Pheromone Based Independent Reinforcement Learning for Multiagent Navigation

No Author Given

No Institute Given

Abstract. Multiagent systems (MAS) have been widely applied in many applications, including computer networks, robotics, and smart grids due to their flexibility, reliability for complex problem-solving. Communication is an important factor for the multiagent world to stay organized and productive. Previously, most existing studies try to pre-define the communication protocols or adopt additional decision modules for instructing the communication schedule, which induces significant communication cost overhead and cannot generalized to a large collection of agents directly. In this paper, we propose a lightweight communication framework—Pheromone Collaborative Deep Q-Network (PCDQN), which combines deep Q-network with the pheromone-driven stigmergy mechanism. The framework takes advantage of the stigmergy mechanism as an indirect communication bridge among independent reinforcement learning agents under partially observable environments. Experiments based on the minefield navigation task have shown that PCDQN exhibits superiority in achieving higher learning efficiency of multiple agents when compared to Deep Q-network(DQN).

Keywords: Multiagent system · Reinforcement Learning · Communication · Stigmergy

1 Introduction

Multiagent systems (MAS) have received tremendous attention in different disciplines, including computer science, civil engineering, and electrical engineering [1], as a means to construct complex systems involving multiple agents and a mechanism for coordination of independent agents' behaviors. Typically, it is intended to act in complex large, open, dynamic, and unpredictable environments. However, for such environments, it is usually difficult and sometimes even impossible to specify the systems a priori at the time of their design and prior to their use. Moreover, the multiagent control task is often too complex to be solved effectively by agents with preprogrammed behaviors. The feasible way to cope with this difficulty is to integrate machine learning techniques in MAS, commonly known as multiagent learning (MAL), which endows individual agents with intelligence to improve the performance of the overall system and their own approach.

In MAS, in order to optimally share resources or to maximize one own's profit, appropriate activity coordination is much concerned by multiple agents.

Whereas previous research on agent coordination mechanisms focused on the off-line design of agent organizations, behavioral rules, negotiation protocols, etc., it was recognized that agents operating in open, dynamic environments must be able to adapt to changing demands and opportunities [14]. The most appropriate technique for this situation in MAL is reinforcement learning (RL) [9], which learns to achieve the given goal by trial-and-error iterations with its environment. Standalone RL algorithms are guaranteed to converge to the optimal strategy, as long as the environment the agent is experiencing is Markovian and the agent is allowed to try out sufficient actions. But an underlying assumption of RL techniques, the dynamics of the environment is not affected by other agents, is violated in multiagent domains. The desired approach to solve this problem is communication, which allows each agent to dynamically adjusts its strategy based on its local observation along with the information received from the other agents.

Communication is one of the hallmarks of bio-intelligent, for instance, humans communicate through languages, while social insects using chemicals as their medium of communication. These allow organisms to share information efficiently between individuals and coordinate on shared tasks successfully. It is therefore not surprising that computer scientists have taken inspiration from studies of the behavior of social insects, to design mechanisms for the communication of the multiagent systems. A particularly interesting body of work is stigmergy [6], which mediates animal-animal interactions to coordinate individual insects' activities and exhibit strong robustness with a simple form. In insect societies, stigmergy is often realized via pheromone-based interactions. For instance, ants are capable of finding the shortest path from a food source to their nest by exploiting pheromone information. And there exists no central control in an ant colony and solutions to problems faced by a colony are emergent rather than predefined. When facing internal perturbations and external challenges, tasks are completed in a decentralized way even if some ants fail. In a word, an ant colony, as a kind of MAS exploiting the stigmergic communication, suite for the solution of problems that are distributed in nature, dynamically changing, and require built-in fault-tolerance [2].

Inspired by stigmergy indirect communication mechanism, in this paper, we propose a novel multi-agent independent RL method called Pheromone Collaborative Deep Q-Network (PCDQN) to enhance mutual communication in MAL with the assistance of a stigmergy mechanism. Particularly, the proposed method employs digital pheromone (DP) network to simulate the existing form of pheromone in the environment, acting as a medium in stigmergy mechanism to enable indirect communication with distributed agents. The remainder of this paper is mainly organized as follows. In Section 2, we describe the concept of reinforcement learning and then present the stigmergy mechanism. Section 3 discusses the details of the proposed Pheromone Collaborative Deep Q-Network (PCDQN), which is a deep q-network algorithm combined with a stigmergy mechanism for multiagent coordination. Subsequently, the experiment result and

analysis of PCDQN are demonstrated on the multiagent navigation problem in Section 4. Finally, we conclude this paper with a summary.

