A Hybrid Federated Reinforcement Learning Approach for Networked Robots

Gayathri Rangu, Divya D. Kulkarni, Jayprakash S. Nair, and Shivashankar B. Nair

1 Introduction

Smartphones, tablets, and other web-enabled devices have emerged as indispensable tools for communication, information, and entertainment worldwide. Each one of these numerous devices produces data every second. This data is used to analyze and learn user patterns, enhancing user experience by sending appropriate recommendations. Conventional edge-cloud Machine Learning (ML) techniques require this humongous amount of data to be uploaded onto a server and are used to train the model centrally. This method suffers from privacy issues since user-sensitive data lands up at the server. McMahan et al. [1] have proposed a way out by using what they call federated learning (FL). In this method, the models are generated locally on each of the devices using the data within. These local models are then sent to the server where they are aggregated and returned to all the connected devices, thereby preserving privacy [2]. This centralized approach, however, suffers from various drawbacks such as single-node dependency, maintenance costs, and scalability issues. To overcome these, decentralized approaches of FL have been proposed [3] wherein all the devices are connected over a network and there is no global server that aggregates the models. In these approaches, the learned model weights are exchanged with all

G. Rangu (✉) · D. D. Kulkarni · S. B. Nair
Indian Institute of Technology Guwahati, Guwahati, India
e-mail: gayathri.rangu@iitg.ac.in

D. D. Kulkarni
e-mail: divyadk@iitg.ac.in

S. B. Nair
e-mail: sbnair@iitg.ac.in

J. S. Nair
Federal Institute of Science and Technology, Angamaly, India
e-mail: jsnair.hi@gmail.com

B. P. Swain and U. S. Dixit (eds.), *Recent Advances in Electrical and Electronic Engineering*, Lecture Notes in Electrical Engineering 1071,
https://doi.org/10.1007/978-981-99-4713-3_47

the other devices. This approach can circumvent the issues of centralized FL but at the cost of increased learning times.

Of late, FL has also been used in networked robotics [4]. Robots may inhabit the same or different environments. Every robot generates its dataset. In such cases, using FL can train models specific to a robot's environment. Time-critical devices such as robots, drones, and other IoT devices usually do not have a predefined dataset to be trained on. In tasks like exploration, obstacle avoidance, etc., where robots need to evolve on their own, most often, Reinforcement Learning (RL) [5] is used. RL focuses on how agents interact with their environment and learn to evolve in a trial-and-error manner. A policy is learned based on which the agent takes action and in turn, gets rewarded or penalized for the action taken. Overall, the agent focuses on maximizing the cumulative rewards in the long run. RL can also be implemented in a federated manner [6].

Federated Reinforcement Learning (FRL) can be implemented either as a client-server model or a peer-to-peer model. The former may not be always suitable in the case of robotics because when the server goes down, the whole network could fail. Although a peer-to-peer approach can handle this drawback, it is prone to delays in model updates. Since both centralized and decentralized approaches of FL have their own drawbacks, a combination of both these approaches could be explored. This paper presents such a Hybrid FRL (*HyFRL*) wherein the learned models are aggregated and shared among a set of connected robots inhabiting different environments. The approach uses mobile agents that move from robot to robot to aggregate and share the models they carry on the go. Unlike the decentralized approaches, [7] aggregation is performed, not just by a mobile agent but also by a server, thus making it partly centralized and partly decentralized.

The following are the significant contributions of the work presented in this paper:

i. A Hybrid Federated Reinforcement Learning (*HyFRL*) model which is a combination of both centralized FRL and decentralized FRL.
ii. A comparison of the performances of centralized, decentralized, and Hybrid FRL approaches.
iii. Use of multiple agents to hasten convergence.

2 Related Work

Zhou et al. [8] have proposed a real-time data processing architecture for multi-robots based on FL for recognition tasks using MNIST and CIFAR10 datasets for training. Agrawal et al. [3] described a decentralized FL approach using mobile agents using Deep Learning. In robotic applications where such datasets are not available *a priori*, the use of RL [9] can be a better option. In standard RL models such as Q-learning, an agent learns to take actions by continuously perceiving and interacting with its environment.

