Noname manuscript No.
(will be inserted by the editor)

Swarm and Pheromone Based Independent Reinforcement Learning Method for multiagent Path Search Problem

First Author · Second Author

Received: date / Accepted: date

Abstract Multiagent systems (MAS), because of the flexibility, reliability, and intelligence available to solve complex tasks, have been applied widely in different applications, such as computer networks, robotics, and smart grids. Communication is an important factor for the multiagent world to stay organized and productive. Nonetheless, most previous multiagent communication studies try to predefine the communication protocols or adopt additional decision modules for the communication schedule, which cause significant communication 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 a stigmergy mechanism. It takes advantage of the stigmergy mechanism as an indirect communication bridge among independent reinforcement learning agents under partially observable environments. We demonstrate the superiority of the PCDQN framework and also transfer our model to solve the multiagent path search problem. With the PCDQN framework, multiagent formations are able to learn appropriate policies to obtain optimal paths in the Minefield Navigation Environment successfully.

Keywords Multi-agent systems · Reinforcement Learning · Stigmergy · Communication · Ant Colony Optimization

F. Author
first address
Tel.: +123-45-678910
Fax: +123-45-678910
E-mail: fauthor@example.com

S. Author
second address

1 Introduction

Multiagent systems (MAS) have received tremendous attention from scholars in different disciplines, including computer science, civil engineering, and electrical engineering [29], as a means to construct complex systems involving multiple agents and a mechanism for coordination of independent agents' behaviors. MAS consists of autonomous entities known as agents that cooperate or compete to achieve a certain objective. Agents interact with neighboring agents or with the environment to learn new contexts and then use their knowledge to perform an action on the environment to solve their allocated tasks. It is this flexibility that makes MAS has played an important role in modeling complex systems, smart grids, computer networks, and so on. Despite their wide applicability, there contains several technological challenges in MAS such as complex dynamics, high degrees of uncertainty, and enormous computational effort required.

To cope with these challenges, traditional centralized control methods, which specify the decision of the agents according to the global system state, are usually impracticable. Centralized control systems, such as PBS [11], SGE [9], and CLDS [20], lack fault-tolerance ability, constrained scalability, and are restricted from limited computing resources in practice. In contrast, decentralized control, in which an agent in the system deals with a partial observation of the global system state and makes local decisions, offers several potential advantages over centralized ones: robustness to single-point failures and scalability [30]. But MAS also poses certain challenges in decentralized control, many of which do not appear in centralized control. The agents have to coordinate their individual behaviors, such that coherent joint behavior results that are beneficial for the system. Conflicting goals, interagent communication, and incomplete agent views over the process are issues that may also play a role [4]. In order to tackle these problems, it is crucial to building artificially intelligent systems that can productively interact with humans and each other.

MAS typically are intended to act in complex large, open, dynamic, and unpredictable environments. Unfortunately, for such environments, it is extremely difficult and sometimes even impossible to correctly and completely specify these 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. Specifically, which condition will emerge in the future, or which agent will be available at the time of emergence, even how the available agent will have to react and interact in response to these conditions are all unpredictable [28]. In a word, hardwired behavior may become inappropriate in an environment that changes over time. The only feasible way to cope with this difficulty is to integrate machine learning techniques in MAS, commonly known as multiagent learning (MAL), which endow individual agents with intelligence to improve their own and the overall system performance.

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 developing 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 [39]. To effectively utilize opportunities presented and avoid pitfalls, agents need to learn about other agents and adapt local behavior based on group composition and dynamics. The most appropriate technique for this situation in MAL is reinforcement learning (RL) [32]. It is an approach to machine intelligence that learns to achieve the given goal by trial-and-error iterations with its environment. An RL agent learns by interacting with its dynamic environment [28, 32]. At each time step, the agent perceives the state of the environment and takes an action, which causes the environment to transit into a new state. Single-agent RL is usually described within the framework of the Markov decision process (MDP) [32]. Some standalone RL algorithms (for example, Q-learning) 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. In some cases, researchers add an extra communication channel into existing RL frameworks. For example, RIAL and DIAL [8] use discrete communication channels, while CommNet [31] uses a continuous vector channel. Also, agents can share information with communication protocols via shared memory, like MD-MADDPG [25]. Nonetheless, approaches like these introduce additional communication overhead in terms of latency and bandwidth during execution, and their effectiveness is heavily dependent on the usefulness of the received information.

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 the one that focuses on the concept of stigmergy [10], which is a class of mechanisms that mediate animal-animal interactions to coordinate individual insects' activities and exhibit strong robustness with a simple form. It consists of indirect communication that is taking place between individuals of an insect society by local modifications induced by these insects on their environment. Currently, this concept has extended to a variety of aspects, ranging from physics and chemical reactions [12] to diverse application fields such as task allocation in a multi-robot system [3], routing in communication networks [14], exploratory data analysis [26], and even play a central role in brain activity [5]. In insect societies, stigmergy often realized via pheromone-based interactions.

For instance, when foraging, by exploiting pheromone-based interactions, ants are capable of finding the shortest path from a food source to their nest by exploiting pheromone information. 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 individuals fail. In a word, an ant colony, as a kind of MAS exploiting the stigmergic communication, exhibits a number of properties like decentralization, self-organization, flexibility, and robustness, which make them well suited for the solution of problems that are distributed in nature, dynamically changing, and require built-in fault-tolerance [6].

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). It shows that how reinforcement learning with the help of a stigmergy mechanism to enhance mutual communication in MAS. PCDQN takes a Q-learning approach, in which the learned action-value function, directly approximates the optimal action-value function, independent of the policy being followed. Our method is based on three main ideas.

First, we extend standalone deep RL algorithms Deep Q-Network (DQN) [19], which learn successful policies directly from raw sensory inputs using end-to-end reinforcement learning, to the coordinate multiagent context. Multiple agents with limited local observations share a decentralized parameter sharing neural network policy in the partially observable environment. Our empirical evaluations show that every agent performs its manner correctly and the whole MAS converges to the optimal although in the nonstationary environments.

Second, we introduce the stigmergy mechanism into independent RL, which can coordinate different independent learning agents in complex environments. In order to bring properties of pheromone to MAS, we employ the digital pheromone (DP) network to simulate the existence form of pheromone in the environment, acting as a medium in stigmergy mechanism to enable indirect communication with distributed agents. Agents judge the state of the surrounding environment, leave a certain number of digital pheromones at the current location, and generate a DP network by integrating the digital pheromones in the environment. As time goes by, the increasingly refined DP network enables all distributed agents to gather toward the destinations with high digital pheromones volume.

Third, we design a set of unique reward functions for different multiagent tasks. Reinforcement learning can be time-consuming because the learning algorithms have to determine the long term consequences of their actions using delayed feedback or rewards. On this account, we apply the reward shaping method incorporating domain knowledge into reinforcement learning to guide algorithms towards more promising solutions faster.

The remainder of this paper is mainly organized as follows. In Section , we describe the concept of reinforcement learning and then present the stigmergy mechanism. In Section , we discuss 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. In Section , we describe the simulation scenario. And simulation results for the autonomous multiagent navigation problem in a complex environment with obstacles and goal locations also have been shown. Finally, we conclude this paper with a summary.

