Deep Reinforcement Learning for Path Planning by Cooperative Robots: Existing Approaches and Challenges

Walaa Othman
ITMO university
Saint-Petersburg, Russia
walaa_othman@itmo.ru

Nikolay Shilov
SPC RAS
Saint-Petersburg, Russia
nick@iias.spb.su

***Abstract*—Cooperative robots became a very important research topic due to their capability to accomplish tasks fast and more efficiently. In order for cooperative robots to accomplish their tasks, they need to be able to plan their paths in the environment without any collision. Path planning for cooperative robots with deep reinforcement learning is a new research topic in robotics and artificial intelligence. Path planning via deep reinforcement learning is an end to end method. The robot directly receives the data from the sensor (usually high dimensional images) and generates an optimal policy that plans a safe path to the target. In this paper, we first present an overview of reinforcement learning, deep learning, and deep reinforcement learning. Then, we introduce methods used for path planning. Finally, we discuss the challenges of deep reinforcement learning path planning for cooperative robots.**

I. Introduction

Cooperative Robots (CR) refer to a group of homogeneous or heterogeneous software or hardware entities that communicate with each other to solve a specific task. CR became an important research area due to their ability to solve complex tasks more efficiently as compared to a single robot.

To solve the task by CR, the following steps have to be performed [1]:

- Task decomposition: dividing the task into sub-tasks.
- Coalition formation: dividing the robots into different teams .
- Task allocation: assigning a sub-task to each team.
- Control and planning: planning the optimal actions to achieve the task and controlling the robots to guarantee doing these actions.

One of the main challenges in CR is navigation since it is a key and a precondition for advanced behaviors. Navigation can be divided into four main sub-problems [2]:

- perception
- localization
- cognition
- path planning

The problem of perception (modelling the environment) and localization has been ongoing for over two decades. The wide used approach is simultaneous localization and mapping (SLAM) algorithms [3]. Many SLAM algorithms have been proposed for cooperative robots depending on the type of the communication between the robots [4], [5].

The problem of multi robot path planning is well studied using classical ways. Some researchers use graph-search algorithms like A* to find the path for each robot individually and re-plan paths of each robot when a conflict occurs. [6]–[8]. Other research approaches are mainly divided into two categories: (1) centralized methods: state spaces of all robots are combined into one space, and then graph search algorithms are used to find the path [9], [10]; (2) decentralized methods: robots share information about their locations and each robot plans its path to the goal taking into consideration the locations of other robots [11], [12].

Deep Reinforcement Learning (DRL) has been widely discussed in the literature [13], [14]. Many surveys on cooperative robots have been published [15], [16], and some surveys cover a wide range of methods for navigation for multiple robots [17], [18]. Although these surveys usually cover the basic elements of navigation, they are general and not focused on the deep reinforcement learning methods for path planning. In addition, new research approaches have emerged. Therefore, A survey on path planning algorithms based on DRL for cooperative robots would be actual and demanded.

In this survey, we focus on the deep reinforcement learning algorithms used to solve the problem of path planning for cooperative mobile robots. The paper seeks to answer the following research questions:

- RQ1: Which existing approaches exist in the area of path planning in CR, what are their advantages and limitations?
- RQ2: What challenges there exist in CR path planning?

The contribution of the paper is in providing a critical state-of-the-art review in the area of CR path planning and outlining actual for the moment problems that require

attention from the scientific community.

This rest of the paper is organized as follows. In section II we first review the principles of the Reinforcement Learning (RL) with the main algorithms: value-based methods, policy-based methods and actor-critic methods, then we briefly review the Deep Learning (DL) and finally the DRL methods. In section III we answer the first research question and introduce the DRL methods for path planning. In section IV we answer the second research question and present the main challenges existing in DRL. Finally, the conclusion is presented.

II. Background

A. Reinforcement Learning

RL is the technique to learn by trial and error. The learner learn by interacting with the environment over a number of steps. At each step, the learner gets the state of the environment and applies an action based on its current behavior (policy). Then it receives the next state and a reward which evaluates the applied action. The learner's goal is to modify its policy in order to maximize the accumulated rewards. The reinforcement learning can be formulated as a Markov Decision Process (MDP) if the observation about the environment satisfies the Markov property Eq.1 [19]

$$P(s_{t+1}|s_t, s_{t-1}, s_{t-2}....s_0) = P(s_{t+1}|s_t) \tag{1}$$

where s_t is the state at time step t, which mean that the environment response in the future depends only on the current state and there is no need to store the past states once the current state is known.

