A Platform for Deploying Multi-agent Deep Reinforcement Learning in Microgrid Distributed Control

Tung-Lam Nguyen*, Yu Wang†, Quoc-Bao Duong‡, Quoc-Tuan Tran§, Ha Thi Nguyen¶, Osama A. Mohammed*
*Florida International University, USA
†Imperial College London, UK
‡AIP PRIMECA Dauphine Savoie, Univ. Grenoble Alpes, France
§Univ. Grenoble Alpes, INES, CEA, LITEN, France
¶University of Connecticut, USA

2022 IEEE Power & Energy Society General Meeting (PESGM) | 978-1-6654-0823-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/PESGM48719.2022.9917136

***Abstract*—Distributed control strategies have been attracted significant attention due to numerous advantages over traditional centralized control strategies. The development of deep reinforcement learning method provides a novel approach to control grid without knowing the system's parameters. The training and validating process with grid simulation as environment have been supported by several toolboxes. In this paper, a platform based on redis NoSQL database is proposed to the deploy the multi-agent system of deep reinforcement learning algorithms for control microgrid in a distributed manner. The accuracy of agent implementation under realistic condition with physical communication network can be evaluated with the proposed platform. The distributed control in islanded DC microgrid using Deep Deterministic Policy Gradient is introduced as an use case to show the operation of the platform.**

***Index Terms*—deep reinforcement learning, distributed control, microgrid, redis database, the multi-agent system**

I. Introduction

With the increase of advanced control and computation intelligence as well as the high integration of distributed generation, distributed and autonomous power systems are emerging. Distributed control strategies are potential required feature for the next generation of power systems. Instead of gathering all involved information and processing it in a central way, the data for distributed processes is only local and adjacent for any unit. The distributed control systems can overcome limitations of traditional centralized control approach [1]. The multi-agent system (MAS) is the advanced technology that has been widely applied to realize the distributed control algorithms in AC and DC microgrids (MGs) [2], [3]. The agent is an independent entity which has ability of calculating and transferring data in a peer-to-peer communication network to achieve global objectives in distributed manner. The grid consists of electrical, communication and intelligence infrastructure that form a complex cyber-physical energy system. The distributed control system based on communication and sensor network is becoming a crucial topic to be investigated in modern cyber-physical design of MGs. Distributed strategies and MAS have attracted great attention recently to enhance the operation of MGs. Multiple objectives of accuracy power sharing, voltage/frequency restoration and balanced state of charge for secondary control can be achieved by using distributed finite-time algorithms [4]. [5] deals with the distributed hierarchical control architecture of meshed multi-terminal dc (MTDC) networks. The optimal power flow solved in distributed manner is presented in [6]–[8] by using Alternating Direction of Multipliers Method (ADMM) and Augmented Lagrangian Alternating Direction Inexact Newton (ALADIN).

In the existing distributed methods, the control design depends largely on knowing parameters, e.g. loads information, line impedance, etc. which can increase significantly with the development of grids and the integration of the renewable energy resources. In order to overcome the issues, the reinforcement learning (RL) [9] as well as deep reinforcement learning (DRL) has been applied in MG control to deal with unknown structure and parameter models. The RL agent can give precise decisions only based on a self learning process without a complex mathematical model. The rapid evolution with continuously improved performance makes RL a promising method for robust control operation in MGs [10], [11]. In RL and DRL, the agent observes the system state, take control actions, observe the effects of these actions, and progressively learn an algorithm to maximize a predefined reward. [12], [13] provide DRL based method for distributed current sharing and voltage regulation in DC grid. RL approaches for frequency control of inverter-based AC MGs are presented in [14], [15]. In [16], the authors developed novel adaptive emergency control schemes using DRL for complex power systems.