2 Related Work

In MAS, agents can leverage machine learning algorithms to discover and forecast the changes in the environment and adapt to unforeseen situations and thus form MAL systems. Therein, a significant part of the research on MAL concerns RL [32] techniques, a way of programming the agents using reward and punishment scalar signals without specifying how the task is to be achieved. RL formalizes the interaction of an agent with an environment using a Markov decision process (MDP) [32] defined by the state, action, and the one-step dynamics of the environment. RL can provide a robust and natural means for agents to learn how to coordinate their action choices in multiagent systems. In the literature, the simplest way to apply RL to multiple agents is to have each agent learn independently, with its own policy, from its own action-observation history. Tampuu et al. [33] first proposed playing the Atari Pong game with two independent DQN [19] agents. The result indicates that DQNs can be extended for the decentralized learning of multi-agent systems. And the same method has been used by Leibo et al. [17] to study the emergence of collaboration and defection in sequential social dilemmas. Bansal et al. [2] trained independent learning agents with Proximal Policy Optimization (PPO) in competitive scenarios. They demonstrated that a competitive multiagent environment trained with self-play can produce behaviors that are far more complex than the environment itself. Palmer et al. [23] proposed LDQN algorithm, which introduced the lenient policy into the DQN and adopted the leniency treatment method for the update of negative policies to improve the convergence and stability. The weakness of independent agents is that by treating other agents as part of the environment, it ignores the fact that their policies change over time. Meanwhile, independent learners make the environment to be non-Markovian from an agent's perspective thus violating the property that traditional single-agent learning methods rely upon [16]. Therefore, MAS, defined as a network of individual agents, is required to share knowledge, make decisions in the uncertain and dynamic environment, coordinate to achieve a goal, as well as communicate.

Communication is one of the crucial components in MAS that needs careful consideration. It is widely accepted that communication can further enhance the collective intelligence of learning agents in their attempt to complete coordinated tasks. To this end, a number of papers have previously studied communication protocols to use among multiple agents in RL. Foerster et al. [8] first proposed the mechanism of updating the communication model through the

backpropagation method, namely Differentiable Inter-Agent Learning (DIAL). DIAL takes information from other agents as input, providing a reference for agent decision-making. Gradient flow will update the communication generator layer at the same time. However, the communication of DIAL is rather simple, just selecting predefined messages. Similarly, Sukhbaatar et al. [31] proposed the CommNet allowing dynamic agents to learn continuous communication alongside their policy for fully cooperative tasks. However, it is only a large single network for all agents, so it cannot easily scale and would perform poorly in the environment with a large number of agents. Unlike the approaches mentioned above, BiCNet [24] uses bidirectional RNN as a communication channel and stores local state memory. But it requires all-to-all communication among the agents and assumes each agent to obtain the global state of the environment, which can cause significant communication overhead and latency. Kim et al. [15] inspired by the dropout technique proposed a similar approach in multiagent environments where direct communication through messages is allowed, but only a fraction of the agents can broadcast their messages at each time. Jiang et al. [13] proposed ATOC to use attention to complete communication. ATOC uses an attention unit to build a communication group and a bidirectional LSTM unit as a communication channel. Although ATOC supports an environment of a large number of agents, it introduced the additional attention unit and communication channel to greatly increase the model complexity.

In comparison with the direct communication techniques within multiagent reinforcement learning (MARL) mentioned above, we introduce a kind of indirect and lightweight communication protocol into multiagent teams, which does not need to consider the constraints imposed by limited communication bandwidth and also does not adopt additional decision modules for the communication. Our work is built on the notion of stigmergic communication introduced by Marco Dorigo [6]. The basic idea underlying this form of communication is that pheromones are used as a medium for transmitting messages among artificial ants. Ants communicate by spreading pheromones into their neighborhood during their activities. Such a deposition of pheromones modifies the environment for other ants and stimulates the performance of their subsequent actions. Based on such implicit information pheromones, agents can obtain the local environmental states through indirect communication to coordinate the actions to ensure the stability of the system.

3 Background

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

We consider a MAS with N agents working coordinately to fulfill a given task in a partially observed environment. In this MAS, coordination is made more difficult by the fact that the environment is unpredictable and the information available about the world and other agents is noisy and imperfect. This prob-

lems can be modeled as a Decentralized Partially Observable Markov Decision Processes (Dec-POMDPs)[29] denoted by a tuple $\mathcal{G} =< \mathcal{S}, \mathcal{N}, \mathcal{A}, \mathcal{O}, \mathcal{Z}, \mathcal{P}, \mathcal{R}, \gamma >$. Within $\mathcal{G}$, $s \in \mathcal{S}$ denotes the globle 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.

Solutions to Dec-POMDPs [22] optimize the behavior of the agents while considering the uncertainty related to the environment and other agents. However, solving Dec-POMDPs [22] is often intractable, especially for communication capabilities are limited or absent. In the regular Dec-POMDP, the setting without explicitly modeled communication. Nevertheless, in a Dec-POMDP the agents might very well communicate through their regular actions and regular observations directly or indirectly. Typically, a multiagent decision framework with a separate set of communication actions supports direct communication. If the act of influencing the observations of one agent through the actions of another, frameworks allowing for indirect communication [22].

3.1.1 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 shown in Fig.1.

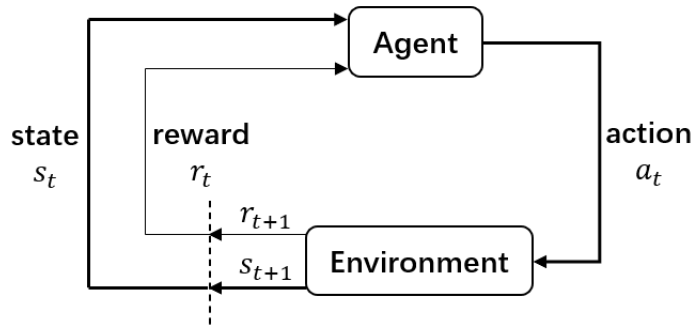


Fig. 1 The agent-environment interaction in reinforcement learning.

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 $P : S \times A \to S$ maps states and actions to a probability distribution over next states, and $r : S \times A \to \mathbb{R}$ denotes the reward. The goal of RL is to learn a policy $\pi : S \to A$ 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:

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

It has been devised for stationary, single-agent, fully observable environments with discrete actions. 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.

3.1.2 Deep Q-network (DQN)

Deep Q-network (DQN) [19] combines reinforcement learning with artificial neural networks known as deep neural networks. It builds on standard Q-learning [37] 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 are the parameters of the Q-network at iteration i and θ_i^- are 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.

3.1.3 Independent DQN

In MARL, learning agents can be distinguished into two fundamental classes of independent learners (ILs) and joint-action learners (JALs). ILs ignore the existence of other agents. JALs, in contrast, are capable of perceiving their actions and/or their rewards. However, in many practical applications, it is not reasonable to assume the observability of the actions of the other agents. On the contrary, ILs, interacting with their surroundings by relying on sensory information and action recognition, work well in practice which has shown in substantial empirical evidence [18].

In independent Q-learning (IQL) [34], the simplest and most popular approach to MARL, each agent learns its own Q-function that conditions only on the state and its own action. IQL avoids the scalability problems of trying to learn a joint Q-function for JALs. And it is also naturally suited to partially observable settings, since, by construction, it learns decentralized policies in which each agent's action conditions only on its own observations. Since our setting is partially observable, IQL can be implemented by having each agent condition on its own action and observation. In deep RL, this can be achieved by having each agent perform DQN using a deep neural network trained on its own observations and actions.

DQN has been extended to cooperative multiagent settings, in which each agent a observes the global s_t, selects an individual action u_t^a, and receives a team reward, r_t, shared among all agents. Tampuu et al. [33] address this setting with the framework that combines DQN with independent Q-learning, in which each agent a independent and simultaneously learns its own Q-function $Q^a(s, u^a; \theta_i^a)$. This work was successfully applied to the two-player pong game in both cooperative and competitive environment.

3.1.4 Reward structure

Homogeneous agents who shared global rewards and co-trained have been widely observed that they usually learn similar behaviors in multiagent coordination settings. Such similar behaviors can easily make the learned policies fall into the local optimum. If agents can respectively take different behaviors, they are more likely to complete the task successfully. Unfortunately, in MARL, a great challenge to coordinate decentralized MAS is generating diversified behaviors for each agent who received only a team reward. The key to solving this problem is to enable agents to develop individuality through interactions with the environment during learning. Reward shaping technique [21], which supplies additional rewards to a learning agent to guide an agent's exploration of its environment, allows agents to develop individuality in a fine-grained manner. Modeling a MAS as a Dec-POMDP allows us to shape an alternative

local reward restricting reward signal to only those agents that are involved in the success or failure at a task. Bagnell and Ng [1] have shown that such local information can help reduce the number of samples required for learning to drastically improve training time.