The MDP can be represented as follows Eq.2

$$(S, A, R, \rho, \gamma) \tag{2}$$

where S is the state space ($s_t \in S$), A is the action space ($a_t \in A$), R is the reward space($r_t \in R$), ρ is the state transition matrix ($\rho_{ss'} = P[s_{t+1} = s'|s_t = s]$), and γ is the discounted factor, which emphasizes on the importance of the intermediate reward over the future rewards. In RL, there are two main important concepts:

- The state value function: measures how good for the learner to be in a state, described in Eq.3

$$V_\pi(s) = \sum_{a\in A} \pi(a|s)[r(s,a) + \gamma \sum_{s'\in S} \rho(s'|s,a)V_\pi(s')] \tag{3}$$

- The action value function: measures how good for the learner to take a specific action, described in Eq.4

$$Q_\pi(s,a) = r(s,a) + \gamma \sum_{s'\in S} \rho(s'|s,a) \sum_{a\in A} \pi(a|s')Q_\pi(s',a) \tag{4}$$

1) Value-based methods: Value-based methods depend on estimating the value of being in a given state. The optimal policy has the maximum value function Eq.5 and vice-versa.

$$V^*(s) = \max_\pi V_\pi(s) \qquad \forall s \in S \tag{5}$$

Two main value-based methods are value-iteration and Q-learning. Value iteration is used when the state transition is known while the Q-learning is used when this information is absence, in other words, the Q-learning used when the robot doesn't know the model of the environment.

The value-iteration tries to maximize the overall value function Eq.6

$$V_{k+1}(s) = \max_{a\in A}[r(s,a) + \gamma \sum_{s'\in S} \rho(s'|s,a)V_k(s')] \tag{6}$$

The Q-learning algorithm updates the action value through Bellman equation Eq.7:

$$Q_{k+1}(s,a) = Q_k(s,a) + \alpha[r_k + \gamma \max_a Q_k(s',a) - Q_k(s,a)] \tag{7}$$

where α is the learning rate.

2) Policy-based methods: The policy-based method doesn't evaluate the value-function, but instead it directly searches for the optimal policy. The most used approach of policy-based methods is RENFORCE (Monte Carlo policy gradient). The policy is modelled with a function respect to θ ($\pi_\theta(a|s)$). The idea can be summarized as optimizing θ which maximizes the total return. The algorithm first initializes the parameter θ arbitrary, then generates a trajectory following the policy π_{theta} ($S_1, A_1, R_1, ...S_T$), then it estimates the return G_t and updates the policy parameter using gradient ascent (Eq.8) for $t = 1, 2, .., T$.

$$\theta = \theta \; + \; \nabla_\theta \alpha \, \gamma^t \, G_t \, ln(\pi_\theta(A_t|s_t)) \tag{8}$$

3) Actor-critic methods: Actor-critic (AC) are hybrid methods, they learn the policy and the value function. The algorithm is composed of two models, that may share parameters:

- Critic: updates the value function parameters (it could be state-value function or action-value function)
- Actor: updates the policy parameters

The method depends on Temporal Difference (TD): it calculates the TD error of the action-value and uses it to update the action-value parameters. Fig. 1 shows the AC.

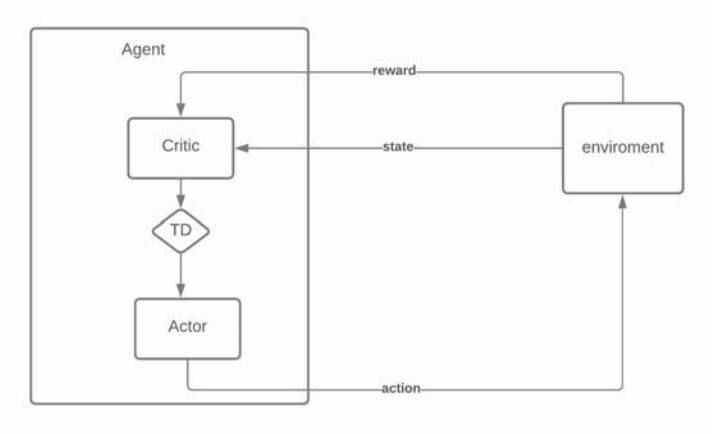

Fig. 1: Actor-Criric reinforcement learning agent

Instead of discounted reward, the AC methods use the advantage Eq. 9.

$$A = Q(s,a) - V(s) \tag{9}$$

The idea of using advantage estimates rather than just discounted returns is to allow the agent to determine not just how good its actions were, but how much better they turned out to be than expected. This allows the algorithm to focus on where the network's predictions were lacking.

B. Deep learning

Deep learning is a field of machine learning that uses neural networks with several hidden layers to extract useful patterns from data. Deep learning showed impressive progress in many tasks like face recognition, image classification, and speech recognition. The most important models in Deep learning are Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), and generative models

- CNNs are known to perform very well in several areas related to Computer Vision and Image Processing like Image Classification and Segmentation, Object Detection, and Video Processing. They use an advantage of applying image related features called "convolutions", which can automatically learn from data. Researches showed that the performance of CNN improved by increasing the networks' width and depth. Since increasing the depth can lead to exploding or vanishing gradients, He et. al. [20] proposed a residual learning framework that solved the degradation problem.
- RNNs are powerful in modelling sequences of data, such as natural language and time series. Their main advantages are:
  - RNNs can process the input of any length,
  - the size of the input doesn't affect the model size.
  - Computations take the historical information into account.
  - The weights are shared across time.

