MULTI TAXI: A MODULAR SETTING FOR MULTI-AGENT SYSTEMS EXPERIMENTS

Ofir Abu

The School of Computer Science
Engineering, The Hebrew University of Jerusalem

ABSTRACT

Experimenting with multi-agent systems involves two main parts, comparing different multi-agents' *algorithms* in different *environments*. Multi agents' environments such as [1] usually provide the user with closed and well-defined rules of a system, while the ability to change and examine the effects of key ingredients of the system might be of great interest.
In this project, we present `MultiTaxiEnv`[1], a python library inspired by the classical Taxis problem [2] and expand it to the multi-agent setting.
Our library supports 3 main advantages: (1) Experiment with either symbolic information or raw information (represented as images) for agents' observations. (2) Easily configure and deploy costume reward functions. (3) Experiment with multiple types of constraints with varying dependency levels between the different agents such as the agent's fuel capacity or limit to the passengers' amount.
These features allow users to easily modify key ingredients in the multi-agent setting, while still experimenting with agents solving the same main problem.

1. MAIN PROPERTIES OF `MultiTaxiEnv`

The Taxis domain is a well-known problem, first introduced in [2] where it was shown that HQ-learning can successfully achieve satisfying performance in the problem. In this work, we present a python library `MultiTaxiEnv`, based and inspired by the original problem which expands the problem to the multi-agent setting. Our environment has many degrees of freedom, including control of resources, encouraging collaboration through agents' constraints, and controlling the reward function. Here we will describe these degrees of freedom and give examples for *easy to set up* experiments that can be done using our environment:

1. **Control of Resources**: `MultiTaxiEnv` allow the user to define a fuel tank amount (ranging from 1 to $+\infty$) and fuel type (either *fuel* or *gas*) for each individual taxi.
2. **Agents' Intrinsic Constraints**: `MultiTaxiEnv` allow the user to define passengers capacity (ranging from 1 to $+\infty$) for each individual taxi.
3. **Controlling the Reward Function**: `MultiTaxiEnv` allow the user to change and specify custom reward functions, trying to encourage/examine the effect of different functions.
4. **Controlling the Environment Map**: `MultiTaxiEnv` allow the user to create any custom map for the agents to operate in, without limitations on size, amount of obstacles, and so on.
5. **Controlling Agents' Observations' Kind:** `MultiTaxiEnv` allow the user to choose wether the agents will receive symbolic vector as observations or raw image representing the *window of sight* of an agent as observation. A symbolic vector of observations could be for example, a vector of the form [`taxi1-row`, `taxi1-col`,... `passenger1-row`, `passenger1-col`, ... `passenger1dest-row`, `passenger1dest-col`,... `passenger1-status...`]. For example see Figure-1.

Note that we also allow a little degree of freedom in controlling the action space, where a user can choose to allow taxis to `stand-by` or force the taxis to take physical action in every timestep.

2. SUGGESTED EXPERIMENTS IN `MultiTaxiEnv`

In this section, we provide suggestions for *easy to set up* experiments using our framework alongside reference to current research in these topics. We claim that the cost of testing a basic and initial hypothesis in the field is relatively high in terms of setting up and adjusting to complex and strict environments whereas `MultiTaxiEnv`allows for basic and easy to set up experiments with many modifications in the same experiment framework.

1. **Describing conditions for collaboration:** We can easily set up an environment where taxis don't have enough fuel to drive passengers to their destination but

This work is submitted as a lab project under the supervision of prof. Jeffrey S. Rosenschein, Course ID: 67874, and in Supervision and Collaboration with Sarah Keren

[1] https://github.com/ofirAbu/MultiTaxiLib

receive a high reward for getting a passenger closer to its destination or dropping her off, and thus we can experiment with different algorithms and planners to check in which taxis deliver passengers successfully. Similarly, at [3], the authors examine the level of restriction on a bandwidth shared between many devices and its effect on their collaboration.

