Multiagent Coordination in Roombas: From a Neural Network Perspective

Jimmy Xin Lin and **Barry Feigenbaum**

***Abstract*—The abstract goes here.**

I. Introduction

Most of existing literatures about Roomba is qualitative.

II. Related Works

A. Coordinated Multiagent Reinforcement Learning

[1] Existing work typically assumes that the prob- lem in each time step is decoupled from the problems in other time steps, which might not hold in some applications. Therefore, in this paper, we make the following contributions: (i) We introduce a new model, called Markovian Dynamic DCOPs (MD-DCOPs), where the DCOP in the next time step is a function of the value assignments in the current time step; (ii) We introduce two distributed reinforcement learning algorithms, the Distributed RVI Q-learning algorithm and the Distributed R-learning algorithm, that balance exploration and exploitation to solve MD-DCOPs in an online manner; and (iii) We empirically evaluate them against an existing multi-arm bandit DCOP algorithm on dynamic DCOPs.

[2] This paper presents a model-free, scalable learning approach that synthesizes multi-agent reinforcement learning (MARL) and distributed constraint optimization (DCOP). By exploiting structured interaction in ND-POMDPs, our approach distributes the learning of the joint policy and employs DCOP techniques to coordinate distributed learning to ensure the global learning performance. Our approach can learn a globally optimal policy for ND-POMDPs with a property called groupwise observability. Experimental results show that, with communication during learning and execution, our approach significantly outperforms the nearly-optimal non-communication policies computed offline.

SBDO: A New Robust Approach to Dynamic Distributed Constraint Optimisation

[3]

[4]

[5]

[] Complex problems involving multiple agents exhibit varying degrees of cooperation. The levels of cooperation might reflect both differences in information as well as differences in goals. In this research, we develop a general mathematical model for distributed, semi-cooperative planning and suggest a solution strategy which involves decomposing the system into subproblems, each of which is specified at a certain period in time and controlled by an agent. The agents communicate marginal values of resources to each other, possibly with distortion. We design experiments to demonstrate the benefits of communication between the agents and show that, with communication, the solution quality approaches that of the ideal situation where the entire problem is controlled by a single agent.

[6] Researchers in the field of Distributed Artificial Intelligence (DAI) have been developing efficient mechanisms to coordinate the activities of multiple autonomous agents. The need for coordination arises because agents have to share resources and expertise required to achieve their goals. Previous work in the area includes using sophisticated information exchange protocols, investigating heuristics for negotiation, and developing formal models of possibilities of conflict and cooperation among agent interests. In order to handle the changing requirements of continuous and dynamic environments, we propose learning as a means to provide additional possibilities for effective coordination. We use reinforcement learning techniques on a block pushing problem to show that agents can learn complimentary policies to follow a desired path without any knowledge about each other. We theoretically analyze and experimentally verify the effects of learning rate on system convergence, and demonstrate benefits of using learned coordination knowledge on similar problems. Reinforcement learning based coordination can be achieved in both cooperative and non-cooperative domains, and in domains with noisy communication channels and other stochastic characteristics that present a formidable c

B. Coordinated Multiagent Neuroevolution

Barry: Add literatures of Neuroevolution here...

III. Problem Formulation

A. Structure of Roomba Environment

The Roomba environment, under the OpenNERO platform [7], is a virtual computer lab with crumbs distributed on the floor. In this virtual lab, there are four classes of objects: agents, crumbs, walls, decorations. Vacuum cleaner agents, shown as grey cylinders in Fig. 1, are supposed to collect the static crumbs that are labelled as blue cubes. The agents will be rewarded if they move to a place where there are some crumbs. Walls are also set up as the limits of the Roomba environment, such that agents are not allowed to move beyond the walls and no pellets can be put outside the walls. Other decorative objects within the computer lab include tables, chairs, and computers. For simplicity at the moment, these decorations only serve as physically transparent decorations, which means they do not block agents' movements.

As shown in the Fig. 1, the small window floating on the top right is the controller that masters the type of scripted AI agents to load in, the number of agents, and particular commands that impact the progress of the Roombas simulation (Pause/Resume, Add/Remove Robots, Exit). On each run of the simulation, only one particular type of AI agents are allowed to be loaded. Note also that the type of AI agents employed cannot be switched to the other one during the intermediate process of crumb collection task. If one needs to switch to the other type of AI agents, the running agents have to be removed first and then the user can load the desired AI agents.

