Bare Demo of IEEEtran.cls for Conferences

Michael Shell
School of Electrical and
Computer Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332–0250
Email: http://www.michaelshell.org/contact.html

Homer Simpson
Twentieth Century Fox
Springfield, USA
Email: homer@thesimpsons.com

James Kirk
and Montgomery Scott
Starfleet Academy
San Francisco, California 96678-2391
Telephone: (800) 555–1212
Fax: (888) 555–1212

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

I. Learning

Learning and planning in this setting is not trivial and can only be achieved through several techniques that allow us to represent the huge state and action space of the *Domination Game* in a minimalistic way. First of all, we are dealing with a continuous state space in respect to the agents' positions, the observations about our opponent agents, the amount of ammo that we have - can be an integer from zero to infinite theoretically, and the general statistics about the game which could be represented by a huge number of discrete integer values. As an example, we can represent the score of the game as a really informative feature of our state space. However, to represent the score you need all 2-decimal point float numbers between 0.00 and 1.00, which are 101 discrete states. In this case, techniques like keeping only the first decimal point of the score value leads to a huge decrease in the number of states, i.e. 11 states. Hence the state space of this game grows rapidly in respect to the number of features we use to represent the state and the action space of our agents. As it is referred in [1], the biggest difficulty when we are dealing with state-based problem formulations, is the *curse of dimensionality*, the state action space grows exponentially with the number of features. We can see this *Domination Game* problem as a *Markov Decision Process*, which is fully observable in respect to our agents' positions, the control points' state, and the game state. However, we have partial observability for the opponent agents, which of course can be represented as full observable case only when opponents are in our field of view.

A. Grid World

Our first approach to deal with the problem of the continuous state space was to split the map into tiles of size 16×16, which was also the original tile size of the game itself. A method which is similar with *tile coding*, as it is described in Chapter 7 of *Hado Van Hasselt* book, about *Continuous State and Action Spaces*. So, the first feature of our state space is the position of our agent in this grid world, meaning the closest from our agent tile coordinate in the 2D space, given by the *Euclidean distance*. For example, the states of the agent position can be given by,

$$S_{position} = \{(0,0), \ldots, (World_{width}/16, World_{height}/16)\} \tag{1}$$

The second feature of the state space is the *Control Points State*, each control point can be the following states,

$$S_{cps} = \{-1, 0, 1\} \tag{2}$$

-1, when is dominated by the opponent team, 0, when is neutral, and 1 when is occupied y our team. Given that we have two control points in our map the state that is able to describe the *Cps* state space is the following,

$$S_{full\_cps} = S_{cps} \times S_{cps} \tag{3}$$

, which is the cross product between the two control points' states. For the action space, each agent had all neighboring tiles to drive there or to stay in the same position, resulting to 9 actions for each given tile. However, actions which would lead the agent hit a wall in the map excluded from selection.

The reward function in this approach was obtained by a single function, which was checking the difference between the previous and the current condition of the *Cps* state.

$$Reward = (S^{t}_{full\_cps(i)} - S^{t-1}_{full\_cps(i)}) \times 10 \tag{4}$$

$$Reward = Reward + (Ammo^{t} - Ammo^{t-1}) \times 4 \tag{5}$$

There was also a reward when the agent succeed to obtain ammo, but weighted less that the reward for capturing a control point.

1) Independent Q-Learning: We decided, to implement *independent Q-Learning* for each one of our agents. Each agent holds its own *Q-table*. The states that an agent can be are the possible position in the grid world, together with the state of the control points. Furthermore, in each of these states, it has a set of possible action that can lead the agent to drive in a neighboring tile. The update rule of the *Q-Learning*, was updating the table of each independent agent every time the agent reached its target location, i.e. its previous action. We used *Q-Learning* with 0.7 learning rate-α and the same value for discount factor-γ.

2) Results: The results of this approach was tested in two settings. The first setting was on the same map but on a *1vs1* game. The second approach, was the original game *3vs3*. The results of *independent Q-Learning* in this grid world representation of the problem were really bad and we think that are not even worth to be mentioned here. One explanation is that we have no information in the state space referring to the agent's orientation. So, each agent can easily choose target locations that are located in its backward direction and this caused the agents to perform really slow turns and causing a delay to the completion of their actions, as well as the update rule for several game steps.

B. A Better Approach

Third component of our state space is the opponents' positions. However, our agents are not always aware of the opponents' positions in the map, due to the limited range they can observe. As a result, we decided to represent our knowledge about the opponents as a vector of four elements, as many as the interesting points in the map. Each element in this vector

References

[1] Craig Boutilier, Thomas Dean, and Steve Hanks. "Decision-theoretic planning: Structural assumptions and computational leverage". In: *arXiv preprint arXiv:1105.5460* (2011).