FRL is a relatively new research direction that has come up to address the problems in FL and RL. A method to decentralize FRL in a multi-robot scenario has been proposed in [7]. Though the need for a central server was eliminated, this model suffers from delayed updates. On a larger scale, when the number of nodes increases, a single mobile agent going around and sending updates results in slowing the entire system. This paper thus proposes a Hybrid of centralized and decentralized FRL to mitigate the issue.

3 Methodology

An *agent* is a piece of software residing in a device that can either be static or mobile. A *mobile agent* can carry information, program, or data as a payload from one device to another. Mobile agents, thus, can knit through connected devices, sharing information thereby keeping the models in these up to date. Agent platforms need to run on individual devices to facilitate the execution of agents. This work uses a Python-based variant of an open-source mobile agent platform, viz. *Tartarus* [10]. This platform allows features like agent programming, execution, mobility, etc. Mobile agents have been used to aggregate and share locally learned models in the network to realize Hybrid FRL (*HyFRL*), which uses both centralized and decentralized approaches.

A mobile agent, A_i, traverses a set of networked N robots, $\boldsymbol{R}$. Each robot, $R_i \in \boldsymbol{R}$, uses RL to churn out its local model. On visiting a robot, the mobile agent, A_i, aggregates the model which it carries as payload with the one locally available within R_i, to generate a new model, M_i. It then proceeds to the next connected robot, R_{i+1}, and performs the same aggregation to produce, M_{i+1}. By doing so, A_i, it tends to aggregate and share the models locally learned by the robots in the set, $\boldsymbol{R}$, achieving decentralized FRL as in [7]. In the proposed *HyFRL*, after every τ hops of such aggregation in the network, A_i sends the currently aggregated model it carries, to a central server which in turn aggregates the received model with the one it already has, and updates all the connected robots with this global aggregate. This *aggregation* and subsequent *updation* constitutes a *round* and augments the decentralized FRL with a centralized flavor.

4 Algorithm

Algorithm 1 depicts the three sub-parts of the proposed *HyFRL* algorithm, viz. *(a)* Robot side, *(b)* Mobile Agent side, and *(c)* Server side. The algorithm takes in the number of robots, N, the number of hops after which the updates are sent to the server, τ, and the Q-learning algorithm, and outputs the aggregated Q-table, Q_{agg}. The algorithm is made to run till the learning is saturated. On the robot side, the robots learn to avoid obstacles using Q-learning and update their respective Q-tables

based on Eq. (1):

$$Q_n(s, a) = Q(s, a) + \alpha * (R_n + (\gamma * Q_{\max}(s', a')) - Q(s, a)) \quad (1)$$

where $Q_n(s, a)$ and $Q(s, a)$ are the new and old Q-values of the state, s. $Q_{max}(s, a)$ is the maximum of the Q-values considering the current state of the agent (after performing the action a), R_n is the reward that the agent received by performing a in s, α is the learning rate, and γ is the discount factor. The agent goes through the episodes several times, each time adjusting its policy based on its newly acquired knowledge (*state*, *action*, *new state*, *reward*). As a human would learn how to play a game to get a good score, the agent eventually learns the policy that produces a good reward over the episode.

The robots also replace their Q-tables with the global updates, Q_g, as and when they receive them from the server. On the Mobile Agent side, after the creation of the agent within R_0, it carries Q_{agg}, as a payload to the next robot, R_{j+1} where it again performs a similar aggregation of Q_{agg} with the locally available R^Q_{j+1}. It carries this new aggregate to another robot and repeats this procedure. After every τ such migrations or hops, it sends the Q_{agg} to the server. At the server side, Q_{agg}s received from the mobile agent are aggregated and sent back to all the robots. It may be observed that while the server aids in centralized FL, the mobile agent facilitates a decentralized version. The three sub-parts thus, constitute the *HyFRL* approach.

Algorithm 1: The proposed *HyFRL* algorithm for a single agent.