2 Background

2.1 Multiagent systems (MAS) and Reinforcement learning (RL)

We consider a MAS with N agents working coordinately in a partially observed environment, in which coordination is made more difficult by the unpredictable environment and imperfect agents' information. This problem can be modeled as a Decentralized Partially Observable Markov Decision Processes (Dec-POMDPs)[29], denoted by a tuple $\mathcal{G} =< \mathcal{S}, \mathcal{N}, \mathcal{A}, \mathcal{O}, \mathcal{Z}, \mathcal{P}, \mathcal{R}, \gamma >$, where $s \in \mathcal{S}$ denotes the global environmental state. At every time-step $t \in \mathbb{Z}^+$, each agent $i \in \mathcal{N} = \{1, \cdots, n\}$ selects an action $a_i \in \mathcal{A}$, where a joint action stands for $\boldsymbol{a} \coloneqq (a_i)_{i \in \mathcal{N}} \in \mathcal{A}^{\mathcal{N}}$. Since the environment is partially observed, each agent only has access to its local observation $o \in \mathcal{O}$ that is acquired through an observation function $\mathcal{Z}(s, a) \colon \mathcal{S} \times \mathcal{A} \to \mathcal{O}$. The state transition dynamics are determined by $\mathcal{P}(s'|s, \boldsymbol{a}) \coloneqq \mathcal{S} \times \mathcal{A} \to [0, 1]$. Agents optimize towards one shared goal whose performance is measured by $\mathcal{R}(s, \boldsymbol{a}) : \mathcal{S} \times \mathcal{A}^{\mathcal{N}} \to \mathbb{R}$, and $\gamma \in [0, 1)$ discounts the future rewards.

Q-learning algorithm Reinforcement learning problems involve learning what to do—how to map situations to actions—to maximize a numerical reward signal. Essentially, it is a closed-loop problem because the learning system's action influences its later input. More specifically, the agent interacts with its environment at every discrete-time step in a time-sequence $t = 0, 1, 2, 3 \cdots$. The agent-environment interaction is shown in Fig.1.

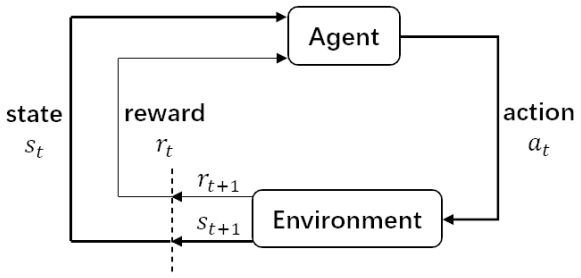

Fig. 1. The agent-environment interaction in reinforcement learning.

In RL, the agent in the state s_t selects action a_t depending on policy π and takes the action and transfers to the next state s_{t+1} with the probability $p[s_{t+1}|s_t, a_t]$, meanwhile receives the reward r_t from the environment. RL

formalizes the interaction of an agent with an environment using a Markov decision process (MDP). An MDP is a tuple $< S, A, r, P, \gamma >$ where S and A are the sets of states and actions, respectively, and $\gamma \in [0, 1]$ is the discount factor. A transition probabilty function $P : S \times A \rightarrow S$ maps states and actions to a probability distribution over next states, and $r : S \times A \rightarrow \mathbb{R}$ denotes the reward. The goal of RL is to learn a policy $\pi : S \rightarrow A$ that solves the MDP by maximizing the expected discount return $R_t = \mathbb{E}[\sum_{k=1}^{\infty} \gamma^k r_{k+t} | \pi]$. The policy induces a value function $V^{\pi}(S) = \mathbb{E}_{\pi}[R_t | s_t = s]$, and an action value function $Q^{\pi}(s, u) = \mathbb{E}_{\pi}[R_t | s_t = s, u_t = u]$.

The optimal quality value is $Q^*(s, u) = \max_{\pi} Q^{\pi}(s, u)$. Given state s, action a, reward r and next state s', it is possible to approximate $Q^*(s, u)$ by iteratively solving the Bellaman equation:

$$Q^*(s, u) = \mathbb{E}_{s'}[r + \gamma \max_{u'} Q^*(s', u') | s, u] \tag{1}$$

A Q-learning agent keeps the estimate of its expected payoff starting in state s_t, taking action u_t as $Q(s_t, u_t)$. Each $Q(s_t, u_t)$ is an estimate of the corresponding optimal Q^* function that maps state-action pairs to the discounted sum of future rewards starting with action u_t at state s_t and following the optimal policy thereafter.

Deep Q-network (DQN) Deep Q-network (DQN) [5] combines reinforcement learning with artificial neural networks known as deep neural networks. It builds on standard Q-learning [12] by approximating the Q-function using a non-linear neural network. And the Q-function parameterized by weights θ called Q-network.

Reinforcement learning is unstable or even diverge when a neural network is used to represent the Q-function. DQN modifies standard online Q-learning in two ways to make it suitable for training large neural networks without diverging. One is experience replay in which at each time step, agent's experiences $e_t = (s_t, u_t, r_t, s_{t+1})$ are stored in a data set $D_t = e_1, \ldots, e_t$ and are sampled uniformly, $(s, u, r, s') \sim U(D)$, as training examples. The other is to use a separate network for generating the Q-learning targets $r + \gamma \max_{u'} \hat{Q}(s', u'; \theta_i^-)$ in the Q-learning update. More precisely, every C updates we clone the network Q to obtain a target network $\hat{Q}$ and use $\hat{Q}$ for generating the Q-learning targets for the following C to Q.