This work focuses on a platform used to implement the MAS in a realistic condition. Currently, the MAS of RL for distributed control in MGs is facing challenges due to the fact that training and validating processes on MG systems are difficult tasks. The MG environment is simulated on power system simulators, meanwhile, the RL agent is designed and programmed to interact with the environment for the observation and training RL policy. Recently, there are toolboxes providing functions to support the training process that can be used in the electrical field, e.g. Reinforcement Toolbox, Microsoft Project Bonsai Toolbox in Matlab/Simulink. They improve the training performance by integrating the interface between RL agent and simulation models. Nonetheless, the deployment of MAS based distributed RL control is still not mentioned. The system states as inputs of agent are not only the local information but also the data from neighbors. Furthermore, in the validation process using existing toolkits, trained agents operate synchronously (in the same time step in the simulation) with other agents as well as with the simulation model, which does not accurately reflect the actual implementation of the system. Therefore, the accuracy of the agents after training can not be guaranteed under the real communication network. The MAS needs a platform to be validated under working environment.

In this paper, we propose a platform for validating and implementing MAS of DRL to control MG in a distributed manner. In this platform, the agents are operated by considering the asynchronous operation of the agents, the interaction with the physical energy system, and the communication network for data exchange between the agents. The platform will allow the trained agents operate independently and run asynchronously with the simulation environment to online control the grid. The trained agents are therefore more ready for applications on real network grids. The platform is presented for the distributed secondary control in DC MG, but it is built in a system level and can be applied for any other MAS of RL algorithms.

II. Deep Reinforcement Learning for Distributed Secondary Control in DC MGs

In this section, the DRL is firstly introduced in an overview. Then a case of islanded DG MG is presented by using DRL algorithm name Deep Deterministic Policy Gradient (DDPG). These trained agents in this use case will be deployed in the proposed platform described in the following section.

A. *Deep reinforcement learning*

In this section, we only present the general idea of DRL for controlling dynamic systems. The detail of RL concept can be found in [9]. RL is a goal-directed computational approach where a processor learns how to perform a task by interacting with a dynamic environment. This learning approach enables a processor to make decisions to maximize the cumulative reward for the task without being explicitly programmed to achieve predefined objectives. Figure 1 illustrates a general structure of a RL case.

The goal of RL is to train agents to complete a task with an unknown environment. The agent gets observations and a reward from the environment and returns actions to the environment. The reward is used to evaluate action based on predefined criteria is with respect to completing the goal.

The agent consists of two main components: a policy and a learning algorithm.

- The policy is a function that gives decision of actions based on the observations from the environment. The parameters of the policy are adjustable.
- Based on the actions, observations and reward, the learning algorithm is used to iteratively update the policy parameters. The objective of the learning algorithm is to maximize the cumulative reward computed during the task.

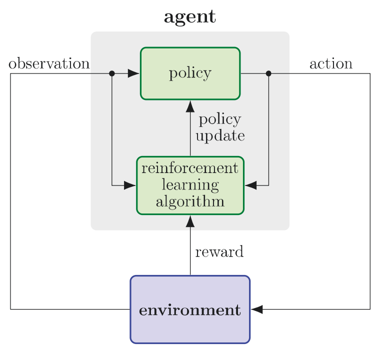

Fig. 1. A general structure of a RL case.

RL can be considered repeated trial-and-error interactions between agent and the environment without human involvement. Depending on the learning algorithm, an agent manages one or more parameterized functions for training the policy. There are two types of functions.

- Critics (policy—based approach): a critic finds the expected value of the long-term future reward from given observation and action.
- Actors (value—based approach): an actor finds the action that maximizes the long-term future reward from a given observation

DRL combines the perception function of deep learning and the decision-making ability of reinforcement learning. For deploying DRL, the policy is a actor represented by a deep neural network (NN) with inputs are state observations and outputs are action returned to the environment.

B. *DDPG Algorithm for Distributed Secondary Control in Islanded DC MGs*

The islanded DC MG consists of three distributed generators (DGs) supplying power for loads in the system. The droop based primary controllers are located locally at each DG to maintain the stability of the MG. The local primary control will react immediately in a decentralized way to balance power between suppliers and consumers when occurring disturbances in the grid. The MAS trained by DDPG algorithm will take the responsibility of the secondary control level in a distributed manner for accurate current sharing and voltage restoration. Figure 2 describes the studied DC MG. The communication topology is illustrated by the communication lines between agents in the figure.