3.2 The Mechanism of Stigmergy

By using reinforcement learning (RL) in a multiagent system (MAS), we could use autonomous and adaptive agents that can learn to resolve complex problems independently. Unfortunately, there are still many issues in using single-agent algorithms directly in a MAS. First, convergence hypotheses of the single-agent framework frequently become invalid in multiagent environments. Another challenge is the difficulty of defining a good learning goal for the multiple RL agents. The matter of the communication between agents is also an important point: only relevant information in communication can reduce some undesirable effects of locality in MAS. Finally, a fundamental issue faced by agents is how to efficiently coordinate themselves such that a coherent joint behavior results. Those issues we mentioned above have already been solved perfectly in nature insect societies by stigmergy mechanism.

In 1959, French zoologist Pierre-Paul Grassé firstly introduced the concept of stigmergy to understand the coordination in social insects. Those insects or agents constitute a complex system to enable them to work together effectively. In that, insect societies were conceptualized as simple agents that can perform complex swarms tasks using various forms of self-organization and especially stigmergy. 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 they find food, they too 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. With such a stigmergy organization, no conflicts between instructions and reality arise, no needless delays occur and no effort is wasted. The agents in an ant colony are individually goal-directed. 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) [12].

The notion of stigmergy [10], 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. As causal processes, actions have an antecedent and a consequence. In agent-based models used in artificial intelligence, the antecedent is usually called condition and the consequent simply action. 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”. The ”medium”

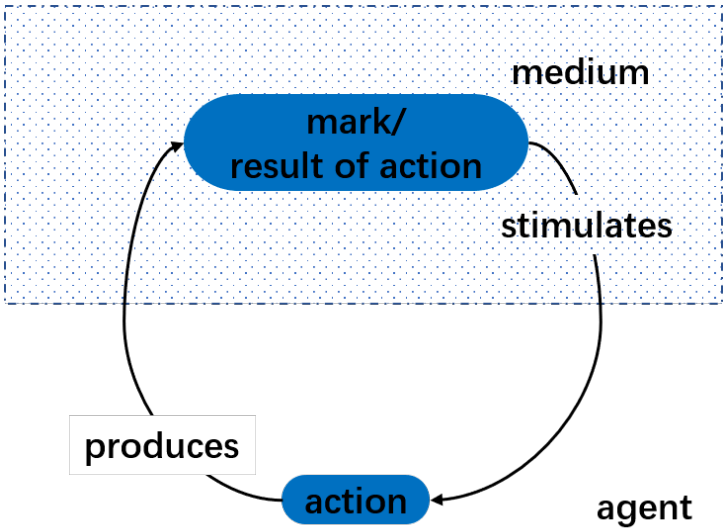


Fig. 2 The stigmergic feedback loop.

is that part of the environment that undergoes changes through the actions and whose states are sensed as conditions for further actions. The medium is a non-trivial entity because the role of the medium is to allow interaction or communication between different actions and thus, indirectly, between the agents that perform the actions. It is this mediating function that underlies the true power of stigmergy. A final component of a stigmergic system is the "trace", i.e. the perceivable change made in the medium by an action, which may trigger a subsequent action. The trace is a consequence of the action and as such, it carries information about the action that produced it.

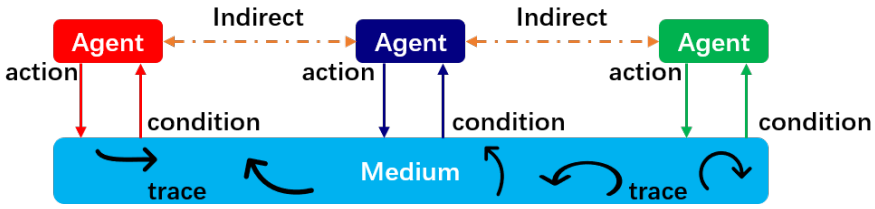


Fig. 3 The stigmergy mechanism.

4 Method

Coordination is a central issue in the distributed MAS. Agents are seldom stand-alone systems and usually involve more than one agent working in parallel to achieve a common goal. When multiple agents are employed to achieve a goal, there is a necessity to coordinate their actions to ensure the stability of the system. A classic coordination technique among agents in a distributed architecture is through a communication mechanism. Rather than transmitting information directly, we instill indirect communication, which based on the observed behavior, not communication, of other agents, and its effects on the environment into general RL architecture. This type of communication is referred to as stigmergic in biology, where it refers to communication base on modifications of the environment not direct message passing. In this section, we describe our approach Pheromone Collaborative Deep Q-Network (PCDQN) that extending DQN [19] to multiagent settings. The proposed Pheromone Collaborative Deep Q-Network (PCDQN) framework (as shown in Fig.4) is composed of modified DQN architecture and stigmergy mechanism respectively. These two components are not independent but reciprocity and mutual benefit. Therein, DQN architecture plays a role as the nervous system that directs agents' policy to acclimate the environment and learn from interaction to achieve their goal. While the stigmergy as an indirect communication bridge among the independent learning agents to reduce the adverse effects of the dynamic environment 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 so as to form Pheromone Collaborative Deep Q-Network (PCDQN) framework eventually.

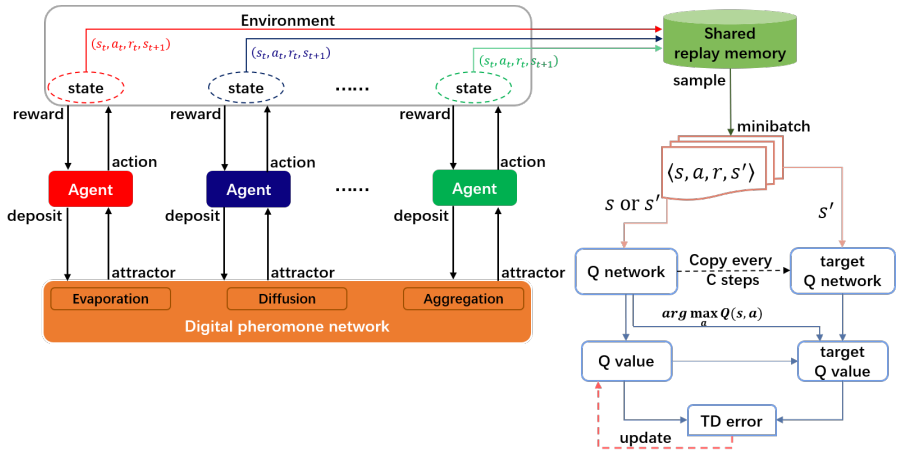


Fig. 4 The framework of Pheromone Collaborative Deep Q-Network (PCDQN).

4.1 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 avoid these drawbacks as to accelerate the learning process and enhance stability, we adopt several DQN variants in our implementation for multiagent settings. In the following subsections, we detail the DQN variants adopted and the network architecture specially designed for our environment.

4.1.1 DQN variants

Deep Q-network (DQN), using deep neural networks to create a kind of novel artificial agent, can learn successful policies directly from high-dimensional sensory inputs using end-to-end reinforcement learning. Although DQN basically solved a challenging problem in RL, the curse of dimensionality, this is just a rudimental step in solving completely real-world applications, DQN has numerous drawbacks, which can be remedied by different schemes, from a simple form to complex modifications.

