Alternating Maximization with Behavioral Cloning

Aleksander Czechowski, Frans A. Oliehoek
Delft University of Technology
{a.t.czechowski, f.a.oliehoek}@ tudelft.nl

Introduction

The key difficulty of cooperative, decentralized planning lies in making accurate predictions about the behavior of one's teammates. In [1] we introduce a planning method of *Alternating maximization with Behavioural Cloning* (ABC) – a trainable online decentralized planning algorithm based on Monte Carlo Tree Search (MCTS), combined with models of teammates learned from previous episodic runs. Our algorithm relies on the idea of alternating maximization, where agents adapt their models one at a time in round-robin manner. Under the assumption of perfect policy cloning, and with a sufficient amount of Monte Carlo samples, successive iterations of our method are guaranteed to improve joint policies, and eventually converge. We apply the algorithm in a simple sequential multi-robot task allocation domain considered in [2, 3], and show that it improves over the existing baselines from [3] for particularly challenging domain configurations.

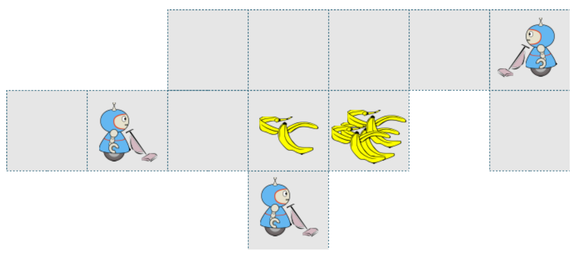


Figure 1: The Factory Floor task allocation domain.

Preliminaries & Notation

- Multi–agent Markov decision process (MMDP) on n agents: (S, A, T, R), S – state space, $T : S \times A \times S \to \mathbb{R}$: transition probabilities, $A = A_1 \times \cdots \times A_n$: factored action space, $R : S \times A \to \mathbb{R}$: common reward, γ – discount factor
- MMDP policy: $\pi = (\pi_1, \ldots, \pi_n) : S \to A$,
- optimal joint policy $\pi^* = \text{argmax}_\pi \mathbb{E}(\sum_t \gamma^t R_t | \pi)$
- individual best responses of agent i to other agents $-i$: $\pi_i = BR_i(\pi_{-i}) = \text{argmax}_{\pi_i} \mathbb{E}(\sum_t \gamma^t R_t | \pi_i, \pi_{-i})$
- Monte Carlo Tree Search for agent i with respect to policies of other agents $\pi_i = MCTS(\pi_{-i}) \approx BR_i(\pi_{-i})$

The ABC method

Our planning algorithm is suitable for fully observable cooperative environments known as Multi-agent Markov Decision Processes (MMDPs). The setting is fully cooperative, and each agent is assumed to receive the same reward at each execution step of an episodic run. The planning is performed in a decentralized manner, and without communication between the agents. Each agent is equipped with an instance of the MCTS algorithm, a set of models of policies of its teammates, and a simulator of the environment. At each episodic step, each agent samples the simulator and teammate models to construct the tree of possible futures, estimate expected episodic rewards for individual actions, and choose the one which appears most beneficial.

Initially, agents are equipped with heuristic models of their teammates. They are assumed to act in a given environment repeatedly, for some large amount of episodic runs – either from simulation, or actual execution. Then, the agents use these experiences to learn to predict the actions of their colleagues. More specifically, every N episodic runs are grouped into one generation, and after each generation, the state-action episodic data, is used to train new agent models represented by convolutional neural networks; these are in turn provided to one of the agents, as the updated teammate models. At each generation only one agent updates its teammate models, which stabilizes training and, under certain assumptions on policy cloning, causes rewards to increase monotonically across the generations.