1) DDPG: The deep deterministic policy gradient (DDPG) algorithm is a model-free, online, off-policy reinforcement learning method [17]. A DDPG agent consists of an actor NN and a critic NN that computes an optimal policy to maximizes the long-term reward.

The DRL agent is trained through three main steps: i) the actor and critic parameters are updated at each step of the training process, ii) the experiment results are then stored into

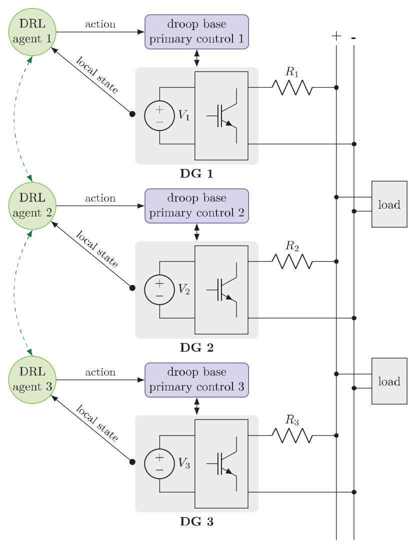

Fig. 2. The studied DC MG with DRL agents.

an experiment buffer, this buffer is extracted randomly by the agent to update the actor and critic NN, and iii) a stochastic noise is added into the policy to perturb the action.

2) Agent design: The secondary control objectives for the DC MG operation are:

$$\begin{aligned} V_1 = V_2 = V_3 = V^* \\ k_1 I_1 = k_2 I_2 = k_3 I_3 \end{aligned} \tag{1}$$

where V_i and I_i are the output voltage and current of ith DG respectively, V^* is the nominal voltage, k_i is the droop coefficient of ith DG.

The inner control loop of each DG receives the reference voltage V_{ref} to determine pulse width modulation (PWM) as the input signal of the DG converter. The DDPG agents are designed to regulate reference voltage to compensate for the deviation of the voltages and output currents simultaneously. The inputs of an agent or the observation signals are local measurements and the data exchange with the neighbors. The reward function of each agent at training step t is defined:

$$r_1^t = \frac{1}{c_1^1|V^* - V_1| + c_1^2|k_1 I_1 - k_2 I_2|} \tag{2}$$

$$r_2^t = \frac{1}{c_2^1|V^* - V_2| + c_2^2|k_2 I_2 - k_1 I_1| + c_2^3|k_2 I_2 - k_3 I_3|} \tag{3}$$

$$r_3^t = \frac{1}{c_3^1|V^* - V_3| + c_3^2|k_3 I_3 - k_2 I_2|} \tag{4}$$

The inputs of the designed DDPG agent i for the training process are $\{V_i,\ I_i,\ \dot{V}_i,\ \dot{I}_i\}$ from local measurement and $\{V_j,\ I_j\}$ from every neighbor agent j. The outputs will be compensation signals which are sent to the primary controllers.

III. The platform for deploying MAS of DRL

Figure 3 shows the proposed platform for deploying validating the agents trained in the previous section. The platform can be extendable for a larger number of agents, different grids and various RL algorithms. The platform consists of three parts: the grid simulation, the multi-agent system and the redis database as an interface. In order to approach to the practical implementation of distributed control systems, the platform fulfils the following requirements:

- The DRL agents run in distinguish processes and operate asynchronously with each other.
- The DRL agents runs asynchronously with the model simulation.
- Each agent only observes local information measured from the outputs of the corresponding DG and the information from neighbor DGs.
- The local controller at each DG only receives the control signal from the corresponding agent.

A. The redis database

A redis database is used as for two purposes: i) the interface between agents and the MG simulation in Matlab/Simulink, and ii) the interface between agents. Redis[1] is an open-source in-memory data structure store, used as a database, cache and message broker. Redis is fast, easy to use, a NoSQL database and being supported in most of the program languages. The redis database can locate on either a local server or a remote server.