A Q-network can be trained by adjusting the parameters θ_i at iteration i to reduce the mean-squared error in the Bellman equation (Eq.1). The Q-learning update at iteration i uses the following loss function:

$$L_i(\theta_i) = \mathbb{E}_{s,u,r,s' \sim U(D)}[(r + \gamma \max_{u'} \hat{Q}(S', u'; \theta_i^-) - Q(s, u; \theta_i))^2] \tag{2}$$

in which γ is the discount factor determining the agent's horizon, θ_i denotes the parameters of the Q-network at iteration i and θ_i^- is the network parameters used to compute the target at iteration i. The target network parameters θ_i^- are only updated with the Q-network parameters θ_i every C steps and are held fixed between individual updates.

2.2 The Mechanism of Stigmergy

French zoologist Pierre-Paul Grassé firstly introduced the concept of stigmergy for studying the coordination in social insects. Those insects constitute a complex system to enable them to work together effectively. A classic example of stigmergy can be found in the pheromone trails left by ants that come back from a food source. The pheromone stimulates other ants to follow the same path. When food is found, the ants will reinforce the pheromone trail while following the trail back to the nest. This mechanism leads to the emergence of an efficient network of trails connecting the nest via the shortest routes to all the major food sources. The foraging behavior of ants is a feedback loop, where an ant left a pheromone mark which in turn incites an action, which produces another mark, and so on (see Fig.2) [4].

Stigmergy [6], as illustrated in Fig.3, typically encompasses four main components: action, condition, medium, and trace. The most primitive is "action", which is a causal process that produces a change in the state of the environment. Normally, an action is performed by an agent that is an autonomous, goal-directed system. The "condition" specifies the state of the world in which the action occurs, while the action specifies the subsequent transformation of that state. Another core component of stigmergic activity is "medium", as part of the environment it undergoes changes through the actions and whose states are sensed as conditions for further actions. The final component is the "trace", i.e. the perceivable change made in the medium by an action, which is a consequence of the action, carrying information about the action that produced it.

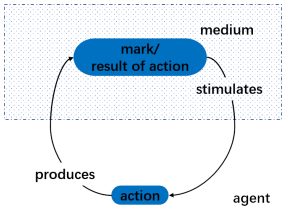

Fig. 2. The stigmergic feedback loop.

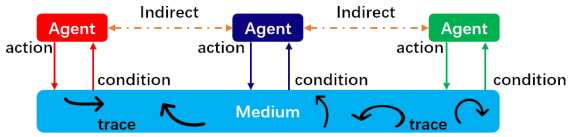

Fig. 3. The stigmergy mechanism.

3 Method

In this section, we propose Pheromone Collaborative Deep Q-Network (PCDQN) (as shown in Fig.4), which is composed of modified DQN and stigmergy respectively. Therein, DQN architecture plays a role as the neural system that directs agents' policy to acclimate the environment and learn from interaction to

achieve their goal. Further, the stigmergy is introduced as an indirect communication bridge among the independent learning agents when coordinating. In what follows, first comes the modified DQN architecture—Dueling Double Deep Q-network with Prioritized Replay that we have used. Then, we show how to fuse this architecture with stigmergy mechanism eventually.

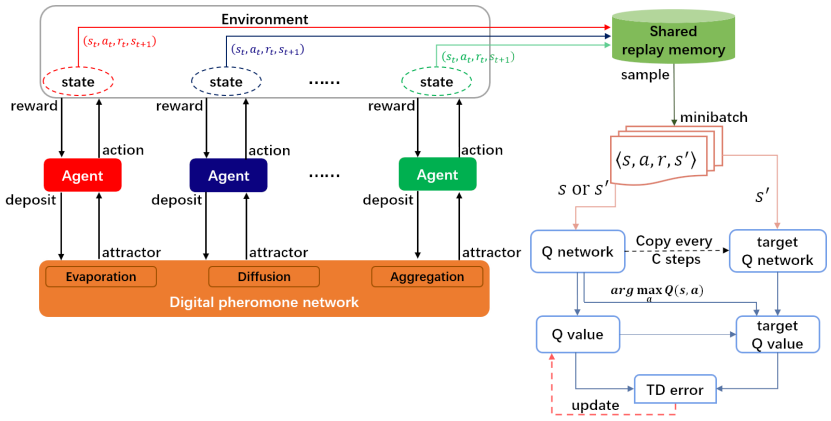

Fig. 4. The framework of Pheromone Collaborative Deep Q-Network (PCDQN).