  Although RNN is a very useful tool, it has major drawbacks include:
  - slow computations,
  - difficulty to reach information from the remote past,
  - inability to use future input for the current state evaluation.
- Generative models are aimed to generate new samples that have similar distribution to the training data. Goodfellow et. al. [21] proposed Generative Adversarial Network (GAN), which was composed of two main parts: a generator G and a discriminator D. The generator generates a new sample close to the real one while the discriminator estimates the probability that the sample comes from the input data. The training procedure for G aims to maximize the probability of D making an incorrect discrimination.

C. Deep reinforcement learning methods

1) Deep Q-network (DQN): The Deep Q-network is a TD method, which uses neural networks to estimate the action value function (Eq.4). DQN replaces the Q-table in the RL Q-learning method by a deep network Fig. 2.

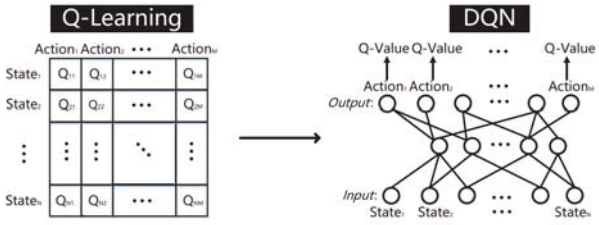

Fig. 2: The difference between Q-learning and DQN [22]

DQN reduces the correlation between the observation by using dataset $D = e1, e2, ...e_T$ contains the robots experience $e_t = (s_t, a_t, r_t, s_{t+1}),\ t \in [1, T]$. At each iteration i , the learning update uses a TD-based loss function (Eq. 10).

$$L_i(\theta_i) = E_{(s,a,r,s')\sim D}[(r+\gamma \max_{a'} Q(s', a', \theta_i^-) - Q(s, a, \theta_i))^2] \tag{10}$$

where θ_i^-,θ_i are the parameters for the target network and the Q-network respectively at iteration i. A major drawback of DQN is that it can only handle low-dimensional discrete action spaces.

2) Double DQN (DDQN): [23] uses two neural networks (DQN and target network) to learn and predict the action the agent needs to take at each time step. The DQN selects the best action with the maximum Q-value of the next state Eq. 11

$$a_{sel} = \max_a Q_q(s_{t+1}, a) \tag{11}$$

The target network calculates the estimated Q-value of the action selected by the DQN Eq.12.

$$q_{est} = Q_t(s_{t+1}, a_{sel}) \tag{12}$$

where Q_t and Q_q are the Q-value for the target network and the DQN respectively. The Q-value of the DQN is updated based on the estimated Q-value from the target network Eq.13.

$$Q_q(s_t, a) = R_{t+1} + Q_t(s_{t+1}, a_{sel}) \tag{13}$$

The update of the parameters of the target network is based on the parameters of the DQN for several parameters, and the update of the DQN is based on the Adam optimizer.

3) Deep Deterministic Policy Gradien (DDPG): DDPG combines the DQN and the actor-critic approaches to learn the policy. DDPG contains four neural networks: current actor networks, target actor currents, current critic networks, and target critic network. The goal of the DDPG is to maintain the actor function which maps the state to action, and to learn the critic function that estimates the value of state-action pairs.The actor is updated as follows (Eq.14):

$$\nabla_{\theta_\mu} J \approx E_{(S_t|\rho_\pi)}[\, \nabla_a Q(s_t, \mu(s_t)|\theta_Q)\ \nabla_{\theta_\mu} \mu(s_t|\theta_\mu)\,] \tag{14}$$

where ρ_π are the transitions generated by a stochastic policy π, $\mu(s_t|\theta_\mu)$ is the actor function, and $Q(s, a)$ is the critic function.

4) Trust Region Policy Optimization (TRPO): TRPO is a policy gradient method, which allows control of the expected improvement of policy during the optimization step. The goal of TRPO is to solve the constrained optimization problem (Eq. 15) by optimizing the stochastic policy at each iteration k.

$$\begin{aligned} &\max_\theta E_{s\sim\rho_{\theta_k}, a\sim\pi_{\theta_k}}[\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A_{\theta_k}(s, a)] \\ &S.T \quad E_{s\sim\rho_{\theta_k}}[D_{KL}(\pi_{\theta_k}(.|s)||\pi_\theta(.|s))] \leq \Delta_{KL} \end{aligned} \tag{15}$$

whereρ_{θ_k} is the discounted frequencies of visiting states reduced by the policy π_θ, $A_{\theta_k}(s, a)$ is the difference between the empirical returns and the baseline, D_{KL} is the KL divergence between two policy distributions, and Δ_{KL} a parameter control of the maximum improvement of policy during the optimization step.

5) Asynchronous Advantage Actor-Critic (A3C): A3C works well in parallel training, it uses the actor-critic framework . In A3C, critics learn the value function while multiple actors are trained and get synced with global parameters. The A3C algorithm is illustrated in Algorithm 1.

