Enhancing Balance Robustness in Swarm-based Multi-Agent Reinforcement Learning Under Random Freeze Events

Riley Roberts
COMP4900-G

Contents

1 Introduction

1.1 Impact

In real world applications, individual robots or drones may occasionally lose control due to hardware malfunctions. By understanding how to design MARL systems that are robust to intermittent control losses it allows us to deploy reliable, real world multi agent systems. The insights gained from this project may inform better design of fault-tolerant algorithms in various domains such as autonomous vehicles and drone swarms.

2 Methods

2.1 Environment

Base Scenario: Building upon the balance VMAS scenario where multiple agents must work together to a bring a package balanced on a line towards a goal without it dropping.

Modified Dynamics: At random intervals a randomly chosen agent will freeze for k steps. During these steps:

- The frozen agent still receives its standard sensory inputs
- It computes an action using its policy
- Instead of executing the computed action the environment overrides it with a default "zero-acceleration" action

Agent Sharing: All agents share the same network parameters and operate with decentralized policies. No explicit communication is allowed between agents.

2.2 Agent Observations

Local State: Each agent observes position, velocity, relative position to the package, relative position to the line, relative position between package and goal, package velocity, line velocity, line angular velocity, and line rotation mod pi.
(Modification) Swarm Summarization Statistics: In addition to the sensory inputs already observed by the agent, they will additionally receive a mean feature vector and

variance of chosen features (likely position and velocity). The goal of this is to provide a sense of how the swarms collective is behaving to help detect anomalies.

2.3 Agents Goal

Balance Scenario: Keep a structure balanced despite occasional loss of control in agents.

2.4 Baseline

For the baseline algorithm I have chosen IPPO as it has been shown to perform above heuristic in previous papers and is readily available via MARLib.

2.5 Validation Strategy

The following validation strategies will be employed: comparison against the baseline, episode reward mean and average episode length. These metrics will allow us to verify how our model performs against a control revealing the impact of the aggregated statistics.

2.6 Implementation

The file `sim_train_modified.py` contains three sections. The first is `CustomVmasEnv()`. This class alters the VMAS environment to enable freezing.
The second is `MyScenario()` which is an altered version of the BALANCE scenario. The first alteration made to the scenario can be found in the observation function. As previously mentioned aggregated swarm statistics (average/variance of position and velocity) are calculated and included in the observations recieved. The second main change is in the reward function. Inside the reward function is where we select our agent to freeze and decrement the counter once one is frozen, this is done as the reward function is called during each step. This is then assisted by the final alteration to the scenario which is the addition of a function `process_action()` which carries out the role of forcing the frozen agent to maintain zero acceleration.
The final section of the file is dedicated towards training the model through the BenchMARL library. In the `task.config` definition, we can dictate the number of agents and k, the variable dictating how long agents are frozen for. Additionally in this section the IPPO algorithm is imported for use.

3 Results

3.1 Training

Each model was trained for just over 100 episodes, after which performance plateaued across models. The graphs shown below reflect training with 4 agents and k set to 35 steps. It should also be noted the the chance for an agent to become frozen when none others are is set to 1 in 10.

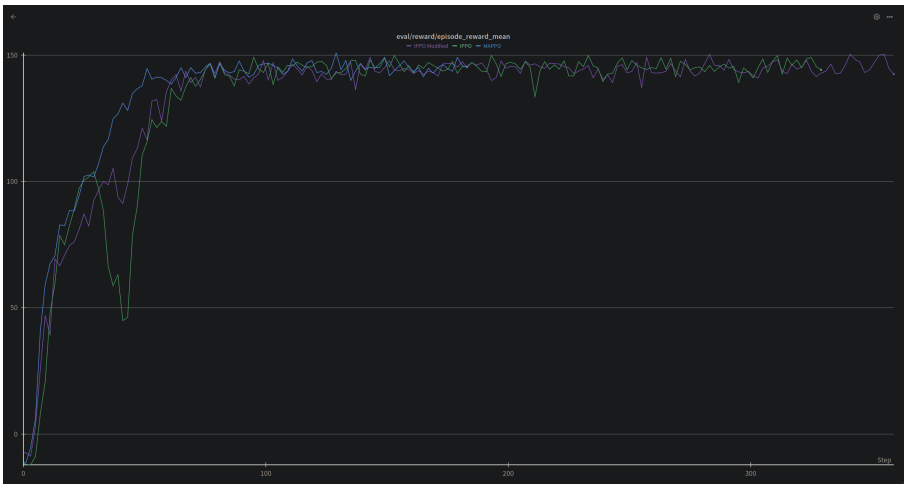

Figure 1: Average Reward Per Episode

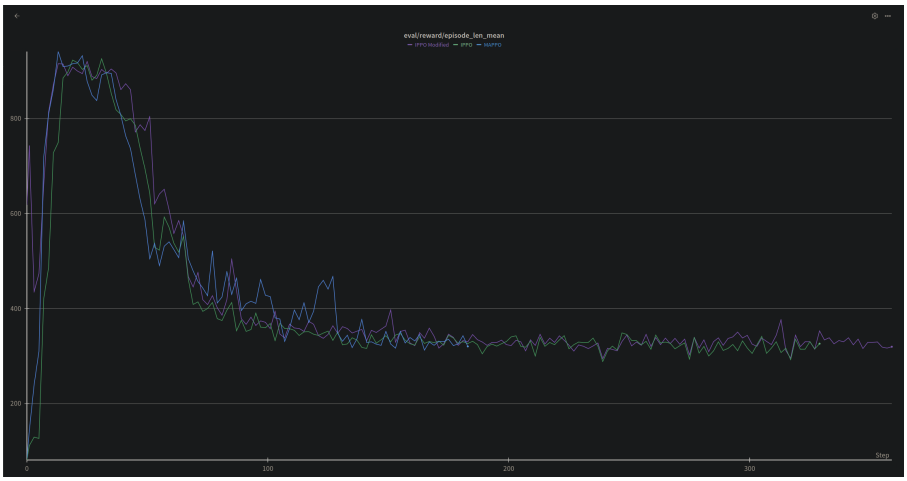

Figure 2: Average Episode Length

3.2 Analysis

As can be seen in Figure 1, IPPO without swarm statistics struggles during the first 75 episodes. The IPPO supplied with swarm statistics does not suffer this same loss however. Could it be more effective with a smaller episode budget? More testing should be done in the future. It can also be seen that MAPPO, there for an additional baseline to compare with, outperforms both under budget constraints. All 3 experiments yielded the same height plateau for the average award and just under 150.

As for the average episode length, all experiments performed roughly equally with swarm statistics and MAPPO faster at learning how to keep the ball alive (the initial spike of episode length).

4 Discussion

4.1 Limitations

While some information has been gleaned through the course of the project, many limitations were ran into that should be addressed. Time was the main limiting factor in the development and experimentation phase of this project and with more of it, further clarification on the uncovered questions may have been answered. Additionally, the swarm statistics approach has shown to be largely ineffective in most scenarios and a different plan of attack may be a good way forward.

4.2 Future Work

As previously mentioned, expansive training and experimenting should be conducted to get a wider grasp on the extent of the minor differences observed. How do these differences hold up when the number of agents change? How do these differences hold up when agents are frozen for longer? What about shorter periods of time? Leaving these aside, we can also dig into how these additional observations would affect the MAPPO algorithm instead. Or we can leave the aggregation statistics behind and explore different avenues for finding freeze tolerant systems.