3.1 Dueling Double Deep Q-network with Prioritized Replay

We propose a parameter sharing multiagent RL structure called Dueling Double Deep Q-network with Prioritized Replay, which extends the standalone DQN algorithm to multiple agents by sharing the parameters of the policy network among our agents. DQN has been an important milestone in DRL, but several limitations of this algorithm are now known in solving completely real-world applications, such as overestimation bias of Q-learning, data inefficiency, and poor approximation of the state-value function. To mitigate the impact of these drawbacks while accelerating the learning performance, we adopt several DQN variants in our implementation for multiagent settings.

DQN variants Deep Q-network (DQN) can learn successful policies directly from high-dimensional sensory inputs using end-to-end reinforcement learning. Although DQN has been well applied for solving completely real-world applications, its has numerous drawbacks, which can be remedied by integrating different schemes from a simple form to complex modifications.

The first and simplest form of DQN's variant is double DQN (DDQN) proposed in [11]. The idea of DDQN is to separate the selection of "greedy" action from action evaluation to reduce the overestimation of Q-values in the training process. In other words, the max operator in Eq.(2) is decoupled into two

different operators, as represented by the following loss function:

$$L_i(\theta_i) = \mathbb{E}_{s,u,r,s\prime\sim U(D)}[(r + \gamma \max_{a'} \hat{Q}(S', \arg\max_{a'} Q(s', a'; \theta_i^-); \theta_i^-) - Q(s, u; \theta_i))^2] \tag{3}$$

DDQN addresses an overestimation bias of Q-learning by decoupling selection and evaluation of the bootstrap action.

Second, the experience replay in DQN plays an important role to break the correlations between samples and remind "rare" samples that the policy network may rapidly forget. However, the fact that selecting random samples from the experience replay does not completely separate the sample data. Specifically, we prefer rare and goal-related samples to appear more frequent than redundancy ones. Therefore, Schaul et al. [8] proposed a prioritized experience replay that gives priority to a sample i based on its absolute value of TD error

$$p_i = |\delta_i| = |r_i + \gamma \max_a Q(s_i, a|\theta_i^-) - Q(s_{i-1}, a_{i-1}|\theta_i)|. \tag{4}$$

Prioritized experience replay improves data efficiency by replaying more often transitions from which there is more to learn.

And third, Wang et al. [13] proposed a novel network architecture called the dueling network. It decouples value and advantage in DQN explicitly by separating the representation of state values and state-dependent action. It consists of two streams that represent the value and advantage functions while sharing a common feature learning module. In this architecture, one network, parameterized by θ, estimates the state-value function $V(s|\theta)$ and the other one, parameterized by θ', estimates the advantage action function $A(s, a|\theta')$. The two networks are then aggregated using the following equation to approximate the Q-value function:

$$Q(s, a|\theta, \theta') = V(s|\theta) + (A(s, s|\theta') - \frac{1}{|\mathcal{A}|} \sum_{a'} A(s, a'|\theta')) \tag{5}$$

The dueling network architecture helps to generalize across actions by separately representing state values and action advantages.

Each of these variants enables substantial performance improvements in isolation. Since they do so by addressing radically different issues and are built on a shared framework, they could plausibly be combined.

Neural Network Architecture In this subsection, we integrate all the aforementioned optimized extensions: Double Q-learning, Prioritized experience reply, and Dueling networks into a single integrated architect, which we call Dueling Double Deep Q-network with Prioritized Replay. As shown in Fig.5, the input of the network is an n-dimensional vector containing sensor signal, digital pheromone information, and the serial number of the agent itself. The first hidden layer is fully-connected and consists of 256 Rectifier Linear Units (ReLu). This is followed by a dueling architecture consisting of two streams to separately estimate state-value and the advantages of each action. Finally, a fully connected linear layer projects its output for each valid action.

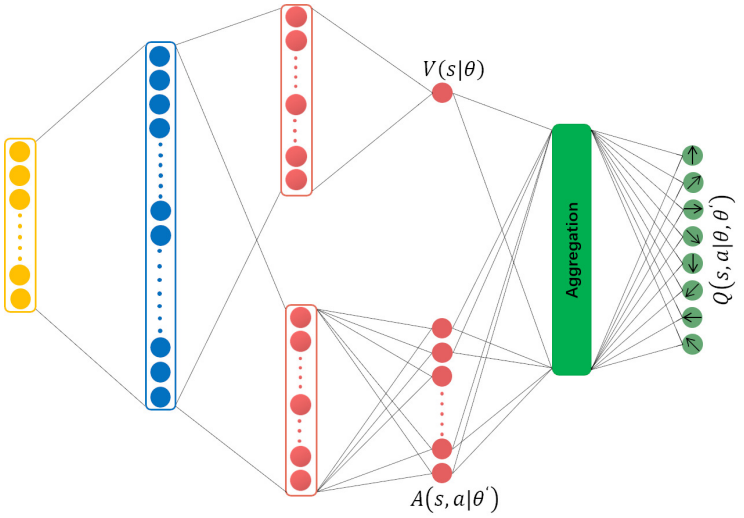

Fig. 5. Schematic illustration of the network architecture.