There are two sets of variables in the redis database. The output set is for transferring measured data of the grid from the Simulink simulation to the agents. The input set is used to transfer control signals from the outputs of agents to the primary controllers in the simulation. The transferring process between the redis database and the simulation is implemented continuously. It can be seen that the variables in the output set are updated by the data from the simulation, and the variables in the input set are updated by the agent system. The setup of the proposed platform with the database allows the data can flow flexibly between entities in the system.

B. The grid simulation

The grid is modelled to run in Matlab/Simulink. The local controllers located at DGs, including inner control loop and primary control are also integrated into the model for the local and fast response. The grid with only local controllers can operate at stable state but not at nominal state. The DDPG agents, after a large number of training steps, will be used to bring the system to the desired state. We use user Datagram Protocol/Internet Protocol (UDP) to broadcast the measurement signal to outside the Simulink domain as well as collect control signals from other domains. This part of the platform can be considered as the simulation environment presented in Section II. An interface is also built for the exchanging information between Simulink and the redis database.

C. The multi-agent system

The DRL agent system is a cluster of agents, and each agent takes in charge of the controlled device in the grid.

[1]https://redis.io

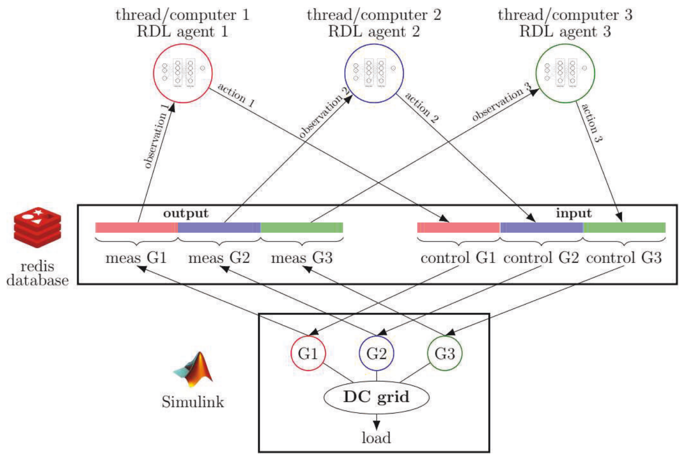

Fig. 3. The platform for deploying MAS of DRL.

The structure of a DRL agent is designed as illustrated in Figure 4. The agent is a C++ program having the ability to collect data, process calculation and send the result back to the system. In order to access data in the redis server, each agent is a redis client and connects to the server when starting. The agents can run in separated threads or in separated machines (computer, embedded system, microprocessor, etc.) within the same communication network connected to the redis server.

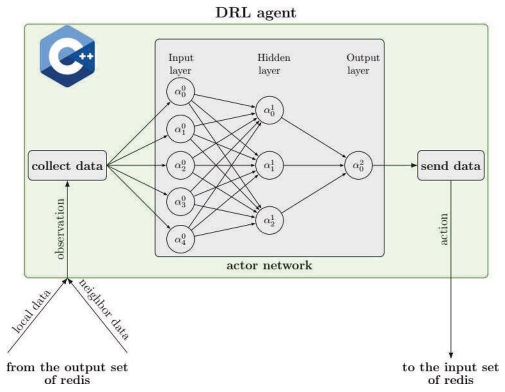

Fig. 4. The RDL agent structure.

In a DRL agent, the learning algorithm and the critic network are only used for the training process, while the policy will be the main part for the deploy process. Once the DRL agents are trained as presented in Section II, the policies of the agents or actor networks are separated and converted from Matlab code to C++ code. Each trained neural network is then integrated into an agent, as shown in Figure 3 and Figure 4 for the deploying process. The core component of the agent is the actor network which handles the system state and gives a proper decision. The agent is set up to collect data only from local measurements and from neighbors which are stored in the output set of redis database. The combination of local data and neighbor data will be the observation of the environment as inputs for the actor network. The output of the network is sent to update the input set of the redis database for adjusting the voltage reference of the corresponding local controller.