Algorithm 1 A3C algorithm

0: Define global parameters θ, w and similar thread-specific parameters θ', w'.
0: Initialize the time step t
0: **While** $T \leq T_{Max}$:
  Reset gradients ($d\theta = 0$, $dw = 0$)
  Synchronize global parameters ($\theta' = \theta$, $w' = w$)
  Sample a starting states_t , $T_{start} = t$
  While $s_t \neq Terminal$ **and** $t - t_{start} \leq t_{Max}$:
    perform action $A_t \sim \pi_{\theta'}(A_t|s_t)$, recieve reward r_t
    Update $t = t + 1$, $T = T + 1$
  Initialize the return estimation:
    $R = V_{w'}(s_t)$ **if** $s_t \neq Terminal$
    $R = 0$ **otherwise**
  For $i = t - 1, ..t_{start}$:
    $R = R + \gamma R_i$
    $d\theta = d\theta + \nabla_{\theta'}\, log(\pi_{\theta'}(a_i|s_i))(R - V_{w'}(s_i))$
    $dw = dw + 2(R - V_{w'}(s_i)\nabla_{w'}(R - V_{w'}(s_i))$
  Asynchronously update θ and w

III. DRL for CR Path Planning

In this section we introduce the main learning schemes for RL multi-robots and we answer the first research question (RQ1) by introducing the main DRL approaches used for CR path planning, its advantages, and limitations.

In general, there are three training schemes for RL that are applied for CR.

- Centralized: The centralized approach assumes a specific model for the actions and observations. In other words, it uses a centralized policy to map the observations of all robots to actions. This is equivalent to Multi-agent Partially Observable Markov Decision Process (MPOMDP) policy. The main problem with this approach is that it is centralized in training and execution, and that causes exponential growth in the observations and actions spaces with the number of robots. This problem is partially solved by factorization of the actor space of the centralized CR.
- Decentralized: In this approach, each agent learn its own policy that maps its own observations to actions. There are two major downsides of this approach: (1) it adds additional computational complexity for the RL task, since the robots don't share knowledge and each robot learns its own policy; (2) for experience-based DRL, changing and adjusting the policies can make the stored experience quickly outdated, since it makes the dynamic of the environment non-stationary.
- Shared parameters: If the CR are homogeneous, it is more efficient when the robots share parameters of single policy. This approach allows the policy to be trained using the experience of all the robots simultaneously, but allows different behaviours among the robots. In other words, the planning is decentralized but the learning is not.

Here we address the main paper and milestone in the field of path planning for CR.

Cruz et al. [24] proposed a method that combines the DRL with the kernel smoothing to solve the path planning algorithm in unknown environments. The kernel smoothing uses the discrete action space without any prior knowledge to approximate the state of the multi-agents, and that reduces the number of states in the Q-table. The main drawback of this algorithm is that the efficiency of the method decreases

as the number of agents increases.

Panov et al. [25] suggested a deep neural implementation of Q-network for path planning on square grids. To force the robots to reach the goal with the shortest path, the reward function is designed as follows (Eq. 16):

$$G(s,g,t) = \begin{cases} \alpha_{opt} r_t^{opt} + \alpha_{rat} r_t^{rat} + \alpha_{euq} r_t^{euq}, & p_t \to 1 \\ r^{obs}, & p_t \to 0 \\ r^{tar}, & p_t = g \end{cases} \quad (16)$$

where $\sum \alpha_i = 1$, p_t is the current position , g is the goal position, $r_t^{opt} = l_t - l_{t-1}$ is the change of optimal distance l to the goal in view of obstacles, $r_t^{rat} = e^{-l_t/l_0}$ gives bigger rewards for cells that are close to the goal, $r_t^{euq} = |p_t - g| - |p_{t-1} - g|$ force the robot to go directly to the goal. If the robot current location is an obstacle ($p_t \to 0$), it receives punishment r^{obs}. If it reaches the goal it receives a big reward r^{tar}. The neural network used to estimate the Q-value consists of 3 convolution layers (3 × 3 filter size and 1 stride) with sigmoid and ReLU activation function , dropout ,5 full connected network accepting flatten output from convolution layers. The paper proved that A*-like algorithms are not suitable for path planning in complex scenes when there are more than one target point and it is better to use robust learning like the proposed DQN in such cases.

Lei et al. [26] used Double DQN for robots local path planning in a high-dimensional space. The reward function is designed to be 1 if the robot reaches the goal, -1 if it hits an obstacle and -0.1 otherwise. In this case, the problem of sparse reward is solved. The Q-network architecture consists of 3 convolution layers (the kernel size of the first and second layer is 2x2 with a stride of 2 and for the third layer is 5x5 with a stride of 1) and one fully connected layer.Also, the state space of the sample pool expanded by inputting lattice information and target coordinates in order to solve the path planning algorithm .