3.2 Digital Pheromones Coordination Mechanism

Stigmergy is defined as a mechanism for the coordination of actions, in which the trace left by action on some medium stimulates the performance of a subsequent action. In a natural ant colony, ants operate on chemical pheromones—as a form of stigmergy—that support purposive actions. It aggregates deposits from individual agents by fusing information across multiple agents. And it evaporates pheromones over time. Inspired by this mechanism, we develop a kind of digital-analog called digital pheromones that is well suited to the dynamical environment in a distributed way. These data structures live in a network of agents, which located in regions of the environment. Each agent is a neighbor to a limited set of other agents, for which they exchange local pheromone information with one another. And pheromone is quantitative so that it perceived conditions that differ in strength or degree, and where stronger traces typically elicit more forceful actions.

In our setting, we look upon the digital pheromone network as the medium, which allows interaction or communication between different actions and thus, indirectly, between the agents that perform the actions. In a digital pheromone network, each agent can perceive the amount of digital pheromone within a certain range. Here, we regard network nodes filled with digital pheromones as "attractors" in the local environment. It is attractive to the agents nearby and can provide effective observation about the local state space(see Fig.6). In the local state space, each agent needs to select an attractor to conduct its behavior (i.e., close to attractor), independent of several potential agents in its perception range. The probability with which agent k chooses attractor j from its current position node i at time step t is defined as follows:

$$C_{i,j}^{k}(t) = \begin{cases} \frac{D(d_{i,j}^{k}(t))\cdot\varepsilon_j(t)}{\sum_{j\in\mathcal{N}_i^k(t)D(d_{i,j}^{k}(t))\cdot\varepsilon_j(t)}}, & if j \in \mathcal{N}_i^k(t) \\ 0, & others. \end{cases} \tag{6}$$

Each agent k chooses the next attractor j with the highest probability as follows:

$$j = \underset{l\in\mathcal{N}_i^k(t)}{\arg\max}\, D(d_{i,l}^{k}(t)) \cdot \varepsilon_l(t) \tag{7}$$

where $d_{i,j}^{k}(t)$ and $\varepsilon_i(t)$ are, respectively, at time step t, the Euclidean distance between current position node i of agent k and the amount of digital pheromone within attractor j. $\mathcal{N}_i^k(t)$ is the set of alternative attractors within the sensing range of agent k when situating at node i. What's more, the attractor selection rule given by Eq.(6) is called a random-proportional rule [3], which favors transitions towards nodes connected by the short path and with a large amount of pheromone. In addition, we employ a monotonous function $D(\cdot)$ to decrease the effect of digital pheromone as the inter-distance $d_{i,j}^{k}(t)$ increases. As illustrated at the bottom of Fig.6, the amplitude of response for the interactive influence using pheromone is determined by the inter-distance between agents. The function $D(\cdot)$ solves the ping-pong effect [7] in the local environment to enable agents to focus on the attractors nearby. In practical terms, we use the Gaussian function to play the role of $D(\cdot)$, which can be expressed as:

$$D(d_{i,j}^{k}(t)) = a \cdot e^{-\frac{(d_{i,j}^{k}(t)-b)^2}{2c^2}} \tag{8}$$

In this formula, a stands for a peak value of 1 and b is averaging and set to 0. c represents the standard deviation and is normally set to 0.25 preventing the pheromone concentration from varying too much and falling the agent into a local optimum.

When a dynamic change occurs in the environment, some previously generated pheromone trails may lead to a possibly poor solution for the new environment. For this reason, ant colony optimization [3] uses an evaporation-based framework to update pheromone trails and, consequently, they adapt to dynamic change. We are loosely inspired by MAX-MIN Ant System (MMAS), the digital pheromone value of every node j at time step $t+1$ is updated by applying evaporation as follows:

$$\varepsilon_j(t+1) = (\varepsilon_j(t) + \Delta_{diffuse}) \cdot (1-\rho), \forall j \notin D \tag{9}$$

where $\rho \in (0,1]$ is the evaporation rate, $\Delta_{diffuse}$ is the digital pheromone amount diffused by nearby nodes and D is the region consisting of every agent's target node. In particular, following the principle of linear superposition, $\Delta_{diffuse}$ is the summation of digital pheromone diffused from the four adjoining nodes above, below, to the left, and the right of the node. After performing the selected action, the agent k will leave the redundant digital pheromone in the medium to provide new condition information for subsequent attractor selection. The digital pheromone level is updated by applying the depositing rule of Eq.(10).

$$\varepsilon_j(t+1) = \varepsilon_j(t) + \varepsilon_0, \forall j \in D \tag{10}$$

where ε_0 is the constant amount of pheromone to be deposited. The lower and upper limits $\varepsilon_{\min}$ and $\varepsilon_{\max}$ of the digital pheromone values are imposed such that $\forall j : \varepsilon_{\min} < \varepsilon_j < \varepsilon_{\max}$.

