Scenario Formalization in RL

1 The problem of Vehicle Platooning

We're considering the simple case of two vehicles, one leader and one follower. They both run on one line, x-axis and the goal is to minimize the distance between them without crashing. We work in an episodic-manner, starting from a trivial case, moving to a more complex environment where an adversary can exist.

1. No adversary. Basically, we're dealing with the simple goal of minimizing distance and not crashing in a deterministic environment.
2. Adversary input w/ false communication.
3. ??

In the following, we explain further how the state looks like in the deterministic environment.

2 MDP in a Deterministic Environment

We consider the environment as a markov-decision problem, such that, the model is represented as follows, $< S, A, R, \phi, \gamma >$.

2.1 S – state vector

The state vector is as follows $[t, x_1, x_2, v_1, v_2]$, s.t. t is the current timestep that increases by one everytime an event/action is done. The positions are represented on the x-axis and we consider both vehicles (1) the leader and (2) the follower. The same is given for the velocities. Precisely, to make it easier in the translation between the RL and the SC, we consider a joint state representation for the two vehicles, rather than consider every vehicle as an autonomous agent. It would be interesting to consider the case where both are. Bunch of question marks here.

2.2 A – action vector

The action vector is a vector of all possible actions that can happen at a given time. Since we are dealing with a joint system, we want the actions to control both the agents at a given time, thus every element of the vector is a pair. For simplicity, we start with defining accelerations as either AL, AM, C, DM, DL. AL and AM define accelerating a lot and a accelerating minimally respectively. C is to stay constant. DL and DM is to deaccelarate a lot vs a little respectively. Thus we have 25 possible actions at a given time. This implementation of accelerations makes sure that at any given time, there will always be the same number of actions available, since these definitions are kind of a blackbox. The SC then acts as the behind-the-scenes solver to decide on actual numbers of these high-level decisions. It would be interesting to consider whether this would look like with actual numbers in the actions vector.

2.3 R – reward function

The reward function $R(s_t) = X - s_t[x_2] + s_t[x_1]$, where X is the leeway of the distance. Consider, $X = 2m$ where $x_1 = 0$ and $x_2 = 1m$. Thus, the AI is rewarded 1 at that given point. For a larger distance, say $x_2 = 3m$ then the AI is punished since the reward is then -1. Precisely, X defines what we would like the maximum distance to be. The less the distance, the more the AI is rewarded, thus states where the distance is more minimal are better and should be exploited.

2.4 ϕ – transition function

Todo.

2.5 γ – discount factor

Todo.