The first and simplest form of DQN's variant is double DQN (DDQN) proposed in [36]. The idea of DDQN is to separate the selection of "greedy" action from action evaluation. In this way, DDQN expects 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:

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

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 at the same time, remind "rare" samples that the policy network may rapidly forget. However, the fact that selecting randomly samples from the experience replay does not completely separate the sample data. Specifically, we prefer rare and goalrelated samples to appear more frequent than redundancy ones. Therefore, Schaul et al. [27] 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. [38] 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. Concretely, the dueling architecture consists of two streams that represent the value and advantage functions while sharing a common feature learning module. In this architecture, two collateral networks coexist: 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 since they build on a shared framework, they could plausibly be combined.

4.1.2 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.

The traditional DQN combines the Q-learning algorithm with a deep convolutional network. It preprocesses the 4 most recent frames and stacks into 84×84×4 image as input to the convolutional network. This standard DQN architecture, in which there is a separate output unit for each possible action, and only the state representation is an input, consists of three convolutional layers and two fully-connected layers. Since observations are from a local perspective, we do not benefit from convolutional networks, but use a fully connected linear layer to process the observations. Actually, we instead use low-dimensional sensory as state representation, which is dramatically lower than 84×84×4 dimension. For this reason, we modify the original DQN architecture greatly to adapt to our environment.

The exact architecture, shown schematically in Fig.5, is as follows. The input to 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.

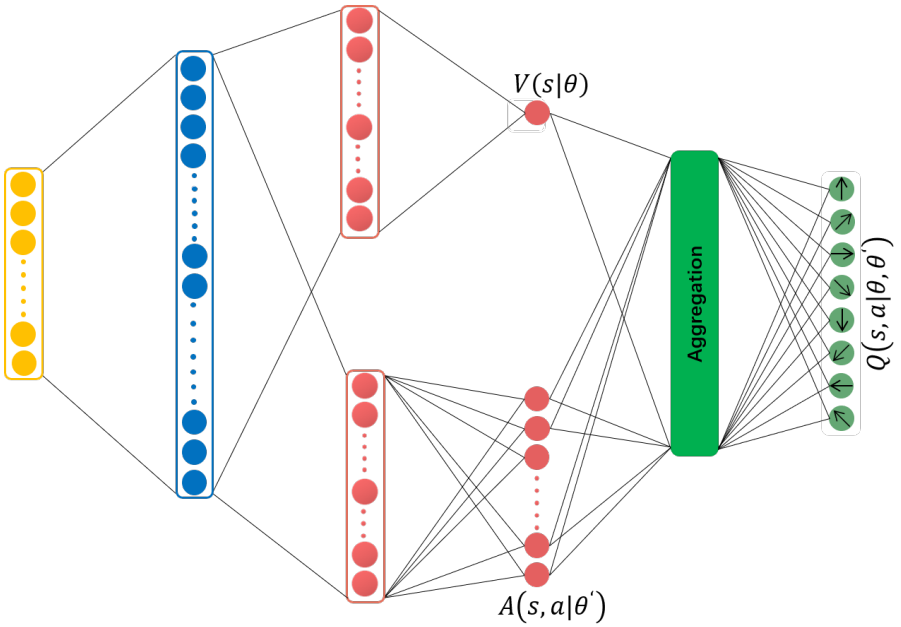


Fig. 5 Schematic illustration of the network architecture.

In our homogeneous MAS architecture, all agents forming the multi-agent system have the same internal architecture. Therefore, we take advantage of the homogeneity of agents and learn in a parameter sharing paradigm. In this setting, we allow all the agents to share the parameters of a single policy. This allows the policy to be trained with the experiences of all agents simultaneously. However, the agents can still behave differently because they receive different observations and thus evolve different hidden states. In addition, each agent receives its own index a as input, allowing it to specialize. The rich representations in deep Q-networks can facilitate the learning of a common policy while also allowing for specialization. Parameter sharing also dramatically reduces the number of parameters that must be learned. Moreover, when similar agents are learning similar behaviors, their parameters can be shared to enhance the speed of learning and to decrease the complexity and resource utilization of the algorithm. During decentralized execution, each agent uses its own copy of the learned network, evolving its own hidden state, selecting its own actions, and indirectly communicating with other agents only through modifying the environment.

4.2 Digital pheromones coordination mechanism

Stigmergy was 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. This generic definition applies to very broad cases, including

path planning in insect colonies, where agent coordination through a shared environment as well as medium by the use of pheromones. For example, the networks of paths that they construct joining their nests with available food sources form minimum spanning trees, minimizing the energy ants expend in bringing food into the nest. This structure emerges as individual ants wander, depositing, and sensing pheromones. In a natural ant colony, ants operate on chemical pheromones that support purposive actions. It aggregates deposits from individual agents, fusing information across multiple agents, and through time. And it evaporates pheromones over time. Traditionally, knowledge-based approaches expend large amounts of computation in an effort to detect inconsistencies that result from changes in the domain being modeled but ants do not. On the contrary, by evaporating, ants immediately begin to forget everything they learn, unless it is continually reinforced. Thus inconsistencies automatically remove themselves within a known period. It also diffuses pheromones to nearby places, disseminating information for access by nearby agents.

The pheromone filed constructed by the ants in the environment is, in fact, a potential field that guides their movements. 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. This kind of synthetic pheromones is digital data structures, not chemicals. 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, those that are responsible for adjacent regions of space, and it exchanges local pheromone information with them. And pheromone is quantitative so that it perceived conditions that differ in strength or degree, and where stronger traces typically elicit more forceful actions. This quantitative variation can be captured using conditional probability: the stronger the trace, the higher the probability of a certain action given that trace. Ant trail laying follows this quantitative logic. The stronger the scent of pheromone on a trail, the less likely an ant is to deviate from that trail, and therefore the higher the probability that it too will reinforce the trail with additional pheromone.

Pheromone is the classical example of stigmergy, which is used to coordinate actions via the trace left in a medium, can explain self-organizing activities in social insects like ants. When considering stigmergic coordination between different agents, medium as that part of the world that is controllable and perceivable by all of them. In our setting, we look upon the digital pheromone network as the medium that is so passive that undergoes shaping by the actions but does not participate in the activity itself. The role of the digital pheromone network is to allow interaction or communication between different actions and thus, indirectly, between the agents that perform the actions. It is this mediating function that underlies the true power of the digital pheromone network.

By using the digital pheromone network in a stigmergic state space, each agent can perceive the amount of digital pheromone within a certain range. Here, we regard every tiny patch of the environment as a network node. And

they are filled with digital pheromones as "attractor" in the local environment. It is attractive to the agents nearby and can provide effective observation about the local state space. The aforementioned mechanism is illustrated in Fig.6. Just like ant colony optimization (ACO) [7] applied to traveling salesperson problem (TSP), each ant chooses the next node probabilistically within a collection of cities based on pheromone value. We modify this process to fit our situation. In 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 [7], 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 in the local environment to enables 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:

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

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.

When a dynamic change occurs in the environment, some of the current pheromone trails will misguide the search towards poor areas of the search space containing low-quality solutions. That is to say, some previously generated pheromone trails may lead to a possibly poor solution for the new environment. For this reason, ACO 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 $\forall j : \varepsilon_{\min} < \varepsilon_j < \varepsilon_{\max}$.

For the sake of maintaining stability and guaranteeing convergence for swarming agents, similar to the law of conservation of energy, we apply three constraints on digital pheromone to ensure its conservation.

- Local Stability: The concentration of the output diffused from any set of nodes to their neighbors at time step $t+1$ is strictly less than the concentration of the aggregate input (external plus deposit) to those nodes at time step t.
- Diffused Stability: There exists a fixed upper limit to the aggregated sum of all diffused inputs at an arbitrary node if a one-time and one-place external input is assumed.
- Global Stability: The digital pheromone concentration in any node is bounded.

Similar to MMAS, our implementation also includes two bounds to limit the digital pheromone value. The lower and upper bounds $\varepsilon_{\min}$ and $\varepsilon_{\max}$ of the digital pheromone concentration are imposed such that $\forall j : \varepsilon_{\min} < \varepsilon_j < \varepsilon_{\max}$. If the concentration of the digital pheromone at a node below $\varepsilon_{\min}$, it will be treated as 0. Also, once the digital pheromone concentration reaches the upper limit $\varepsilon_{\max}$, it will no longer increase.

