Deep reinforcement learning for active hypothesis testing with heterogeneous agents and cost constraints (Supplementary material)

George Stamatelis, Nicholas Kalouptsidis

In this supplementary material, we will further compare our algorithms in synthetic data and data originating from an actual dataset. The number of processes and the communication graphs remain the same as the main body. The only thing that changes is the observation generating process. Due to limited resources we will only consider unconstrained problems and we will decrease the training episodes to 25000.

We will evaluate the decentralised PPO with a joint reward, the decentralised PPO with a global critic and the centralised PPO on the same environment as the main text, but with different observation probabilities. As benchmark we will consider a single PPO agent with acess to all sensors.

I. Gaussian data

A sensor model often considered in recent literature e.g. [1] is the following

$$y_t \sim \begin{cases} N(0,\sigma^2), & \text{if corresp. proc is normal} \\ N(1,\sigma^2), & \text{else} \end{cases} \tag{1}$$

Similarly to [1] we set $\sigma = 0.5$. The results can be seen in tables I,II,III.

TABLE I: Accuracy and average stopping time, out of 10000 episodes for the first decentralised problem with Gaussian observations

(a) accuracy

algorithm	$\epsilon = 0.05$	$\epsilon = 0.1$
single Agent PPO	0.979	0.973
centralised PPO	0.962	0.955
decentralised PPO with joint reward	0.951	0.937
PPO with global critic	0.969	0.963

(b) average stopping time

algorithm	$\epsilon = 0.05$	$\epsilon = 0.1$
single Agent PPO	17.371	14.349
centralised PPO	8.141	7.789
decentralised PPO with joint reward	7.673	6.961
PPO with global critic	7.24	6.227

TABLE II: Accuracy and average stopping time, out of 10000 episodes for the second decentralised problem with Gaussian observations

(a) accuracy

algorithm	$\epsilon = 0.05$	$\epsilon = 0.1$
single Agent PPO	0.979	0.973
centralised PPO	0.962	0.955
decentralised PPO with joint reward	0.978-1.0-1.0	0.978-0.999-0.999
PPO with global critic	0.985-0.999-0.999	0.974-0.982-0.982

(b) average stopping time

algorithm	$\epsilon = 0.05$	$\epsilon = 0.1$
single Agent PPO	17.371	14.349
centralised PPO	8.141	7.789
decentralised PPO with joint reward	12.211	10.8
PPO with global critic	9.329	8.43

TABLE III: Accuracy and average stopping time, out of 10000 episodes for the third decentralised problem with Gaussian observations

(a) accuracy

algorithm	$\epsilon = 0.05$	$\epsilon = 0.1$
single Agent PPO	0.979	0.973
centralised PPO	0.962	0.955
decentralised PPO with joint reward	0.977-0.999-0.999	0.967-0.981-0.981
PPO with global critic	0.991-0.994-0.994	0.974-0.98-0.98

(b) average stopping time

algorithm	$\epsilon = 0.05$	$\epsilon = 0.1$
single Agent PPO	17.371	14.349
centralised PPO	8.141	7.789
decentralised PPO with joint reward	12.392	11.283
PPO with global critic	10.526	9.392

The main takeaways of our experiments are the following

1) Decentralised PPO with joint reward and PPO with global critic perform better than the centralised algorithm in the fully connected setting.
2) Decentralised PPO with a global critic performs slightly better than the fully decentralised algorithm in all communication graphs.
3) All multiagent algorithms achieve a significant shorter stopping times compared to the single agent.

II. ToN IOT windows10 data

We will now evaluate our algorithms in an example closer to reality, using an cybersecurity dataset. While this example is not entirely realistic and our goal is not to build a state of the art threat detection system, we think it is valuable to evaluate our algorithms in an example closer to reality. A lot of papers on DRL for AHT only examine their algorithms on synthetic data. Machine learning practitioners almost always encounter significant discrepancies between training environments and deployment.

We used the windows 10 dataset from to Ton IOT data [2] [3] [4] [5] [6] [7] [8] [9]. Each row of the dataset contains information about one process. It has a lot of information about the processor activity, the network activity, the memory activity, the disk activity e.t.c. Each process has a binary label about wether it is abnormal or not and a second label describing the type of attack, which we ignore. We selected the following five features

1) 'Memory Demand Zero Faults sec'
2) 'Network_I(Intel R _82574L_GNC)Bytes Received sec'
3) 'Network_I(Intel R _82574L_GNC) Bytes Sent sec'
4) 'Process_Page Faults_sec'
5) 'Memory Page Writes sec'

Using these features we will train a decision tree that performs intrusion classification. More specifically, we split the dataset into a train set, a validation set and a test set. The train set contains 70 % of the data and the validation set 10% of the data. We used the sklearn [reference] implementation of decision trees setting class_weight='balanced' and a standard scaler.

Our process can be described in the following steps

1) Train the decision tree on the train set
2) Estimate the conditional probabilities $P[Y|\text{process is abnormal/normal}]$ on the validation set. It turns out that $P[Y=1|\text{process is abnormal}] \approx 0.842$ and $P[Y=1|\text{process is normal}] \approx 0.058$.
3) Use the conditional probabilities to create an artificial simulation training environment and train the agents.
4) Test the algorithms using the test sets as described bellow.

