**TEM**

**Project**: Task Assignment in a Stochastic Environment Using Hierarchical Reinforcement Learning

**Scholar**: Colin Acker

**Mentor**: Scott Nivison

**Program**: Doolittle Scholar

**School**: Georgia Institute of Technology

***Abstract***

A task assignment problem for heterogeneous agents within a stochastic and uncertain environment is studied in the paper. This work considers a base defense scenario where the goal is to minimize the number of assets destroyed by a missile, while effectively using the least number of interceptors. The scenario is developed with known deficiencies within the sensor model (which provides information to identify threat type) and assumptions for a priori knowledge of unusable regions of the state-space (e.g., high attrition zones) and the path planning algorithm used by the interceptor. The main contribution of this paper is the hierarchical reinforcement learning framework which accurately predicts incoming threat types and utilizes this information to estimate the most effective time to release interceptors, which types of interceptors to send, and where to send interceptors. This work also combines a “lever” neural network, (LNN) with RL-based techniques to better account for large state and action spaces, which are environments where traditional RL-based techniques (e.g., deep Q networks) perform poorly. More particularly, this “lever” neural network is shown to be useful when trying to estimate the near-optimal time step to release interceptors. Unlike many task allocation schemes, the algorithm presented in the paper offloads the online computation to an offline learning procedure, which results in a fast and efficient task assignment scheme.

***Introduction***

In the last few years, the task allocation problem has been combined with multi-agent systems, especially multi-UAV systems. In the problem formulated in this paper, the task planner must make decisions and choose the right interceptor given environment uncertainty. There have been many different techniques used to solve this problem. Many people have used genetic algorithms to solve this task allocation problem [1, 2]. There have been particle swarm optimization algorithms [3]. There solutions are not without faults. Firstly, using the genetic algorithms and particle swarm optimization algorithms has been proven to be quite computationally expensive, and they do poorly with stochastic environments. More recently, reinforcement learning algorithms (DQN) have been used to solve this problem quicker, and with a stochastic environment [4]. This solution does do well, however, the stochastic elements in the environment were too simple, and any more complex uncertainty would render DQN useless.

This paper presents a solution using RL, but with a more complex environment. We solve the task allocation problem using hierarchical reinforcement learning. The goal of this research is to be able to solve complex stochastic environments while being both computationally quick and accurate. Consider the Israeli Iron Dome, the interceptors must make quick and accurate decision based on its environment, in order to save its citizens from being killed by the missile.

***Experiment***

Consider a base defense scenario, where the goal is to minimize the number of assets destroyed by an incoming missile. We can do this by sending out interceptors, to intercept these incoming missiles. We used 3 different neural networks, one which chooses what type of interceptor to send out, one which chooses which silo to release the interceptor, and one which chooses when to release each interceptor.

In our environment, we are assuming there are 1-6 incoming missiles at any given time. Each missile has an X and Y coordinate, a type, and a frequency that has been sampled. Note that the missile type is not known to the task planner, only the frequency sampled. Frequency sampling works as such:

(1)

(2)

where is the distance and is the type of missile. The frequency is then sampled from a normal distribution with this standard deviation and mean. We are assuming each missile has some mean frequency as such:

|  |  |
| --- | --- |
| Mean Frequency | |
| Missile type 1 | 2000 |
| Missile type 2 | 4000 |
| Missile type 3 | 6000 |

**Table 1. Mean frequency for each missile type**

For each timestep, a frequency is sampled, and the missile comes one step closer. As the distance decreases, so does the standard deviation, so the sample becomes more closer to the mean. The “radar” uses the frequency sampled to estimate probabilities and what it thinks each type could be. Estimated probabilities are defined as such:

(3)

We are using hierarchical reinforcement learning (HRL) to solve this task assignment problem. In HRL, the main goal is divided into a hierarchy of subproblems. This reduces the computational complexity due to the smaller state and action spaces. The hierarchy includes the main task assignment problem at the top, with choosing when to release interceptors below that, and choosing what types and where to release interceptors under that.

When choosing when, we use a novel LNN. A LNN is defined as a neural network that has a binary output, (one of which is a waiting action and the other is a lever action), and when the waiting action is selected, it updates the input until the lever action is selected. The waiting action updates the input until the lever action is pulled. Every time the waiting action is chosen, the missiles come 1 step closer. We want the LNN to predict the optimal timestep to intercept. If it is too far, we are quite uncertain on the type of missile, if it is too close, the interceptor will not have time to intercept.

The networks that choose which types and where to launch each interceptor are much simpler. The choose which types network takes into account what the radar says and makes a prediction on which interceptor to send out. The choose when network takes into account the missiles target and places the interceptor at the optimal silo.

***Data/Results***

We turned a simple, deterministic task assignment environment into a complex, stochastic environment. I plan on comparing the DQN [4] approach to my HRL approach, but I did not have enough time this summer. However, the fact that HRL preformed so well on this stochastic environment is a good sign.

I created the LNN because a normal NN combined with the large action space needed to predict the timestep wasn’t preforming well. The LNN shrinks the action space from 100 to 2, and it preforms much better.

|  |  |  |  |
| --- | --- | --- | --- |
|  | LNN NN | | |
| Average reward convergence | | ~ -220 | ~ -888 |
| Average timestep convergence | | ~ 62 | ~ 44 |

**Table 2. Comparison between LNN and NN**

When this becomes a paper, there will be much more data and results, however, most of my time was used creating the HRL framework and this complex environment. Therefore, I do not have many results.

***Discussion/Conclusion***

I believe the LNN has many sorts of applications. It effectively shrinks the action space so RL algorithms such as PPO work much better. It is obviously most effective when choosing when to launch something, as there is a waiting action and a launching action. I plan to apply the LNN on environments that use a NN to predict a timestep.

The choose where to launch interceptors network assumes the missile’s target. Obviously, in a real-life scenario, we are not sure of the asset the missile is going after. I plan to make this network more realistic by predicting the asset it’s attacking, given previous data, and using that prediction to choose the silo to launch from. Also, the choose which types network is quite simple, so I should find a way to make it more complex.

Turning this environment into a dynamic battleship game would be interesting as well. We can assume there are 2 players and treat it as a game theory problem. Both players would learn information about how each other plays and develop strategies accordingly.

Lastly, it would be interesting to give each missile a velocity, and we can use guidance laws for both missiles and interceptors. Right now, the missile locations are discrete, but if we turn them into continuous coordinates, we could apply more realistic dynamics to these missiles.