5 Experiments

In this section, we describe the Minefield Navigation Environment that using to perform our experiments, as well as demonstrate the effectiveness of PCDQN, its superiority for the multiagent path search problem.

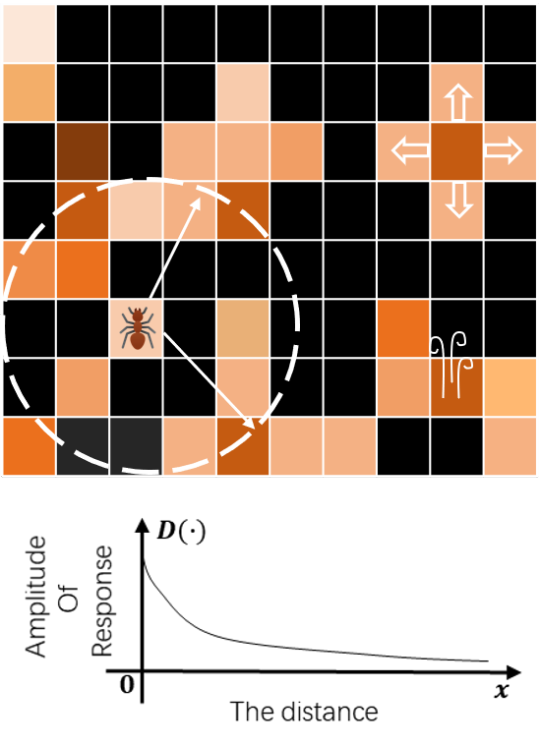


Fig. 6 The digital pheromone network.

5.1 Minefield Navigation Environment

In this paper, we adopt the minefield navigation environment that is a two-dimensional world with discrete space and time, consisting of N agents and M obstacles. Akin to the real-time strategy game StarCraft II, Minefield Navigation Environment also focus on micromanagement challenges where 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). The objective is to navigate through a minefield to a randomly selected target position in a specified time frame without hitting a mine. For each episode, agents start at randomly chosen positions in the minefield and then detect their surroundings. An episode terminates when all agents have either died or arrived at their own target, or when a pre-specified time limit is reached. The target and the mines remain stationary during the trial.

5.1.1 State and Observation

At each timestep, 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

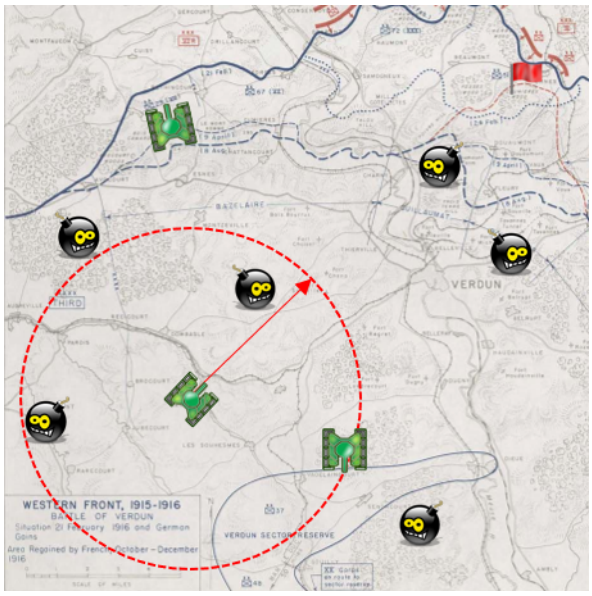


Fig. 7 The red circle borders the sensing range of the agent.

information about obstacles and other agents within a circular area around the unit and with a radius equal to the sensing range. The sensing range makes the environment partially observable from the standpoint of each agent.

In a general way, the feature vector observed by each agent contains the following attributes: sensor signal, relative position to its destination, attractor location, and its identification number. Additionally, the state vector can also include other features like the coordinates of the leader in multiagent formation. And 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.

5.1.2 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.

5.1.3 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. We adopt reward shaping to solve sparse reward problems in our experiments.

The overall goal is to maximize the success rate for arriving at destinations. Our setting is to use the shaped reward, which produces a reward based on whether the agent reaches its goal, hits a mine, collides with others, and so on. Reward functions vary according to the different experiments we considered.

5.2 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 minefield navigation environment containing 15 mines.

5.2.1 Experimental Setup

Following reward shaping, 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 $r_attractor^k$. Details about reward function r_t^k are exhibited schematically in Fig.8.

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:

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

where $\sigma \in [0, 1]$ determines the importance of sensory signal s_i, as well as r_{close}^k in our shaped reward function. The r_{stop}^k 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 the agent at the current moment are the same as the previous moment, a certain penalty will be given to it. And considering the overall goal, the agent k cannot just focus on its own interests so that r_{range}^k is defined by the sum of the distance between every agent and its destination as follows:

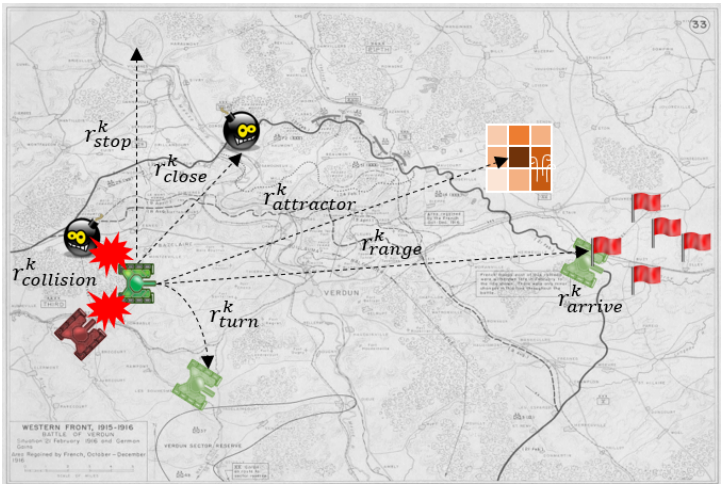


Fig. 8 The schematic diagram of shaped reward r^k_t for agent k.

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

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.

5.2.2 Results

Each agent learns from scratch based on the feedback signals received from the environment. We run the experiments for 2000 trials. In each trial, the initial locations of agents, the location of their destinations, and the positions of mines are randomly set. To evaluate PCDQN's performance, we adopt the following evaluation procedure: for each run of the method, we evaluate training at 100-trial intervals within 2000 trials. The success rate of a multiagent system in a trial is determined by the percentage of agents that reach their target positions within the specified time steps in the trial. 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.

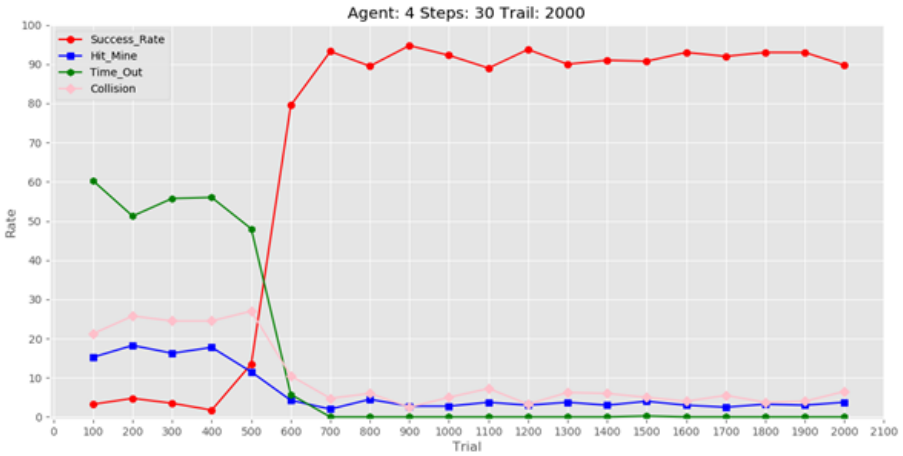


Fig. 9 Success rate and failure (hit mine, time out, and collision) rate of the PCDQN multiagent system averaged at 100-trial intervals on the minefield collaboration task.