A. Testing procedure

The system state (correct hypothesis) is sampled according to the prior. Then at each time instance each agent probes on of the sensors. We sample a process from the test set which is either labeled normal or abnormal depending on the hypothesis and the decision tree outputs a label y. That label is used to update the belief using the conditional probabilities estimated on the validation set. The rest of the problem remains the same as the environments with the synthetic data.

B. Results

TABLE IV: Accuracy and average stopping time, out of 1000 for the first decentralised problem with intrusion detection data

(a) accuracy

algorithm	$\epsilon = 0.05$	$\epsilon = 0.1$
single Agent PPO	0.998	0.989
centralised PPO	0.995	0.998
decentralised PPO with joint reward	0.999	0.992
PPO with global critic	0.999	0.998

(b) average stopping time

algorithm	$\epsilon = 0.05$	$\epsilon = 0.1$
single Agent PPO	14.951	11.878
centralised PPO	6.399	7.611
decentralised PPO with joint reward	6.418	7.172
PPO with global critic	5.88	6.001

TABLE V: Accuracy and average stopping time, out of 1000 for the second decentralised problem with intrusion detection data

(a) accuracy

algorithm	$\epsilon = 0.05$	$\epsilon = 0.1$
single Agent PPO	0.999	0.992
centralised PPO	0.999	0.995
decentralised PPO with joint reward	0.999-1.0-1.0	0.995-0.996-0.996
PPO with global critic	0.999 -1.0- 1.0	0.98-0.999-0.999

(b) average stopping time

algorithm	$\epsilon = 0.05$	$\epsilon = 0.1$
single Agent PPO	14.951	11.878
centralised PPO	6.399	7.611
decentralised PPO with joint reward	10.073	8.323
PPO with global critic	9.008	7.611

TABLE VI: Accuracy and average stopping time, out of 1000 for the third decentralised problem with intrusion detection data

(a) accuracy

algorithm	$\epsilon = 0.05$	$\epsilon = 0.1$
single Agent PPO	0.999	0.992
centralised PPO	0.999	0.995
decentralised PPO with joint reward	0.999-1.0-1.0	0.994-0.996-0.996
PPO with global critic	1.0-1.0-1.0	0.992-0.995-0.995

(b) average stopping time

algorithm	$\epsilon = 0.05$	$\epsilon = 0.1$
single Agent PPO	14.951	11.878
centralised PPO	6.399	7.611
decentralised PPO with joint reward	10.897	9.458
PPO with global critic	9.704	8.878

The main takeaways are

1) Decentralised PPO with global critic achieves a shorter stopping time than its fully decentralised version.
2) For the fully connected setting both decentralised algorithms achieve a shorter stopping time than the global controller.
3) In any case, all multiagent algorithms are faster than the single agent algorithm.

We reach more or less the same conclusions for all three observation models considered.

References

[1] C. Zhong, M. C. Gursoy, and S. Velipasalar, "Controlled sensing and anomaly detection via soft actor-critic reinforcement learning," in *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4198–4202, 2022.

[2] N. Moustafa, "A new distributed architecture for evaluating ai-based security systems at the edge: Network ton_iot datasets," vol. 72, 05 2021.

[3] A. Alsaedi, N. Moustafa, Z. Tari, A. Mahmood, and A. Anwar, "Ton_iot telemetry dataset: A new generation dataset of iot and iiot for data-driven intrusion detection systems," *IEEE Access*, vol. 8, pp. 165130–165150, 2020.

[4] N. Moustafa, M. Ahmed, and S. Ahmed, "Data analytics-enabled intrusion detection: Evaluations of ton_iot linux datasets," in *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pp. 727–735, 2020.

[5] N. Moustafa, "A systemic iot-fog-cloud architecture for big-data analytics and cyber security systems: A review of fog computing," 2019.

[6] N. Moustafa, M. Keshk, E. Debie, and H. Janicke, "Federated ton_iot windows datasets for evaluating ai-based security applications," 2020.

[7] T. Booij, I. Chiscop, E. Meeuwissen, N. Moustafa, and F. den Hartog, "Ton_iot: The role of heterogeneity and the need for standardization of features and attack types in iot network intrusion data sets," *IEEE Internet of Things Journal*, vol. PP, pp. 1–1, 05 2021.

[8] J. Ashraf, M. Keshk, N. Moustafa, M. Abdel-Basset, H. Khurshid, A. D. Bakhshi, and R. R. Mostafa, “Iotbot-ids: A novel statistical learning-enabled botnet detection framework for protecting networks of smart cities,” *Sustainable Cities and Society*, vol. 72, p. 103041, 2021.

[9] N. Moustafa, “New generations of internet of things datasets for cybersecurity applications based machine learning: Ton_iot datasets,” *Proceedings of the eResearch Australasia Conference, Brisbane, Australia.*, 2019.