4 Experiments

In this section, we describe the Minefield Navigation Environment [10] that using to perform our experiments, as well as demonstrate the effectiveness of PCDQN, its superiority for multiagent navigation.

4.1 Minefield Navigation Environment (MNE)

We adopt MNE that is a two-dimensional world with discrete space and time, consisting of n agents and m obstacles. In the MNE, each agent unit is controlled by an independent agent that must act based on local observations restricted to a limited field of sense centered on that unit (see Fig.7). Agents aim to navigate through a minefield to a selected target in a specified time frame without hitting a mine. An episode terminates when all agents have either died or arrived at their own target, or when a pre-specified time limit is reached.

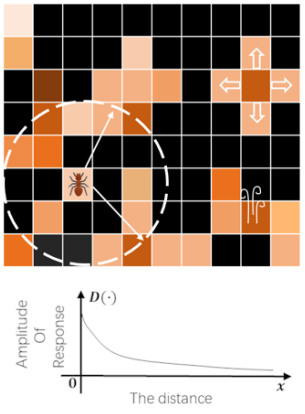

Fig. 6. The digital pheromone network.

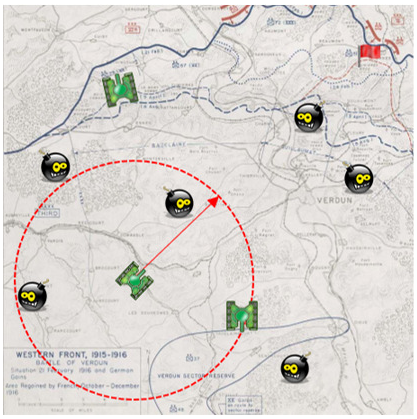

Fig. 7. The sensing range of the agent.

- **State and Observation**: At each step, every agent receives local observation drawn within their field of sense as well as selects a node in the digital pheromone network as its own attractor in the meantime. Among local observation, it encompasses information about obstacles and other agents within

a circular area. The feature vector observed by each agent contains the following attributes: sensor signal, relative position to its destination, attractor location, and its identification number. Every agent has a crude sensory capability with a 360° view based on eight sensors. For each direction $i \in [0, 7]$, the sensory signal is measured by $s_i = \left(\frac{1}{d_i}\right)$, where d_i is the distance to an obstacle or another agent in the direction i.

- **Action Space**: The discrete set of action space that agents are allowed to take consists of eight actions. At each timestep, the agent can choose one possible action within this set, namely, *move north, move northeast, move east, move southeast, move south, move southwest, move west*, and *move northwest*.
- **Rewards**: The sparse reward problem is the core problem of reinforcement learning in solving practical tasks. Solving the sparse reward problem is conducive to improving the sample efficiency and the quality of optimal policy, and promoting the application of deep reinforcement learning to practical tasks. To cope with this, We adopt the shaped reward, which produces a reward based on whether the agent reaches its goal, hits a mine, collides with others, and so on.

4.2 Effectiveness of PCDQN

We start by measuring the effectiveness of PCDQN on a multiagent collaboration task. Our testbed consists of 4 agents, using a 16×16 environment containing 15 mines.

Experimental Setup By using additional reward features, we shape the primary reward to appropriately encourage or punish interactions, filling the gap of the original sparse reward feature. At each timestep t, agent k receives its immediate reward r_t^k defined as follows:

$$r_t^k = \begin{cases} r_{arrive}^k, & \text{if reaching its destination} \\ r_{collision}^k, & \text{if hitting a mine or another} \\ r_{turn}^k + r_{close}^k + r_{stop}^k + r_{range}^k + r_{attractor}^k & \text{otherwise} \end{cases} \tag{11}$$

where r_{arrive}^k and $r_{collision}^k$ are, respectively, the success reward and failure penalty for agent k. When reaching its destination successfully, it receives the reward $r_arrive^k = +1$. While hitting a mine or another agent, it fails and is penalized by getting points $r_collision^k = 1$. Otherwise, r_t^k is composed of r_{turn}^k, r_{close}^k, r_{stop}^k, r_{range}^k, and $r_{attractor}^k$.

To prevent agents from serpentining through the minefield, r_{turn}^k is used to punish agent k for deviating from its previous direction to force it to walk in a straight line. The punishment to agent k who close to obstacles is measured by r_{close}^k as follows:

$$r^k_{close} = -\sigma \cdot \max(s_i), \forall i \in [0, 7] \tag{12}$$

where $\sigma \in [0, 1]$ determines the importance of sensory signal s_i. The r^k_{stop} is set up to prevent the agent k from being stuck on the edge of the minefield for a long time. When the coordinates of agent at current are the same as the previous moment, a certain penalty will be given to it.