Fig.9 shows the average success and failure (hit mine, time out, and collision) rate as a function of trials for PCDQN. We can see that the success rate of multiagent is highly unstable in the beginning but then increase rapidly after 400 trials. Our agent's learning algorithm is based on DQN [20] and includes its techniques of experience replay and the target network. Agents apply minibatch updates to samples of experience, drawn at random from the pool of stored samples. At the beginning of training, the experience pool has not yet stored enough experiences to learn so that the neural network cannot be updated sufficiently. By the end of 700 trials, the multiagent system achieves more than a 90% success rate. 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 progresses 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 the right 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 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 Figure 10.

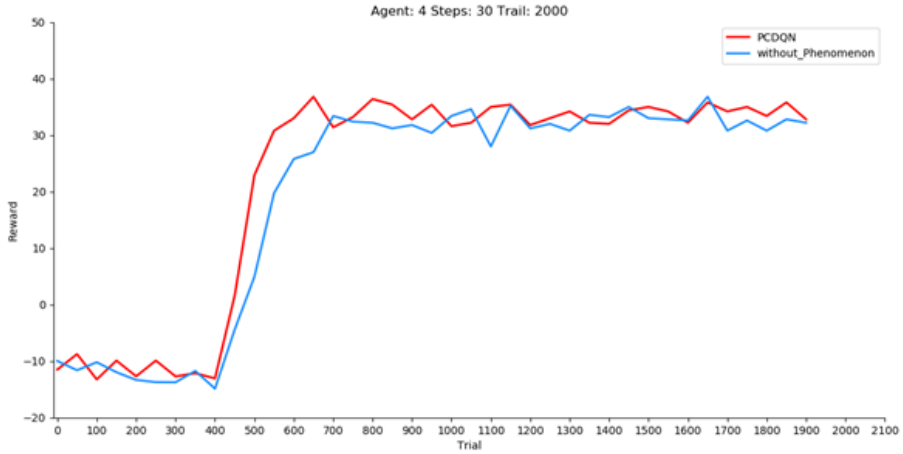


Fig. 10 The accumulated reward curves for PCDQN with (red) and without (blue) the digital pheromones coordination mechanism.

The very clear conclusion is that DQN architecture mixed with the digital pheromones coordination mechanism performs much better than individual DQN. Fig.10 shows that the PCDQN agents learn much faster than agents without pheromones guide. For example, without the digital pheromones coordination mechanism, the multiagent system takes about 700 trials to achieve its peak performance. In contrast, the PCDQN multiagent system achieves its peak at just 600 trials. And in terms of stability, PCDQN agents maintain relatively more stability after convergence in the later stage of training. But to the lack of pheromone guidance (blue curve in Fig.10), some strong fluctuations will occur due to environmental instability after agents converge to their optimum.

Fig.11 shows the searched navigation paths in the multiagent collaboration task by using the PCDQN framework, in the Minefield Navigation Environment. 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 sample one trial within 2000 trials and display digital pheromone concentration at 2-step intervals within 30 time steps in Fig.12.

As Fig.12 has shown, at the beginning of the trial, the digital pheromone concentration is 0 everywhere. After several time steps, some of the agents reach their goals and release digital pheromone. And then these digital pheromone diffuses into their surroundings to guide other agents to achieve their targets faster. The digital pheromone concentration clearly increases as time goes by

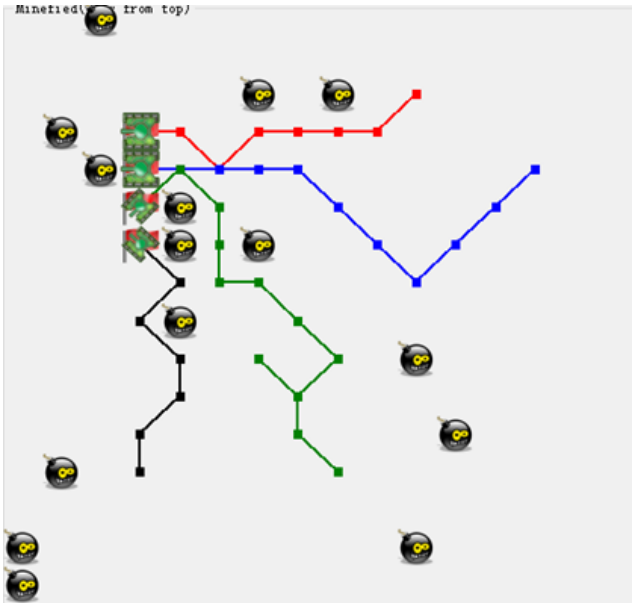


Fig. 11 The searched navigation paths in the multiagent collaboration task.

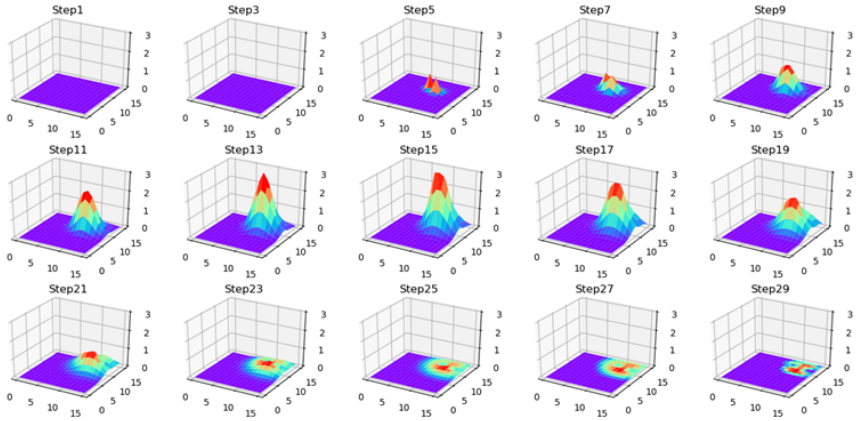


Fig. 12 The change of digital pheromone concentration during training for a single trial.

and achieves its peak at about time step 15. At this point, almost all agents reach their destination successfully. After 15 time steps, the digital pheromone gradually evaporates and drops to 0 at the end, which corresponds with the phenomenon observed in the natural ant colony.

5.3 Multiagent Path Search

A significant amount of research efforts have been focused on the control of multiagent systems due to both their practical potential in various applications and theoretical challenges arising in coordination and control of them. Formation control is one of the most actively studied topics among multiagent systems, which aims to control the relative position and orientation of the robots in a group while allowing the group to move as a whole. Its theoretical challenge mainly arises from controlling a formation system based on a partially observable environment without the intervention of a central controller. In order to demonstrate the superior performance of PCDQN on the multiagent system, we build a simulation scenario where mobile agent formation navigates through a minefield to a randomly selected target position in a limited time without hitting a mine. This simulation scenario including a certain multiagent formation lies in the 16×16 minefield navigation environment containing 15 mines.

5.3.1 Experimental Setup

Among robot formation control schemes, leader-follower architectures are particularly appreciated for their simplicity and scalability, where a robot of the formation designed as the leader, moves along a predefined trajectory, while the other robots, the followers, keep desired distance and orientation with the leader. Inspired by the leader-follower approach, we designed distinctive multiagent formations in our simulations. Rather than prescribing a track for the leader, we let the leader search path based on local observation.