2. **Sharing resources:** We can look at the passengers as resources that may produce a reward for the taxis. We can then modify the taxis' passengers' capacity, passengers' `dropoff` and `pickup` locations and the reward function such that it is not worthy for a taxi to deliver many passengers all the way, and drop off some of them along their way for another taxi to continue the drive with them, these adjustments can be formulated as a dynamic resource allocation and consumption problem. At [4], the authors experiment with producing policies for resource allocation to applications in the cloud within concurrent demands.

3. **Adversarial Behavior:** We can easily set some taxis with very limited fuel capacity, say group A, and some taxis with unlimited fuel capacity, say group B. In this setting, in each timestep group A will get a reward of -1 with the inevitable end of being stuck until the episode is ended, while group B can try and deliver passengers. The goal of A is thus, to make the episode as short as possible, and the only way to do that is to collide with group B, while the goal of group B is to deliver passengers. The goal of [5] is to produce an algorithm that can convert a coordinated group of RL agents to change its goal.

In each of the suggestions above, `MultiTaxiEnv` allow the user for a "sanity check" of her method that is easy to use and modify, and thus get some fast initial insights for the research to get going in more complex and advanced domains, such as in the references.

3. FUTURE WORK

For future work, we mark 3 main improvements that can advance the library.

1. Allow modifications to the symbolic vector observations, what information is represented, and even what information is represented in which individual taxi's observations.

2. Support the addition of baseline *Taxis Bots* that are controlled by some well-known heuristic or learning method (such as REINFORCE [6] or a DQN [7]) and operate in the environment alongside and independent of the user deployed agents.

We believe that these improvements to the library can reduce even more the work involved in basic and preliminary experiments. The reader is welcome to install and view our library and documentation in detail at our repository[2].

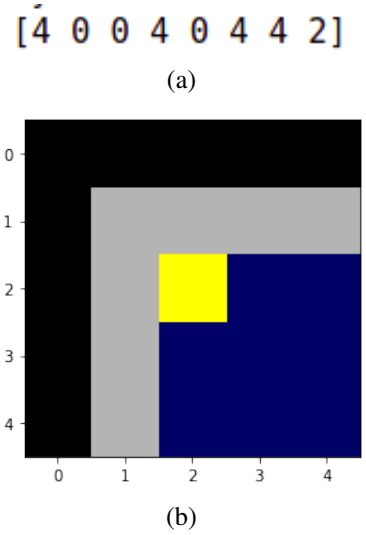

Fig. 1. (a) Symbolic vector of observations, representing the state of a single taxi and the passengers on the map. (b) Image observation showing the window of sight of a yellow agent facing the edges of two white walls in the map.

4. REFERENCES

[1] Justin K. Terry, Benjamin Black, Ananth Hari, Luis Santos, Clemens Dieffendahl, Niall L. Williams, Yashas Lokesh, Caroline Horsch, and Praveen Ravi, "Pettingzoo: Gym for multi-agent reinforcement learning," *CoRR*, vol. abs/2009.14471, 2020.

[2] Thomas G Dietterich, "The MAXQ method for hierarchical reinforcement learning," *ICML*, , no. c, pp. 118–126, 1998.

[3] Motahareh Mobasheri, Yangwoo Kim, and Woongsup Kim, "Toward an adaptive threshold on cooperative bandwidth management based on hierarchical reinforcement learning," *Sensors*, vol. 21, no. 21, 2021.

[4] L. Adhianto, S. Banerjee, M. Fagan, M. Krentel, G. Marin, J. Mellor-Crummey, and N. R. Tallent, "HPCTOOLKIT: Tools for performance analysis of optimized parallel programs," *Concurrency and Computation: Practice and Experience*, vol. 22, no. 6, pp. 685–701, 2010.

[2] https://github.com/ofirAbu/MultiTaxiLib

[5] Martin Figura, Krishna Chaitanya Kosaraju, and Vijay Gupta, "Adversarial attacks in consensus-based multi-agent reinforcement learning," in *2021 American Control Conference (ACC)*, 2021, pp. 3050–3055.

[6] Ronald J. Willia, "Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning," *Machine Learning*, vol. 8, no. 3, pp. 229–256, 1992.

[7] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al., "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.