Considering the overall goal, agent k cannot simply focus on its own interests such that r^k_{range} is defined by the sum of the distance between every agent and its destination as follows:

$$r^k_{range} = -\lambda \cdot \sum_{j=1}^{n} \min_{1 \geq i \leq n} (dist(i, j)) \tag{13}$$

where $\lambda \in [0, 1]$ determines the significance of the total distance, and $dist(i, j)$ means the distance between agent i and destination j for $\forall i, j \in [1, n]$, and n is the total number of agents existing in the minefield. The $r^k_{attractor}$ as the feedback of its attractor when the agent k closing to or leaving away it is defined as:

$$r^k_{attractor} = \beta \cdot \left(\frac{1}{d_{k,j}} \right) \tag{14}$$

where $\beta \in [0, 1]$ represents the proportion of digital pheromone attribute we adopt in r^k_t, and $d_(k, j)$ is the distance between agent k and its attractor j.

Results Each agent learns from scratch based on the feedback signals received from the environment. We totally run the experiment for 2000 trials and evaluate training at 100-trial intervals. The success rate is determined by the percentage of agents that reach their target positions within the specified time steps. While the failure rate can divide into three parts: hit mine, time out, and collision. They are, respectively, categorized by percentage of agents that hit mines, cannot achieve their goals within the max time steps, or collide with others.

Fig.8 shows the average success and failure rate of PCDQN. The success rate is highly unstable in the beginning but then increases rapidly after 400 trials. By the end of 700 trials, the multiagent system has achieved more than a 90% success rate. It's because the replay buffer has not yet stored enough experiences to update the policy sufficiently at the beginning. In addition, the training still keeps stable despite the non-stationarity of the environment which arises due to the other agents changing their behavior during training. In the meantime, the failure rate decreases as the training progress especially for time out. At the beginning of training, there is a high probability of time out for agents, but after 700 trials, the time out drops to about 0%, indicating that agents have learned to take correct actions when navigating. Besides, we particularly see the benefit of the digital pheromone network. Hit mine and collision also drop significantly after a certain number of rounds of training, indicating that agents have sensed

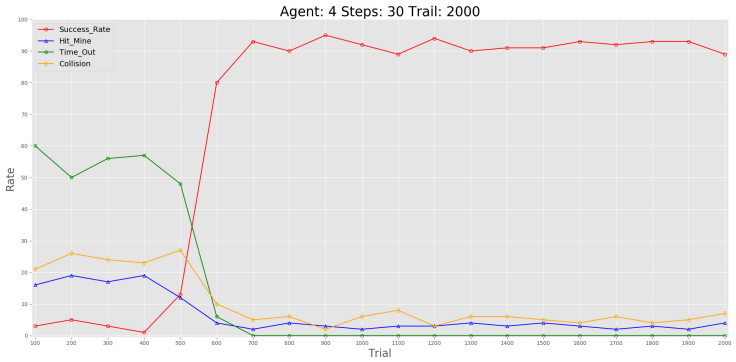

Fig. 8. Success rate and failure (hit mine, time out, and collision) rate of the PCDQN multiagent system averaged at 100-trial intervals on the minefield collaboration task.

traces left in medium and learned to avoid getting too close to mines and other moving agents in the process of moving.

We perform an ablation experiment to investigate the influence of the digital pheromone network in PCDQN. We analyze the significance of digital pheromone on the mixing framework by comparing it against PCDQN without the digital pheromones coordination mechanism. We calculated curves of the average performance for 2000 trials for each approach on this task and we display the accumulated reward over time in Fig.9. With the help of digital pheromones, agents converge to a better policy faster. The multiagent system takes about 700 trials to achieve the best performance without the digital pheromones coordination mechanism. In contrast, the PCDQN multiagent system obtains the best success rate around 600 trials. In terms of stability, PCDQN agents maintain relatively more stability after convergence in the later stage of training. To the lack of pheromone guidance (blue curve in Fig.9), some strong fluctuations will occur due to environmental instability after agents converge to their optimum.

Fig.10 shows the searched navigation paths in the multiagent collaboration task by using the PCDQN framework in MNE. Every agent has successfully not only planed a safe collision-free path when performing its task but also that path is relatively straight not zigzag. We specifically examine the variation of digital pheromone concentration in the whole environment during the process of trial. And we extract a sample in 2000 trials and display digital pheromone concentration at 2-step intervals within 30 steps shown in Fig.11. In the beginning, digital pheromone concentration is 0 everywhere. After several time steps, some of the agents reach their goals and release digital pheromone. In the meantime, these digital pheromone diffuses into their surroundings to guide other agents to achieve their targets faster. Digital pheromone concentration clearly increases as time goes by and achieves its peak at about step 15. At this point, almost all agents reach their destination successfully. After that, digital