In our experiments, we select the agent as our leader whose identification number is 0 as well as the other agents are followers. Furthermore, we amend immediate reward r_t^k in Eq.15 by augmenting the additional constraint item $r_{formation}^k$. And the $r_{formation}^k$ consisted of $r_{f_1}^k$ and $r_{f_2}^k$ is defined as follows:

$$r_{formation}^k = r_{f_1}^k + r_{f_2}^k \tag{15}$$

where $r_{f_1}^k$ and $r_{f_2}^k$ are distance and orientation constraints relative to the leader respectively, shown in Fig.13. Therefore, $r_{f_1}^k$ is determined by the distance between agent k and the leader agent 0 and defined as $r_{f_1}^k = -\chi \cdot |d_{k,0} - d_0|$, where $\chi \in [0, 1]$ and d_0 is prescribed offset when the follower track the leader's trajectory. Likewise, $r_{f_2}^k = -\chi \cdot |\Delta Align(k, 0)|$, where $\chi \in [0, 1]$ and $\Delta Align(k, 0)$ is declination between agent k and the leader agent 0.

To incorporate both the individual payoff and formation constrains, the reward function of agent k at time step t is defined as follows:

$$\tilde{r_t^k} = r_t^k + r_{formation}^k \tag{16}$$

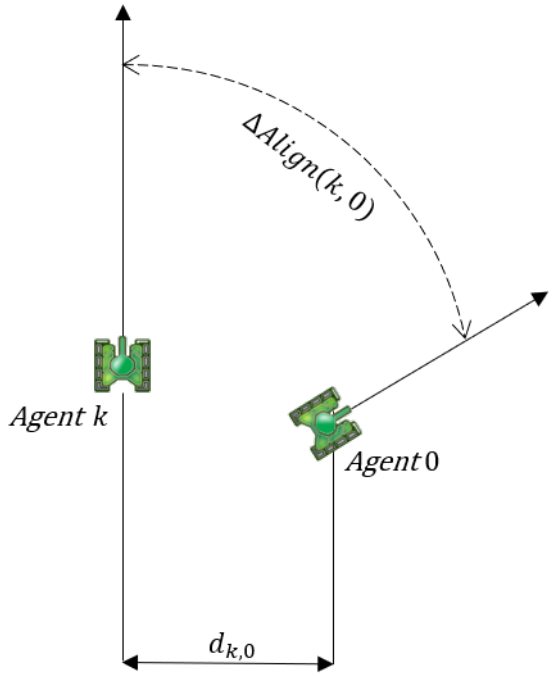


Fig. 13 The schematic diagram of $r^k_{f_1}$ and $r^k_{f_2}$ for agent k.

5.3.2 Results

In this subsection, we present results for multiagent scenarios. The purpose of these results is to demonstrate the performance of the PCDQN framework in solving complex tasks. For this reason, we fully implement and test our algorithm in the multiagent formation control domain for the path search problem. However, in this formation control task, agents are required to coordinate with other teammates through an autonomous exploration of the environment. Apparently, learning this complex task from scratch is impractical due to the huge sample complexity of RL algorithms. The intuitive solution is to reuse knowledge like a human to utilize principles learned in previous tasks to alleviate the burden of learning. Since the similarity between the previous task in subsection B and the current one, we utilize its model as pre-training to speed up learning. This idea is inspired by transfer via inter-task mapping (TVITM) [35], agents are able to learn one task and then markedly reduce the time it takes to learn a more complex task.

Two formations for a team of eight agents are considered (Fig.14):

- Square - where the agents travel in a square;
- Arrow - where the agents travel in an arrow.

For each formation, each agent has a specific position based on its identification number. Fig.14 shows the formations and agents' positions within them.

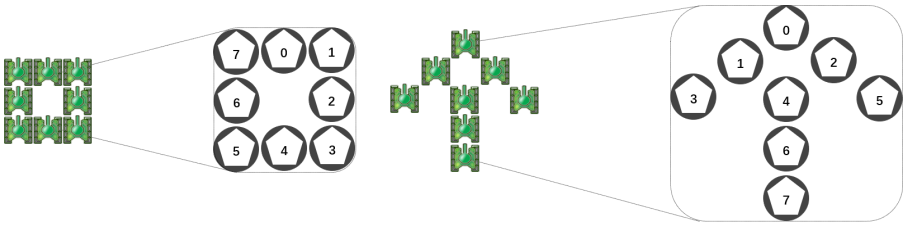


Fig. 14 Formations for eight agents

Designed agent 0 as the leader, while the other agents, the followers, are to maintain a desired distance and orientation to the leader.

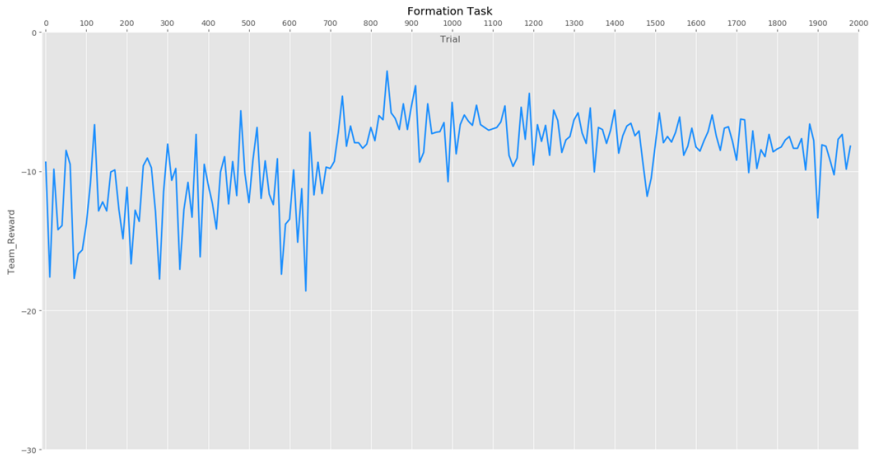
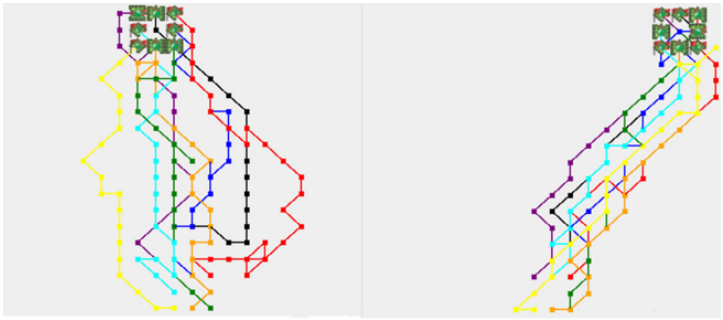


Fig. 15 The accumulated reward curve for multiagent formation control tasks, which shows the total penalty the whole formation received during training.

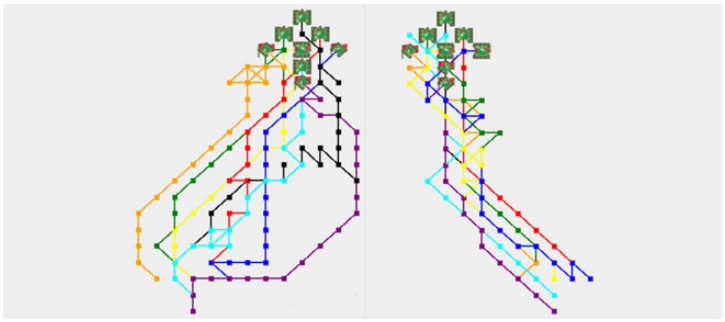
We especially estimate the additional constraint $r^k_{formation}$ for formation tasks during training showed in Fig15. As the training progresses, the penalty score of the whole team increases from about -15 to about -8 and tends to be stable after 2000 trials. Despite slight fluctuations, the PCDQN teammates have learned to coordinate and converge gradually to the optimum.

We perform ablation experiments to validate the effectiveness of the digital pheromone network in PCDQN by comparing moving trajectories of agents.