Sui et al. [27] used parallel DQN for multi-agent path planning. The double dueling DQN handled the training and the testing for the multi-agents. The algorithm used for leader-follower multi-robots like the case when the robots need to follow the leader to a specific location to accomplish a certain task. The reward function is designed as in Eq.17.

$$r_t = r_{reach} + r_{collision} + r_{formation} \quad (17)$$

where r_t is immediate reward at time t, r_{reach} is the reward to get when reaching the goal(affect only the leader), $r_collision$ negative reward if leader or follower hit an obstacle, an $r_{formation}$ is a reward to keep the follower robot and the leader robot within a distance range to each other. The sub-rewards are given as follows:

$$r_{collision} = (r_{L-avoid}, r_{F-avoid})$$

$$r_{L-avoid} = r_{F-avoid} = \begin{cases} -1 & if\ collision\ happens \\ 0 & no\ collision \end{cases}$$

$$r_{formation} = \begin{cases} (1,1) & if\ formation\ remaining \\ (-1,-1) & if\ formation\ broken \end{cases}$$

Two networks are used to control the leader and the follower. Each on has the structure of three convolution layers. The output of third layer is input to two streams of fully connected layers (one estimates of the value, the other estimate the advantage function)s of the four actions. Finally the two streams are combined to generate the Q-value function.

Zheng et al. [28] proposed a hierarchical path planning for multi-robots. The design consists of two parts, the upper uses Deep Deterministic Policy Gradient (DDPG) algorithm to achieve global path planning. While lower layer uses the reciprocal velocity obstacles algorithm to achieve collision avoidance. The paper proposes a framework of congestion detection based RL. In this framework, a multi-agent system model is considered for the leader grouping. Each leader find the path for its followers.

Bae et al. [29] proposed multi-robot path planning algorithm using Deep Q-learning combined with Convolution Neural Network (CNN) algorithm. The proposed algorithms uses empirical representation technique. Each agent takes an image of the environment, the transmission layer transfers the image information to the CNN (16 convolution and 3 fully connected layers) taking into account the image area and maintaining the relationship between objects on the screen. The CNN uses the image information to extract the features on the image-level (without the need to treat each pixel independently). The Q value is learned for each robot, while the CNN has the same input with a different expected value. The algorithm used A* time to find the path as parameter to measure the learning success (the algorithm is considered to be successful if it reaches the goal in time shorter than A*). An advantage of this algorithm is that it can be applied on static and dynamic environments, but the training is slow and it gives errors at the beginning of the learning.

Xue et al. [30] proposed an algorithm for avoiding collision using a deep reinforcement learning method based on Double DQN. The input of the algorithm is information including the robots' positions, the target position, and the obstacle size.The output is the robots' directions of movement. The reward function is designed as follows (Eq. 18):

$$R(s^c, a) = k(d_{g_{t-1}} - dg_t) + R_{obs} - c\ timer + R_{goal} \quad (18)$$

where d_g is the distance between the robot and the goal at time t, R_{obs} the collision penalty, timer: the time since the robot starts (this to shorten the time needed to reach the target), and R_{goal} is the reward to reach the goal. The collision penalty and the goal reward are as follows:

$$R_obs = \begin{cases} -500 & d < d_min \\ 0 & else \end{cases}$$

$$R_{goal} = \begin{cases} 5 & if reach the goal \\ 0 & else \end{cases}$$

where d and d_{min} are the current distance and the minimum safe distance between the robot and the obstacle respectively. Experiments show that the proposed method can navigate the mobile robots to the desired target position without colliding with any obstacle and other moving robots. The method was successfully implemented on a physical robot platform.

Lin et al. [31] proposed a DRL method to derive decentralized policy for a team of robots. The learned policy directly maps the raw laser data into a velocity command, which allow the robots to cooperatively plan their motion to accomplish the navigation task (reaching target positions and avoiding obstacles) based on each robot's observations. The reward function is designed as follows (Eq.19):

$$\begin{aligned} R(s_t, a_t) = r_c + \omega_g R_g(s_t, a_t) + \omega_c R_c(s_t, a_t) \\ + \omega_f R_f(s_t, a_t) + \omega_p R_p(s_t, a_t) \end{aligned} \quad (19)$$

Where r_c is a negative constant to motivate robots to reach the goal, and $R_g(s_t, a_t), R_c(s_t, a_t), R_f(s_t, a_t), R_p(s_t, a_t)$ are the rewards for reaching the goal position, avoiding collisions, maintaining connectivity, and achieving smooth motion of the robots, respectively. The proposed algorithm used a mechanism of centralized learning and decentralized execution. Through the learning, the robot interacts with the environment and evolve team policy in trial and error manner. In the execution stage, each robot takes its action depends on its own observation.

Semnani et al. [32] proposed a hybrid algorithm of DRL (GPU based A3C) and Force-based motion planning to solve distributed motion planning problem in dense and dynamic environments. The proposed approach improved the performance of DRL by introducing a new reward function that doesn't only eliminate the requirements of pre-supervised learning but also decreases the probability of collisions in crowded environments.