Pseudocode

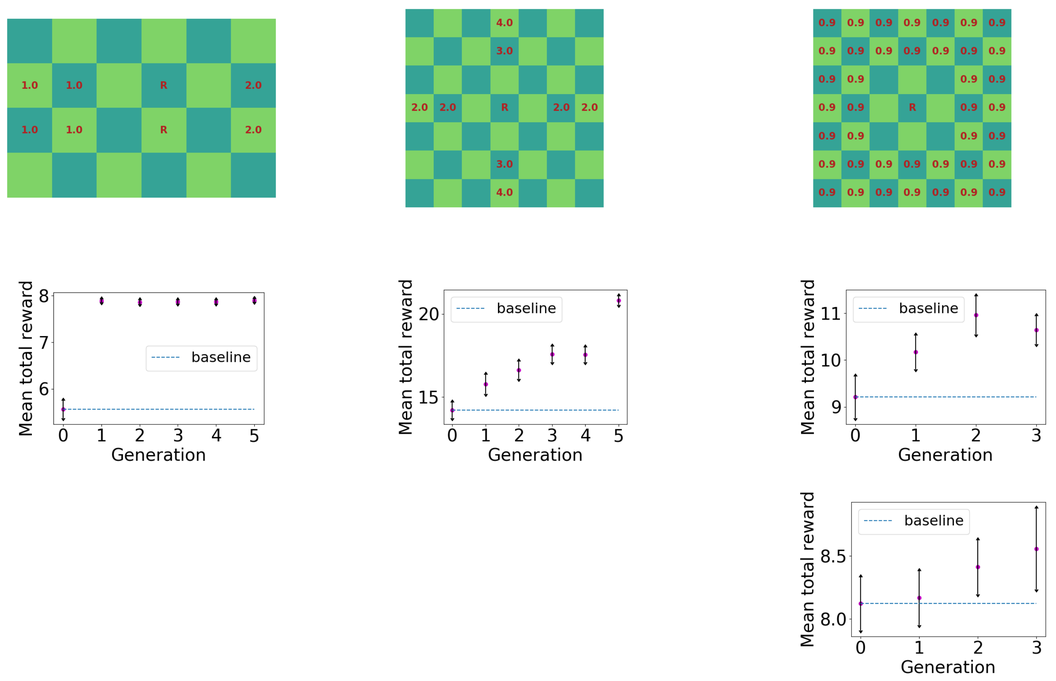
Inputs: MMDP M
initial heuristic policy models of all agents $\pi^{h,0} = (\pi_1^{h,0}, \ldots, \pi_n^{h,0})$
Start:

$\forall i : \pi_i^0 := MCTS\left(\pi_{-i}^{h,0}\right)$ //all agents with heuristic teammate models
for g in 1:nGenerations **do**
  perform simulation with π^{g-1}, collect data d
  $\forall i$: train new teammate models $\pi_i^{h,g-1} \approx \pi_i^{g-1}$
  $\forall i : \quad \pi_i^g := \pi_i^{g-1}$ //copy previous policies of all agents
  $j := (g \bmod n) + 1$ //one agent to update learnt models
  $\pi_j^g := \text{MCTS}\left(\pi_{-j}^{h,g-1}\right)$
end for

Experiments

We test the efficiency of the algorithm by performing experiments in the spatial task allocation environment introduced in [2]. The domain consists of a gridworld-like planar map, where each position can be occupied by (cleaning) robots and tasks (e.g. litter). Each robot can perform either a movement action, which shifts the position of the robot accordingly, or a cleaning action, which removes one task at the current position. Attempted actions may succeed or not, according to predefined probabilities.

We present map configurations and results from experiments, in order of increasing difficulty. Left: two robots and preallocated tasks, middle: four robots and preallocated tasks, right: four robots and randomly appearing (2/3) tasks. The joint reward graphs are presented below respective maps.



References

[1] Aleksander Czechowski and Frans A. Oliehoek. Decentralized MCTS via Learned Teammate Models. Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, pages 81–88, 2020.

[2] Daniel Claes, Philipp Robbel, Frans A. Oliehoek, Karl Tuyls, Daniel Hennes, and Wiebe Van der Hoek. Effective approximations for multi-robot coordination in spatially distributed tasks. Proceedings of the Fourteenth International Conference on Autonomous Agents and Multiagent Systems, pages 881–890, 2015.

[3] Daniel Claes, Frans A. Oliehoek, Hendrik Baier, and Karl Tuyls. Decentralised online planning for multi-robot warehouse commissioning. Proceedings of the Sixteenth International Conference on Autonomous Agents and Multiagent Systems, pages 492–500, 2017.

Acknowledgments

This project received funding from EPSRC First Grant EP/R001227/1, and the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 758824 —INFLUENCE).