Learning with the PCDQN framework has the obvious advantage when comparing with not considering the digital pheromones coordination mechanism. Clearly, it can be seen from Fig.16 that PCDQN multiagent formations have more concentrated and straightforward trajectories. Overall, PCDQN is a better performing and more general method.



(a) Trajectories of the Square formation.



(b) Trajectories of the Arrow formation.

Fig. 16 Moving trajectories of multiagent formations for PCDQN without (left) and with (right) the digital pheromones coordination mechanism.

6 Conclusion

This paper presented PCDQN, a deep multiagent RL method that allows end-to-end learning of decentralized policies and makes efficient use of digital pheromones coordination mechanism. PCDQN combines the advantage of both sides, independent DQN guides agents to learn successfully control policies, while the digital pheromones network integrates diverse pheromone sources, process pheromone in a completely distributed environment, and cope with dynamic changes occurring in the environment. Further, the method can be nicely combined with shaped reward functions, so that making agents adapt to various practical tasks efficiently.

Our results in the multiagent collaboration task in the minefield navigation environment 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 addition, multiagent formation control experiments demonstrate that PCDQN agents are competent for complex tasks, and simulation results have verified the effectiveness of the digital pheromones coordination mechanism with a significant improvement in the final performance over the independent DQN as well as its scalability.

In the near 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. Moreover, developing more sample-efficient variants that are practical for real-world applications such as robot swarms is our goal in the long term.

References

1. Bagnell, D., Ng, A.: On local rewards and scaling distributed reinforcement learning. Advances in Neural Information Processing Systems **18**, 91–98 (2005)
2. Bansal, T., Pachocki, J., Sidor, S., Sutskever, I., Mordatch, I.: Emergent complexity via multi-agent competition. arXiv preprint arXiv:1710.03748 (2017)
3. Beckers, R., Holland, O.E., Deneubourg, J.L.: Fom local actions to global tasks: Stigmergy and collective robotics. In: Prerational Intelligence: Adaptive Behavior and Intelligent Systems Without Symbols and Logic, Volume 1, Volume 2 Prerational Intelligence: Interdisciplinary Perspectives on the Behavior of Natural and Artificial Systems, Volume 3, pp. 1008–1022. Springer (2000)
4. Buşoniu, L., Babuška, R., De Schutter, B.: Multi-agent reinforcement learning: An overview. Innovations in multi-agent systems and applications-1 pp. 183–221 (2010)
5. Correia, L., Sebastião, A.M., Santana, P.: On the role of stigmergy in cognition. Progress in Artificial Intelligence **6**(1), 79–86 (2017)
6. Dorigo, M., Bonabeau, E., Theraulaz, G.: Ant algorithms and stigmergy. Future Generation Computer Systems **16**(8), 851–871 (2000)
7. 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)
8. Foerster, J.N., Assael, Y.M., De Freitas, N., Whiteson, S.: Learning to communicate with deep multi-agent reinforcement learning. arXiv preprint arXiv:1605.06676 (2016)
9. Gentzsch, W.: Sun grid engine: Towards creating a compute power grid. In: Proceedings First IEEE/ACM International Symposium on Cluster Computing and the Grid, pp. 35–36. IEEE (2001)
10. Grassé, P.P.: La reconstruction du nid et les coordinations interindividuelles chezbellicositermes natalensis etcubitermes sp. la théorie de la stigmergie: Essai d'interprétation du comportement des termites constructeurs. Insectes sociaux **6**(1), 41–80 (1959)
11. Guide, A.: Portable batch system (1998)
12. Heylighen, F.: Stigmergy as a universal coordination mechanism i: Definition and components. Cognitive Systems Research **38**, 4–13 (2016)
13. Jiang, J., Lu, Z.: Learning attentional communication for multi-agent cooperation. arXiv preprint arXiv:1805.07733 (2018)
14. Kassabalidis, I., El-Sharkawi, M., Marks, R., Arabshahi, P., Gray, A.: Swarm intelligence for routing in communication networks. In: GLOBECOM'01. IEEE Global Telecommunications Conference (Cat. No. 01CH37270), vol. 6, pp. 3613–3617. IEEE (2001)
15. Kim, W., Cho, M., Sung, Y.: Message-dropout: An efficient training method for multi-agent deep reinforcement learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 6079–6086 (2019)
16. Laurent, G.J., Matignon, L., Fort-Piat, L., et al.: The world of independent learners is not markovian. International Journal of Knowledge-based and Intelligent Engineering Systems **15**(1), 55–64 (2011)
17. Leibo, J.Z., Zambaldi, V., Lanctot, M., Marecki, J., Graepel, T.: Multi-agent reinforcement learning in sequential social dilemmas. arXiv preprint arXiv:1702.03037 (2017)
18. Matignon, L., Laurent, G.J., Le Fort-Piat, N.: Independent reinforcement learners in cooperative markov games: a survey regarding coordination problems. Knowledge Engineering Review **27**(1), 1–31 (2012)

19. 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)
20. Moradi, M.: A centralized reinforcement learning method for multi-agent job scheduling in grid. In: 2016 6th International Conference on Computer and Knowledge Engineering (ICCKE), pp. 171–176. IEEE (2016)
21. Ng, A.Y., Harada, D., Russell, S.: Policy invariance under reward transformations: Theory and application to reward shaping. In: Icml, vol. 99, pp. 278–287 (1999)
22. Oliehoek, F.A., Amato, C.: A concise introduction to decentralized POMDPs. Springer (2016)
23. Palmer, G., Tuyls, K., Bloembergen, D., Savani, R.: Lenient multi-agent deep reinforcement learning. arXiv preprint arXiv:1707.04402 (2017)
24. Peng, P., Wen, Y., Yang, Y., Yuan, Q., Tang, Z., Long, H., Wang, J.: Multiagent bidirectionally-coordinated nets: Emergence of human-level coordination in learning to play starcraft combat games. arXiv preprint arXiv:1703.10069 (2017)
25. Pesce, E., Montana, G.: Improving coordination in multi-agent deep reinforcement learning through memory-driven communication. arXiv preprint arXiv:1901.03887 (2019)
26. Ramos, V., Merelo, J.J.: Self-organized stigmergic document maps: Environment as a mechanism for context learning. arXiv preprint cs/0412075 (2004)
27. Schaul, T., Quan, J., Antonoglou, I., Silver, D.: Prioritized experience replay. arXiv preprint arXiv:1511.05952 (2015)
28. Sen, S., Weiss, G.: Learning in multiagent systems. Multiagent systems: A modern approach to distributed artificial intelligence pp. 259–298 (1999)
29. Shamshirband, S., Anuar, N.B., Kiah, M.L.M., Patel, A.: An appraisal and design of a multi-agent system based cooperative wireless intrusion detection computational intelligence technique. Engineering Applications of Artificial Intelligence **26**(9), 2105–2127 (2013)
30. Stone, P., Veloso, M.: Multiagent systems: A survey from a machine learning perspective. Autonomous Robots **8**(3), 345–383 (2000)
31. Sukhbaatar, S., Szlam, A., Fergus, R.: Learning multiagent communication with backpropagation. arXiv preprint arXiv:1605.07736 (2016)
32. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction, second edn. The MIT Press (2018). URL http://incompleteideas.net/book/the-book-2nd.html
33. Tampuu, A., Matiisen, T., Kodelja, D., Kuzovkin, I., Korjus, K., Aru, J., Aru, J., Vicente, R.: Multiagent cooperation and competition with deep reinforcement learning. PloS one **12**(4), e0172395 (2017)
34. Tan, M.: Multi-agent reinforcement learning: Independent vs. cooperative agents. In: Proceedings of the tenth international conference on machine learning, pp. 330–337 (1993)
35. Taylor, M.E., Stone, P., Liu, Y.: Transfer learning via inter-task mappings for temporal difference learning. Journal of Machine Learning Research **8**(9) (2007)
36. 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)
37. 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)
38. 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)
39. 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)