IV. Experiment results

The proposed platform is used to verify the operation of the agent system, which is trained for the studied DC MG in Section II. In the test case, three battery systems supply energy to DC MG through DC/DC converter interfaces. The rated voltage of the system is 170VDC. There are three agents corresponding to three DG in the grid, and these three agents are run in three separated threads. The requirements when operating the grid: the grid voltage is maintained at the nominal value $V^* = 170V$, and the output currents of the DGs are shared with a predefined ratio as $I_1 : I_2 : I_3 = 2 : 1 : 1$.

The sequence of starting the whole system in the platform is as follows. Initially, the redis server is launched with values of the input set, and the values of output set are zeros. Then the simulation of the DC MG with local controllers is run in Matlab/Simulink. Finally, the agents are started simultaneously in different threads to control the simulation system.

The communication delays between agents are illustrated in Figure 5. The latencies are not constant but varied in a range with the median is lower than 0.05s. It can be seen that the process of exchanging message among agents reflect working condition of the agent system. The time required to process actor networks when they receive data from local measurements and data exchanges is shown in Figure 6. Although training is time consuming, agents can respond quickly to the system.

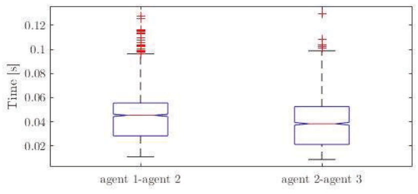

Fig. 5. The communication delay between agents.

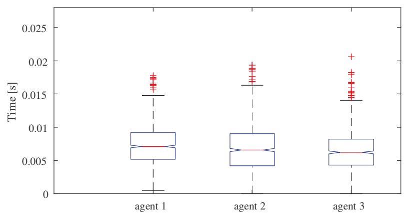

Fig. 6. The processing time for actor networks in agents.

Figure 7 and Figure 8 show the voltage and the output currents of the system. During the initial moment without control signals from the agents, the system appears the voltage deviation and inaccurate current sharing. When the MAS inaugurates, it can be seen that the voltage returns to the nominal value and the output currents are shared at required ration. The MAS of DDPG algorithm is therefore proved convincingly due to it is validated in a more realistic way with the proposed platform. At $2.5s$, the load power is increased by 10%, the actor networks in agents adapt with the variation of the observation inputs. After transient responses, the voltage is maintained at $170V$ and the generated currents of DGs increase proportionally as the objectives of control system.

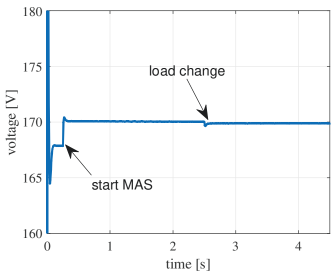

Fig. 7. The voltage of the grid.

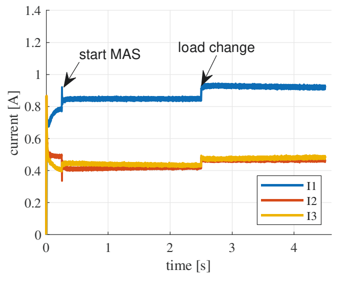

Fig. 8. The output currents.

V. Conclusions

This paper provided a platform for the deployment of the MAS, which implements DRL algorithms in a distributed way. The agents and the grid simulation in the platform operate in separated threads or machines that reflects the practical operation of the system. The data between components in the system is transferred asynchronously through a redis database which fast and flexible. The implementation of the MAS based DDPG algorithm on DC MG showed the operation of the proposed platform.

In the future work, we will develop the platform on a real-time simulator for the grid simulation and a cluster micro-processor for the MAS. The emulation of the communication network will also be integrated into the platform.

References

[1] M. Yazdanian and A. Mehrizi-Sani, "Distributed control techniques in microgrids," *IEEE Transactions on Smart Grid*, 2014.