Input: N, τ, Q-learning algorithm

Output: Learned Q-table, Q_g

(a) Robot Side:

```
1 for i = 0 to (N − 1) :
2     Initialize R_i^Q
3 while(True):
4     for i = 0 to (N − 1):
5         avoidObstacle()
6         update R_i^Q
7         if received(Q_g):
8             R_i^Q = Q_g
```

(b) Mobile Agent Side:

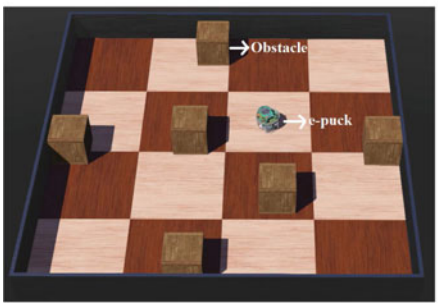
```
1 Q_agg = R_0^Q
2 while(True):
3     for i = 0 to (N − 1):
4         for j = 1 to τ:
5             Payload(Agent, Q_agg)
6             R_{j+1} ←move(Agent)
7             Q_agg=aggr(R_{j+1}^Q, Q_agg)
8         server← send(Q_agg)
```

(c) Server Side:

```
1 initialize Q_g
2 while(True):
3     receive(Q_agg)
4     Q_g = aggr(Q_g, Q_agg)
5     for i = 0 to (N − 1):
6         R_i ← send(Q_g)
```

5 Experiments and Results

We have used *Webots*® [11] and *Tartarus* platforms on computers connected via a LAN to create a network of robots that can facilitate agent mobility. Each *Webots*® platform had a robot performing obstacle avoidance using Q-learning which is a form



(a) A *Webots*® arena with an e-puck robot and blocks constituting the obstacles

State/Action	A0	A1	A2	A3
S0	-6.61	-0.83	-0.16	10.51
S1	-6.03	-0.29	0.55	11.05
S2	-5.18	0.32	1.19	12.07
S3	-4.61	0.95	1.82	12.83
S4	-4.12	1.65	2.31	12.56
S5	-4.47	2.17	2.81	14.23
S6	-2.49	2.43	3.65	15.33
S7	-1.81	3.46	4.55	15.91
S8	-1.32	2.95	4.33	17.01
S9	-7.53	-1.58	-0.57	9.84

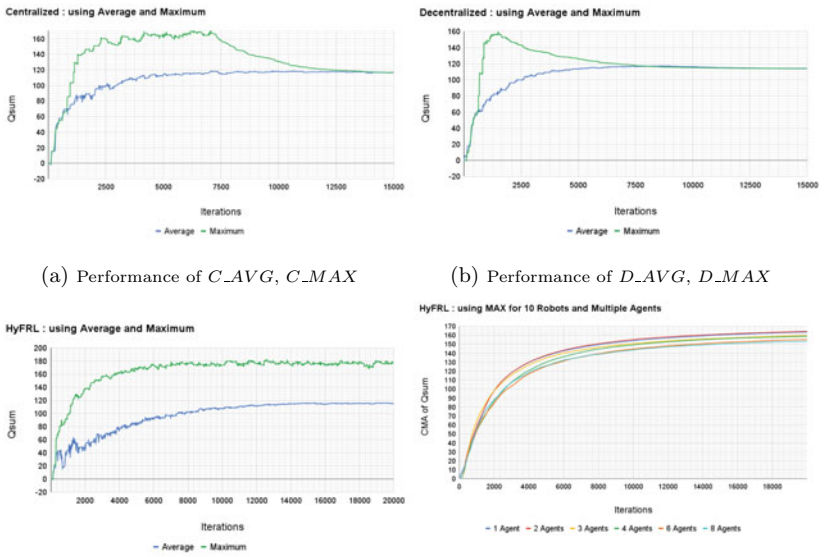
(b) Structure of a Q-table with 10 states (rows) and 4 actions (columns)

Fig. 1 A Sample arena and a Q-table

of RL. The environments of all robots were kept different. In the *HyFRL* implementation, a mobile agent within one of these robots was made to carry the locally generated Q-table (model) as its payload and migrate to the neighboring robot where it aggregated this Q-table with the local one available in this robot and shared the same with it. After visiting τ robots, it was made to send the aggregated Q-table it carries, to the central server hosted on a separate computer. The server aggregated this with the model it already had and sent the aggregated global Q-table to all the robots, thereby constituting a *round*. While the movement and local aggregations facilitated decentralized FRL, the aggregation at the server constituted its centralized counterpart. After several such migrations, the aggregated Q-table formed the optimal policy. The agent was made to continue to do this until the performance saturates.