Wang et al. [33] proposed a DRL-based multi-robot cooperative algorithm to solve multi-robot coordination problem. The algorithm solved the problem of source competitions and obstacle avoidance. The input of the algorithm is the image generated by each robot's perspective, and each robot's reward. The algorithm used a neural network structure modified from the Duel neural network structure [34]. The Duel network structure uses two streams that represent the state value function and the state-dependent action advantage function, then merge the results of the two streams. The proposed method can solve the resource competition problem on the one hand and can solve the static and dynamic obstacle avoidance problems between multi-robot in real time on the other hand. The proposed algorithm shows higher accuracy and robustness as compared with DQN and DDQN.

Damani et al. [35] introduced a distributed reinforcement learning framework for long-life multi-agent path planning. In this framework, agents learn fully decentralized policies to plan paths online in a partially observable world, based on local information.The reward function is simply -0.3 at each time step except for the goal which is +5 and for collision -2 .This motivate agents to reach their goals quickly. The researchers focused on achieving implicit agent coordination by helping agents learn ideal behaviour through conventions.

Table I summarizes the main DRL approaches used in path planning for CR, with the advantages and limitations of each approach answering RQ1.

IV. Main Challenges in DRL for CR

This section presents the existing challenges in using DRL for CR path planning thus answering RQ2. The challenges mainly came from the complexity of the environment which produce two main challenges: lack of generalization and slow learning.

- Lack of generalization: The DRL methods for cooperative robots use neural networks (mainly CNNs followed by fully connected layer) to prepossess the sensor data, extract the features and output the probability distribution over the action. This methods produce very good results on the trained environment, but they lack the generalization. By generalization we mean the main two types:
  - Generalization from one environment to another.
  - Generalization from a simulation environment to a real environment.

  To solve the generalization problem researchers suggested to save data from previous experiments to enhance the robots' reasoning [36].
- Slow learning (long training time): This problem came from two reasons:
  - The inputs of the path planning algorithms are the sensory data (mainly images). This data increases dramatically when the environment is complex (the more complex the environment, the higher the number of interactions with the environment the robot should achieve).
  - The reward is sparse (the robots only get the reward when they reach the goal)

  To solve this problem, researchers suggested to use hierarchy policy (divide the path-planning problem into sub-problems, find the best policy to solve each sub-problem then combine these policies into general policy) [37]–[39]. Another important solution and a very hot topic in research is the meta-learning principle. For CR path planning [40] meta learning approach uses a small amount of data which does not only enhance the performance of the DRL algorithm by reducing the training time but also makes the algorithm more applicable on different environment.

TABLE I. DRL MAIN APPROACHES FOR CR PATH PLANNING

References	Main features	Advantages	Limitations
[24]	Q-learning with neural network and smooth kernel estimation.	Not strongly affected by sensor fluctuations.	The efficiency of the method decreases as the number of agents increases.
[25]	DQN (the neural network for estimation of the Q-value consisting of 3 convolutional layers and 5 fully connected layers).	Efficient when the representation of the environment map is a grid and the map is small.	1. Overestimation of the action value (robots need to estimate Q-value first then update it in unknown environments; the convergence to optimal policy isn't guaranteed for noisy input data). 2. Grid-based representation of the environment is required
[26] [30]	Double DQN. The architecture of the Q-network and the target network is CNN (3 convolutional layers and 1 fully connected layer).	1. Reduces the overestimations. 2. Capable to perform data acquisition and network training in parallel, which improves the training efficiency of the network.	1. Lacks generalization. 2. Discrete number of actions.
[27]	Double dueling DQN. Two separate networks for the leader and the follower with the same structure (3 convolutional layer and 2 steams of fully connected layers to estimate the value and advantage function, that combined together to estimate the Q-value function).	1. Reduces the overestimations more efficiently than Double DQN. 2. Good generalization for new and rapidly changing environments.	The reward function is designed specifically for the leader-follower type of robots and for 1 leader 1 follower robots, so the algorithm fails when robots have to take different paths or when there is more than one follower.
[28]	Hierarchical path planning method. Deep Deterministic Policy Gradient (DDPG) for global path planning, and reciprocal velocity obstacles algorithm for collision avoidance.	1. Efficient in complex environment. 2. Can be used for more than one leader and more than one follower robots. 3. Can be applied to both simulated and real environments.	The approach is limited to the leader-follower type of robots.
[29]	DQN. The neural network consists of 16 convolutional and 3 fully connected layers.	Memory requirements for storing operations and values obtained by using A* algorithm for learning are small.	Ideal movements without taking into consideration the dynamic of the robots are considered, so it is not applicable to real world tasks.
[31]	Centralized learning and decentralized execution	1. Effective and applicable to real world tasks. 2. Does not require building an obstacle map of the environment.	1. Long training time. 2. Validated only on a small number of agents.
[32]	Hybrid algorithm of DRL (GPU based A3C) and Force-based motion planning.	1. High performance of path planning in terms of percentage of successful scenarios and time. 2. Applicable to 2D and 3D simulation environments.	1. The algorithm doe not generalize (works only in simulated environments). 2. Discrete outputs (number of actions) are generated what makes the agent movements inflexible.
[33]	Dueling network-based deep reinforcement learning.	1. Higher accuracy as compared to DQN and DDQN. 2. Effective and robust solution for CR path planning.	1. Only applicable to 2D planes, the training time significantly increases in complex 3D spaces. 2. Lacks generalization.
[35]	Combination of distributed reinforcement learning and imitation (learning while watching other experiments) with improvements in the training code (Ray-based training code).	1. The algorithm is suitable for large number of robots. 2. The planning is done online. 3. The training time is less than of the other approaches.	There is no connection between the robots (each robot plans its path depending on its local observation), so the robots don't use the advantages of multi-robot systems.

