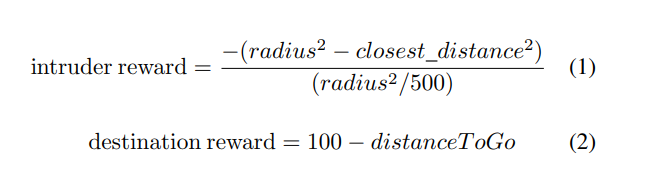
Implementation:

Reward Function:

The Reward Function implemented in the paper is



But the authors are clear how they actually integrated these 2 rewards into the SARSA Algorithm. Since, SARSA only takes one reward in the update step. Due to this ambiguity we used a reward function

Our Method:

Total Reward = intruder reward + destination reward

Total Reward range = [-500,100]

If closest\_distance > 50 => intruder reward = 0

This meant our reward takes both the reward into account. And motivates the agent to get maximum reward i.e., try to avoid collisions but also avoid in such a way which maximizes the destination reward.

SARSA:

In the paper see, SARSA Implementation

The points 5 and 7 contradict each other or atleast ambiguous on how their SARSA implemention works.

According to point 5, they use a saved Q table to generate the next timestep action.

According to point 7, they only train the SARSA model on the planes which are within the colliding radius.

Therefore their SARSA Q table is only good for generating actions for the planes which are in colliding distance. But, what about the planes which are not colliding, how are their actions determined? .

Our Method:

To solve this problem we actually experimented with 2 methods

Method 1(default):

This is the default method in our project. Here the SARSA policy generates all the actions of all the planes without regard to if its colliding or not.

There is actually a problem with this method, since the Q-table is only a 4D matrix, which has info of closest\_distance,angles... Therefore, it has no information about the destination direction or even its distance from the agent. Therefore it can’t perfectly represent all the states possible by our agent. But its performance is still good. And somehow performed better than our method 2.

Method 2 :

We use a greedy + SARSA approach. Here we take a greedy straight line path towards the destination for the planes which are non-colliding. For the planes which are colliding, we use SARSA to determine its action.

Therefore our greedy+SARSA approach only learns / updates the Q-values on the planes which are in colliding distance.

Scoring Function:

Since, we didn’t have an exact information on what the reward function was, that was given to the SARSA model. We also had to wonder what the ‘Scoring Function’ they used. There was not a single mention on how they calculated the average score. So, we just used

Average Scoring func = Cumulative sum over episodes (average of 25 planes per episode)

\*\* So we cannot compare our results directly to the ones in the paper but still we can do a relative comparison between the different methods we implemented

Extra Achievements:

The authors suggested to implemented their SARSA with different and many destinations( more than 1). And we were successfully able to implement that with our environment thanks to our custom environment code, which we wrote from the ground-up. In our environment we can select how many destinations we would like to have. It can range from 1, where all the planes land at same destination ( same as implemented in the paper), to number of planes, where each plane lands at a different destination.