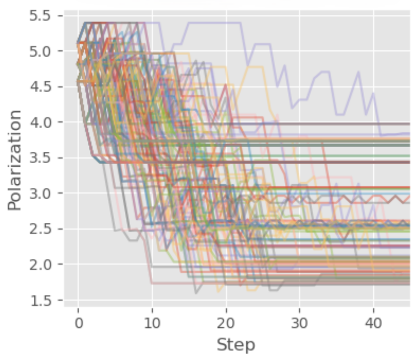
In this section more involved experiments are examined. Here, the DQN Agent is being trained on random and echo-chamber graphs, for discrete opinions of step 0.5, (5 opinion levels) and of step 0.01 (201 opinion levels). In this part, the nets examined are (re-)created randomly if the parameter `regenerate_graph` is set to `True`, otherwise, while resetting the environment, the graph takes the same initial state. This option is available both for training and evaluation, however, setting this parameter to `True` is meaningful when the agent has more information regarding the graph.



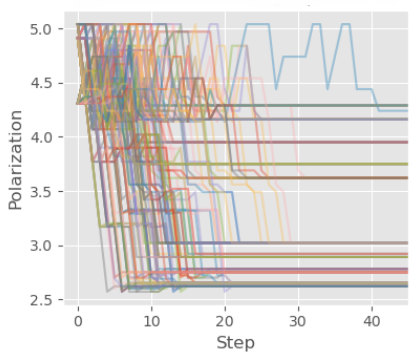
(a) Rand. Discrete 10 nodes: avg of 14.5 steps for half polarization



(b) E.C. Discrete N=10: avg of 6.7 steps for half polarization



(c) Rand. Continuous N=10: avg of 20.4 steps for half polarization



(d) E.C. Continuous N=10: nan

Figure 3: Experiments on various graphs. The plots show stepwise polarization of the last 20 evaluation episodes. In the logging we can also find average steps needed for each to reduce the polarization of the graph 10%, 50% and 90%. nan values appear when the agent did not manage to reach aforementioned polarizations.

A couple experiments on bigger graphs (N=40) were also done (Section N = 40 in `.ipynb`). The conclusion is that the terminating conditions and the reward structure must be crafted again, in addition to that the number of maximum steps must be increased. Also, for such bigger graphs, significantly higher number of episodes are needed to see meaningful results, as well as the replay buffer to be more stable over episodes; in the current setting the target network is updating far too quickly.

DQN in Full Knowledge

Evaluating the DQN Agent for different graphs while being trained on only one, or training the agent for different graphs and evaluating for either one or more, was also examined.

Overall, for random networks DQN showed higher difficulty, terminating at the maximum steps threshold significantly more times than in echo-chambers. This is expected, as the thing that the Agent can generalize upon, is only that in echo-chamber graphs, the first half of the nodes exist in the first chamber, and the second half of the nodes exist in the second chamber. To derive more meaningful results, we must inject more information to the agent. That was not implemented to-date, but will be soon be available in the public GitHub repository [2] .

Conclusion

Here, this Reinforcement Learning project as part of the MLDS MSc RL course is finally concluded. In short, I implemented a Deep Q-Network agent capable of reducing opinion polarization in social networks. Our Agent outperformed the traditional value iteration methods in terms of runtime while also achieving comparable, if not better, depolarization. The results show the importance of the reward disign and structure, as well as the design of the environment, especially for larger graphs. Echo chamber structures appeared easier to depolarize than random ones, most likely due to their symmetry in opinion assignment. While partial knowledge scenarios proved more challenging, the agent still generalized reasonably. Future work incorporates graph-aware architectures and/or richer state representations for better generalizations and more rewards.

[2]`github.com/chivintar/reducing-polarization`

AI DECLARATION: Commercial LLM was used for assistance in part for developing of the code, with all prompts specifically focusing on my complete understanding. All lines are written by me, with specific external references inline.