V. CONCLUSION

Path planning is a fundamental technology for robots, artificial games and unmanned vehicles. Path planning methods based on deep reinforcement learning has drawn a lot of attention from researchers over the last decade. The main contribution of this paper is providing a critical state-of-the-art review of DRL path planning approaches for cooperative robots. We answered two research questions, namely (i) analyzed existing approaches in the area and identified their advantages and limitations in CR path planning, and (ii) identified and discussed challenges that are still existing in cooperative robots DRL path planning.

We hope that this paper will benefit researchers in the field of navigation and path planning.

ACKNOWLEDGMENT

This research is partially due to the State Research, project number 0073-2019-0005, and Grant 08-08 by the Government of Russian Federation.

REFERENCES

[1] P. Caloud, Wonyun Choi, J. . Latombe, C. Le Pape, and M. Yim, "Indoor automation with many mobile robots," in *EEE International Workshop on Intelligent Robots and Systems, Towards a New Frontier of Applications*, 1990, pp. 67–72 vol.1.

[2] S. Pendleton, H. Andersen, X. Du, X. Shen, M. Meghjani, Y. Eng, D. Rus, and M. Ang, "Perception, planning, control, and coordination for autonomous vehicles," *Machines*, vol. 5, no. 1, p. 6, Feb 2017. [Online]. Available: http://dx.doi.org/10.3390/machines5010006

[3] S. Wang, Z. Wu, and W. Zhang, "An overview of slam," in *Proceedings of 2018 Chinese Intelligent Systems Conference*, Y. Jia, J. Du, and W. Zhang, Eds. Singapore: Springer Singapore, 2019, pp. 673–681.