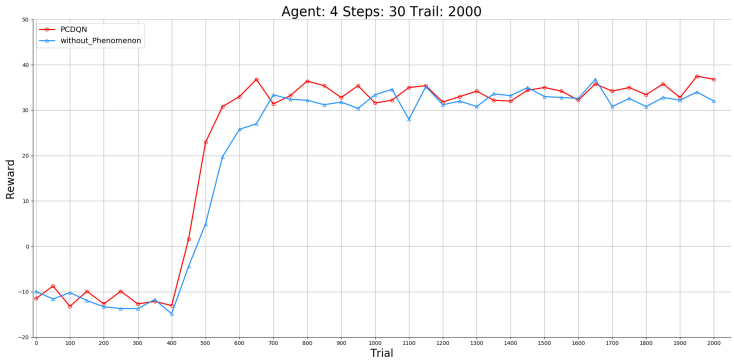

Fig. 9. The accumulated reward curves for PCDQN with (red) and without (blue) the digital pheromones coordination mechanism.

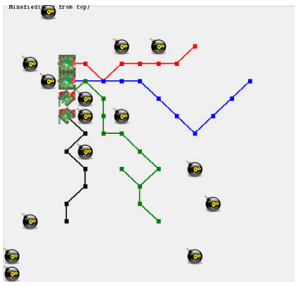

Fig. 10. The searched navigation paths in the MNE.

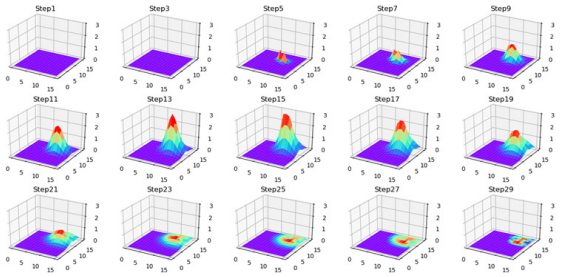

Fig. 11. The change of digital pheromone concentration during training for a single trial.

pheromone gradually evaporates and drops to 0 at the end, which corresponds with the phenomenon observed in the natural ant colony.

5 Conclusion

This paper propose a PCDQN, a deep multiagent RL method that allows end-to-end learning of decentralized policies and makes efficient use of digital pheromones coordination mechanism. Our results in the multiagent navigation task in the MNE show that PCDQN can produce superior performance in terms of stability as well as its learning efficiency, and converge to an optimal policy eventually. In the future, we will investigate the multiagent credit assignment, which produces different rewards for each agent to deduce its own contribution, in turn, make teams achieve their success in more challenging tasks.

References

1. Bressane, A., Silva, P., Fiore, F.A., Carra, T.A., Mota, M.: Fuzzy-based computational intelligence to support screening decision in environmental impact assessment: A complementary tool for a case-by-case project appraisal. Environmental Impact Assessment Review **85** (2020)
2. Dorigo, M., Bonabeau, E., Theraulaz, G.: Ant algorithms and stigmergy. Future Generation Computer Systems **16**(8), 851–871 (2000)
3. Dorigo, M., Gambardella, L.M.: Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Transactions on evolutionary computation **1**(1), 53–66 (1997)
4. Heylighen, F.: Stigmergy as a universal coordination mechanism i: Definition and components. Cognitive Systems Research **38**, 4–13 (2016)
5. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. nature **518**(7540), 529–533 (2015)
6. Musil, J., Musil, A., Biffl, S.: Introduction and Challenges of Environment Architectures for Collective Intelligence Systems. Agent Environments for Multi-Agent Systems IV (2015)
7. Naeem, B., Javed, S., Kasi, M.K., Sani, K.A.: Hybrid fuzzy logic engine for ping-pong effect reduction in cognitive radio network. Wireless Personal Communications **116**(1), 177–205 (2021)
8. Schaul, T., Quan, J., Antonoglou, I., Silver, D.: Prioritized experience replay. arXiv preprint arXiv:1511.05952 (2015)
9. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. The MIT Press, second edn. (2018), http://incompleteideas.net/book/the-book-2nd.html
10. Tan, A.H., Lu, N., Xiao, D.: Integrating temporal difference methods and self-organizing neural networks for reinforcement learning with delayed evaluative feedback. IEEE transactions on neural networks **19**(2), 230–244 (2008)
11. Van Hasselt, H., Guez, A., Silver, D.: Deep reinforcement learning with double q-learning. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 30 (2016)
12. Vinyals, O., Babuschkin, I., Chung, J., Mathieu, M., Jaderberg, M., Czarnecki, W.M., Dudzik, A., Huang, A., Georgiev, P., Powell, R., et al.: Alphastar: Mastering the real-time strategy game starcraft ii. DeepMind blog **2** (2019)
13. Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., Freitas, N.: Dueling network architectures for deep reinforcement learning. In: International conference on machine learning. pp. 1995–2003. PMLR (2016)
14. Weiß, G.: Adaptation and learning in multi-agent systems: Some remarks and a bibliography. In: International Joint Conference on Artificial Intelligence. pp. 1–21. Springer (1995)