[2] T. L. Nguyen, E. Guillo-Sansano, M. H. Syed, V. H. Nguyen, S. M. Blair, L. Reguera, Q. T. Tran, R. Caire, G. M. Burt, C. Gavriluta, and N. A. Luu, "Multi-agent system with plug and play feature for distributed secondary control in microgrid—controller and power hardware-in-the-loop implementation," *Energies*, vol. 11, no. 12, pp. 1–21, 2018.

[3] Y. Wang, T. L. Nguyen, M. H. Syed, Y. Xu, E. Guillo-sansano, V.-h. Nguyen, G. Burt, Q.-T. Tran, S. Member, and R. Caire, "A Distributed Control Scheme of Microgrids in Energy Internet and Its Multi-Site Implementation," *IEEE Transactions on Industrial Informatics*, vol. 3203, no. c, pp. 1–10, 2020.

[4] Y. Wang, T. L. Nguyen, Y. Xu, and D. Shi, "Distributed control of heterogeneous energy storage systems in islanded microgrids: Finite-time approach and cyber-physical implementation," *International Journal of Electrical Power and Energy Systems*, vol. 119, no. May 2019, p. 105898, 2020. [Online]. Available: https://doi.org/10.1016/j.ijepes.2020.105898

[5] C. Gavriluta, R. Caire, A. Gomez-Exposito, and N. Hadjsaid, "A Distributed Approach for OPF-Based Secondary Control of MTDC Systems," *IEEE Transactions on Smart Grid*, 2018.

[6] Y. Zhang, M. Hong, E. Dall'Anese, S. V. Dhople, and Z. Xu, "Distributed controllers seeking AC optimal power flow solutions using ADMM," *IEEE Transactions on Smart Grid*, vol. 9, no. 5, pp. 4525–4537, 2018.

[7] T. Faulwasser, A. Engelmann, T. Mühlpfordt, and V. Hagenmeyer, "Optimal power flow: An introduction to predictive, distributed and stochastic control challenges," *At-Automatisierungstechnik*, vol. 66, no. 7, pp. 573–589, 2018.

[8] M. Aragüés-peñalba, T. Lam, R. Caire, A. Sumper, S. Galceran-arellano, Q.-T. Tran, I. Tecnològica, E. Elèctrica, and U. P. D. Catalunya, "Electrical Power and Energy Systems General form of consensus optimization for distributed OPF in HVAC-VSC-HVDC systems," *Electrical Power and Energy Systems*, vol. 121, no. October 2019, p. 106049, 2020. [Online]. Available: https://doi.org/10.1016/j.ijepes.2020.106049

[9] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2017.

[10] M. Glavic, "(Deep) Reinforcement learning for electric power system control and related problems: A short review and perspectives," 2019.

[11] Z. Zhang, D. Zhang, and R. C. Qiu, "Deep reinforcement learning for power system applications: An overview," *CSEE Journal of Power and Energy Systems*, vol. 6, no. 1, pp. 213–225, 2020.

[12] Z. Liu, Y. Luo, R. Zhuo, and X. Jin, "Distributed reinforcement learning to coordinate current sharing and voltage restoration for islanded DC microgrid," *Journal of Modern Power Systems and Clean Energy*, 2018.

[13] X. K. Liu, H. Jiang, Y. W. Wang, and H. He, "A Distributed Iterative Learning Framework for DC Microgrids: Current Sharing and Voltage Regulation," *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2020.

[14] M. Adibi and J. V. D. Woude, "A Reinforcement Learning Approach for Frequency Control of Inverted-Based Microgrids," in *IFAC-PapersOnLine*, 2019.

[15] Z. Yan and Y. Xu, "Data-driven load frequency control for stochastic power systems: A deep reinforcement learning method with continuous action search," *IEEE Transactions on Power Systems*, vol. 34, no. 2, pp. 1653–1656, 2019.

[16] Q. Huang, R. Huang, W. Hao, J. Tan, R. Fan, and Z. Huang, "Adaptive power system emergency control using deep reinforcement learning," *IEEE Transactions on Smart Grid*, vol. 11, no. 2, pp. 1171–1182, 2020.

[17] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, 2016.