[4] L. Paull, G. Huang, M. Seto, and J. J. Leonard, "Communication-constrained multi-auv cooperative slam," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 509–516.
[5] K. Y. K. Leung, T. D. Barfoot, and H. H. T. Liu, "Decentralized cooperative simultaneous localization and mapping for dynamic and sparse robot networks," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 3554–3561.
[6] K. Dresner and P. Stone, "A multiagent approach to autonomous intersection management." *J. Artif. Intell. Res. (JAIR)*, vol. 31, pp. 591–656, 01 2008.
[7] M. Jansen and N. Sturtevant, "A new approach to cooperative pathfinding," vol. 3, 05 2008, pp. 1401–1404.
[8] M. Kiadi, J. R. Villar, and Q. Tan, "Synthesized a* multi-robot path planning in an indoor smart lab using distributed cloud computing," in *15th International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO 2020)*, Á. Herrero, C. Cambra, D. Urda, J. Sedano, H. Quintián, and E. Corchado, Eds. Cham: Springer International Publishing, 2021, pp. 580–589.
[9] P. Surynek, "An application of pebble motion on graphs to abstract multi-robot path planning," in *2009 21st IEEE International Conference on Tools with Artificial Intelligence*, 2009, pp. 151–158.
[10] T. Standley and R. Korf, "Complete algorithms for cooperative pathfinding problems." 01 2011, pp. 668–673.
[11] V. Desaraju and J. How, "Decentralized path planning for multi-agent teams with complex constraints," *Autonomous Robots*, vol. 32, 05 2012.
[12] C. Wei, K. V. Hindriks, and C. M. Jonker, "Multi-robot cooperative pathfinding: A decentralized approach," in *Modern Advances in Applied Intelligence*, M. Ali, J.-S. Pan, S.-M. Chen, and M.-F. Horng, Eds. Cham: Springer International Publishing, 2014, pp. 21–31.
[13] S. S. Mousavi, M. Schukat, and E. Howley, "Deep reinforcement learning: An overview," in *Proceedings of SAI Intelligent Systems Conference (IntelliSys) 2016*, Y. Bi, S. Kapoor, and R. Bhatia, Eds. Cham: Springer International Publishing, 2018, pp. 426–440.
[14] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.
[15] Y. Rizk, M. Awad, and E. W. Tunstel, "Cooperative heterogeneous multi-robot systems: A survey," *ACM Comput. Surv.*, vol. 52, no. 2, Apr. 2019. [Online]. Available: https://doi.org/10.1145/3303848
[16] A. Torreño, E. Onaindia, A. Komenda, and M. Štolba, "Cooperative multi-agent planning: A survey," *ACM Comput. Surv.*, vol. 50, no. 6, Nov. 2017. [Online]. Available: https://doi.org/10.1145/3128584
[17] F. Zeng, C. Wang, and S. S. Ge, "A survey on visual navigation for artificial agents with deep reinforcement learning," *IEEE Access*, vol. 8, pp. 135 426–135 442, 2020.
[18] X. Ye and Y. Yang, "From seeing to moving: A survey on learning for visual indoor navigation (vin)," 2020.
[19] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018.
[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.
[21] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014.
[22] J. Zhang, L. Liu, M. Ran, Z. Li, Y. Liu, K. Zhou, G. Li, Z. Xiao, B. Cheng, J. Xing, W. Yangtao, and T. Cheng, "An end-to-end automatic
[26] X. Lei, Z. Zhang, and P. Dong, "Dynamic path planning of unknown environment based on deep reinforcement learning," *Journal of Robotics*, vol. 2018, pp. 1–10, 09 2018.
cloud database tuning system using deep reinforcement learning," 06 2019, pp. 415–432.
[23] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," *CoRR*, vol. abs/1509.06461, 2015. [Online]. Available: http://arxiv.org/abs/1509.06461
[24] D. L. Cruz and W. Yu, "Path planning of multi-agent systems in unknown environment with neural kernel smoothing and reinforcement learning," *Neurocomputing*, vol. 233, pp. 34 – 42, 2017, sI: CCE 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231216313856
[25] A. Panov, K. Yakovlev, and R. Suvorov, "Grid path planning with deep reinforcement learning: Preliminary results," *Procedia Computer Science*, vol. 123, pp. 347–353, 01 2018.
[27] Z. Sui, Z. Pu, J. Yi, and X. Tan, "Path planning of multiagent constrained formation through deep reinforcement learning," in *2018 International Joint Conference on Neural Networks (IJCNN)*, 2018, pp. 1–8.
[28] S. Zheng and H. Liu, "Improved multi-agent deep deterministic policy gradient for path planning-based crowd simulation," *IEEE Access*, vol. 7, pp. 147 755–147 770, 2019.
[29] H. Bae, K. Gidong, J. Kim, D. Qian, and S. Lee, "Multi-robot path planning method using reinforcement learning," *Applied Sciences*, vol. 9, p. 3057, 07 2019.
[30] X. Xue, Z. Li, D. Zhang, and Y. Yan, "A deep reinforcement learning method for mobile robot collision avoidance based on double dqn," in *2019 IEEE 28th International Symposium on Industrial Electronics (ISIE)*, 2019, pp. 2131–2136.
[31] J. Lin, X. Yang, P. Zheng, and H. Cheng, "End-to-end decentralized multi-robot navigation in unknown complex environments via deep reinforcement learning," in *2019 IEEE International Conference on Mechatronics and Automation (ICMA)*, 2019, pp. 2493–2500.
[32] S. H. Semnani, H. Liu, M. Everett, A. de Ruiter, and J. P. How, "Multi-agent motion planning for dense and dynamic environments via deep reinforcement learning," 2020.
[33] D. Wang, H. Deng, and Z. Pan, "Mrcdrl: Multi-robot coordination with deep reinforcement learning," *Neurocomputing*, vol. 406, 04 2020.
[34] Z. Wang, N. de Freitas, and M. Lanctot, "Dueling network architectures for deep reinforcement learning," *CoRR*, vol. abs/1511.06581, 2015. [Online]. Available: http://arxiv.org/abs/1511.06581
[35] M. Damani, Z. Luo, E. Wenzel, and G. Sartoretti, "Primal2: Pathfinding via reinforcement and imitation multi-agent learning – lifelong," 2020.
[36] A. Ramezani and D. Lee, "Memory-based reinforcement learning algorithm for autonomous exploration in unknown environment," *International Journal of Advanced Robotic Systems*, vol. 15, p. 172988141877584, 05 2018.
[37] L. Zuo, Q. Guo, X. Xu, and H. Fu, "A hierarchical path planning approach based on a$*$ and least-squares policy iteration for mobile robots," *Neurocomputing*, vol. 170, 06 2015.
[38] B. Bakker, Z. Zivkovic, and B. Krose, "Hierarchical dynamic programming for robot path planning," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005, pp. 2756–2761.
[39] S. Zhang, M. Sridharan, and C. Washington, "Active visual planning for mobile robot teams using hierarchical pomdps," *IEEE Transactions on Robotics*, vol. 29, no. 4, pp. 975–985, 2013.
[40] D. Chen, Y. Liu, B. Kim, J. Xie, C. S. Hong, and Z. Han, "Edge computing resources reservation in vehicular networks: A meta-learning approach," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 5, pp. 5634–5646, 2020.