We have compared the performance of proposed *HyFRL* with those of the fully centralized and fully decentralized versions of FRL, using five connected robots (*Webots*®), each with a single robot. These robots were situated in distinct environments, E_0, E_1, E_2, E_3, and E_4, where the number of obstacles and their respective positions were different. Each robot needed to learn a model suited to its environment. A robot along with its environment is shown in Fig. 1a, and the corresponding Q-table structure is shown in Fig. 1b. Our experiments are conducted by taking 10 states (S_0,...S_9) and 4 actions (A_0, ..., A_4). These four actions are for the robot to move Forward, Backward, Stop, and Left.

We have investigated the performance of two aggregating techniques—*Maximum*, where the maximum of the entries within the Q-tables was taken for aggregation and *Average*, where the mean of the entries formed the aggregated Q-table entries. The 5-robot network was used to compare the performances of the centralized, decentralized, and the proposed *HyFRL* versions. The numbers of agents and of hops were taken to be 1 and 2, respectively. In order to find the effect of using multiple mobile agents to aid in the aggregation process, we conducted experiments on a 10-robot



(a) Performance of C_AVG, C_MAX

(b) Performance of D_AVG, D_MAX

(c) Performance of H_AVG, H_MAX

(d) Performance of $HyFRL\_MAX$ using 10 Robots and Multiple Agents

Fig. 2 Performances of centralized, decentralized, and *HyFRL* approaches

network using the *HyFRL* approach. Each agent was made to start from a different node so as to hasten the process of learning. The performance of an approach was measured based on the *Qsum* values obtained at every iteration. These values were calculated using Eq. (2).

$$Qsum = \frac{\sum_{n=0}^{N-1} \sum_{s=0}^{S-1} \sum_{a=0}^{A-1} Q_{s,a}^{n}}{N} \tag{2}$$

As can be observed, *Qsum* is the average of the sum of all the values, $Q_{s,a}^{n}$, within the Q-tables, of all the robots where N, S, and A stand for the number of robots, states, and actions, respectively. Each experiment was performed 5 times with the robots positioned at different starting locations, and the average of the *Qsum* values obtained from these experiments was used to plot the relevant graphs.

For clarity, the curves in the graphs are labeled as A_M where A could be either C, D, or H signifying Centralized, Decentralized, or Hybrid FRL, respectively. M could be either AVG or MAX indicating the aggregation methods—*Average* or *Maximum*. Figures 2a–c depict the performances of the Centralized, Decentralized, and *HyFRL* in a 5-robot network using both *Average* and *Maximum* for aggregation. Figure 2a shows the performance of the centralized FRL version using the aggregation methods. As can be seen in Fig. 2a, though the C_MAX performs better in between, both C_AVG and C_MAX eventually saturate to the same range of 115–118. The sudden

initial surge in the performance in the case of C_MAX is mainly due to the selection of the maximum values from across the Q-tables. The maximum of one robot may not always be suitable to the others, consequently increasing penalties and lowering the curve to join C_AVG. Another reason for this could be the frequent updates sent to the server by each robot which gives lesser time for the robot(s) to adapt to the updates received. In the decentralized FRL, Fig. 2b, both D_AVG and D_MAX saturate to *Qsum* values in the range of 113–114 over time, indicating a slightly lower performance than their centralized counterparts. This could be due to the delayed updates received by all the robots. In the case of the *HyFRL* version in Fig. 2c, there is a noticeable difference in the performance of both H_AVG and H_MAX. Unlike the centralized and decentralized approaches, both curves do not converge over time. H_AVG saturates to around 115–117, whereas H_MAX saturates to 171–182. This is observed because the frequency of updates is less compared to the centralized version which allows the robots to adapt to the updates received. *HyFRL* using *maximum* as the aggregation method outperforms all the other models. In the subsequent experiment, we, therefore, investigated the effect of increasing the number of mobile agents in the case of *HyFRL* using the *maximum* aggregation method.