In the Roombas environment, the movements of vacuum cleaner agents are not constrained by four directions (left, right, forward, backward). Instead, agents are able to move towards all directions and each moving action is denoted as a continuous radius value. Besides, all agents move in a synchronous way. Agents are allowed to make their movements of the following step if and only if all agents have completed their movements in the last step.

Three different modes (MANUAL, RANDOM, and CLUSTER) are employed to specify the placing position of each individual crumb. In the MANUAL mode, one pellet is deterministically placed at a user-specified position. In the RANDOM mode, the placement of one pellet is totally randomized; the environment will throw a rejection and repeat the random generation, if an invalid position is yielded. The last one is the CLUSTER mode, where the position of pellets are partially randomized. In this mode, environment samples the position of one pellet from a gaussian distribution whose centers and spreads at x and y coordinates are specified. In this paper, the specifications for the placement of crumbs are generated from the starting CLUSTER-mode experiment and remained invariant by employing MANUAL mode in all experiments afterwards.

What each agent could perceive from the computer lab is limitless. The implementation of Roomba environment allows each agent to sense all crumbs on the floor about their positions, existence status, and even rewards. In addition, the spatial information of other vacuum cleaner robots is available for each individual agent, as well as the user-defined working status of these collaborated robots. Although the Roomba environment provides a large design space for sensors, it is a practical treatment to start from a small number of simple sensors. For example, a combination of the bumping status, the position of its own, and the location of closest crumb suffice for an agent to learn a greedy strategy, as illustrated in the initial setup experimentation .

The reward design is another big issue for configuring the Roomba environment. By default, the only reward being set up for agents is when they successfully collect some pellets on the floor. In order to facilitate the learning of agents, an penalty for being alive is supposed to be incorporated to the reward system. That is, agents should receive some negative rewards, typically a very small quantity, for each step they move. Similarly, penalties can also be granted to the collisions between roombas, bumping of agents towards the world boundaries, and repetitive movements around an area.

Fig. 1. Overall Picture of the Roomba Environment

B. Expected Multiagent Behaviors

C. Difficulties and Challenges

IV. Experiments: Initial Setup

This section will present some preliminary results coming from the simulation of greedy agents. Agents with greedy strategy simply approaches to the direction where the closest pellet to it is there. These results may not be directly related to the multiagent behaviors, but (i) demonstrate that we have set up the environment correctly from both reinforcement learning and neuroevolution. (ii) illustrate intuitively the benchmark intelligence for the crumb collection.

A. Reinforcement Learning

Add implementation details of Q-Learning and variants here...

B. Neuroevolution

Barry: Add implementation details of Neuroevolution here...

V. Experiments: Multiagent Coordination

After the experimentation for testing the initial setup of Roombas environment, the intention of this section is to investigate the problem of how to incorporate effective coordinations for this particular multiagent system.

A. Reinforcement Learning

B. Neuroevolution

VI. Conclusions

The conclusion goes here.

Future Works go here.

REFERENCES

[1] D. T. Nguyen, W. Yeoh, H. C. Lau, S. Zilberstein, and C. Zhang, "Decentralized multi-agent reinforcement learning in average-reward dynamic dcops," in *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2014, pp. 1341–1342.

[2] C. Zhang and V. R. Lesser, "Coordinated multi-agent reinforcement learning in networked distributed pomdps." in *AAAI*, 2011.

[3] C. Zhang and V. Lesser, "Coordinating multi-agent reinforcement learning with limited communication," in *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2013, pp. 1101–1108.

[4] B. Banerjee, J. Lyle, L. Kraemer, and R. Yellamraju, "Sample bounded distributed reinforcement learning for decentralized pomdps." in *AAAI*, 2012.

[5] L. Kraemer and B. Banerjee, "Informed initial policies for learning in dec-pomdps." in *AAAI*, 2012.

[6] S. Sen, M. Sekaran, J. Hale *et al.*, "Learning to coordinate without sharing information," in *AAAI*, 1994, pp. 426–431.

[7] I. Karpov, J. Sheblak, and R. Miikkulainen, "Opennero: A game platform for ai research and education." in *AIIDE*, 2008.