Using a network of ten robots, six separate experiments were performed using 1,2,3,4,6, and 8 agents, respectively. Figure 2d shows the comparison of the performances of each experiment wherein the average of all the *Qsum* values within the Q-tables of all the robots per iteration, in an experiment, was taken into consideration. Thus, each curve denotes the cumulative performance of all the robots in an experiment. In each experiment, the average of all the *Qsum* values within the Q-tables of all the robots per iteration was logged. Figure 2d shows the Cumulative Moving Average (CMA) of these values for each of the six experiments. It can be observed that when multiple agents are introduced into the network, the overall performances of the robots vary. For lower ratios of the number of agents μ to the number of robots N, (viz. μ equal to 1,2), the performances seem to increase due to a proportionate increase in updates. However, for larger μ:N ratios, (viz. μ equal to 3,4,6,8), the performances deteriorate because of a drastic increase in the frequency of updates. This gives lesser time for the robots to learn and modify the Q-tables, thereby resulting in redundant models being aggregated. It may thus be inferred that this ratio needs to be determined empirically to increase the performance of the system.

6 Conclusions and Future Work

In this paper, we have proposed a combination of centralized and decentralized FRL, *HyFRL*, to eliminate the problems of single-node dependency and delayed updates. Experiments conducted reveal that the *HyFRL* approach using *maximum* as the aggregation method outperforms centralized and decentralized methods irrespective of the aggregation methods used in the latter two. Learning can be further accelerated by increasing the number of mobile agents in the network. However, this needs to be

done with caution. An appropriate μ:N ratio may need to be found to enhance the performance. In the future, we plan to work on experimenting *HyFRL* using more robots to converge on a prescription for μ:N ratio. In addition, we also intend to investigate the effect of agent hops, τ, on the performance of the system. Heterogeneous learning mechanisms, where robots learn using different learning algorithms (such as State-Action-Reward-State-Action, (SARSA) and Deep Learning in addition to Q-learning), in conjunction with other aggregation methods, in a federated manner, will also aid in hastening convergence and increasing the performance.

References

1. McMahan B, Moore E, Ramage D, Hampson S, Arcas BA (2017) Communication-efficient learning of deep networks from decentralized data. Artific Intell Stat 1273–1282. PMLR
2. AbdulRahman Sawsan, Tout Hanine, Ould-Slimane Hakima, Mourad Azzam, Talhi Chamseddine, Guizani Mohsen (2020) A survey on federated learning: the journey from centralized to distributed on-site learning and beyond. IEEE Internet of Things J 8(7):5476–5497
3. Agrawal A, Kulkarni DD, Nair SB (2020) On Decentralizing Federated Learning. In: 2020 IEEE international conference on systems, man, and cybernetics (SMC)
4. Xianjia Y, Queralta JP, Heikkonen J, Westerlund T (2021) Federated learning in robotic and autonomous systems. Proc Comput Sci 191:135–142
5. Kober J, Bagnell JA, Peters J (2013) Reinforcement learning in robotics: a survey. Int J Robot Res 32(11):1238–1274
6. Liu Boyi, Wang Lujia, Liu Ming (2019) Lifelong federated reinforcement learning: a learning architecture for navigation in cloud robotic systems. IEEE Robot Autom Lett 4(4):4555–4562
7. Nair JS, Kulkarni DD , Joshi A, Suresh S (2022) On decentralizing federated reinforcement learning in multi-robot scenarios. In: 2022 7th South–East Europe design automation, computer engineering, computer networks and social media conference (SEEDA-CECNSM), 1–8. IEEE
8. Zhou W, Li Y, Chen S, Ding B (2018) Real-time data processing architecture for multi-robots based on differential federated learning. In: 2018 IEEE smartWorld, ubiquitous intelligence and computing, advanced and trusted computing, scalable computing and communications, cloud and big data computing, internet of people and smart city innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), 462–471
9. Sutton RS, Barto AG (2018) Reinforcement learning: an introduction. MIT press
10. Semwal T, Bode M, Singh V, Jha SS, Nair SB (2015) Tartarus: a multi-agent platform for integrating cyber-physical systems and robots. In: Proceedings of the 2015 conference on advances in robotics, pp 1–6
11. Michel O (2004) Cyberbotics Ltd. Webots™: professional mobile robot simulation. Int J Adv Robotic Syst 1(1):5