Digital Improvement of Propeller Inspection Throughput

Manasa Akella, Undergraduate at Georgia Institute of Technology

Investigators: Manasa Akella (GT Undergraduate)

November 1 - December 31, 2020

Project Objectives

The main objective of this project is to create an optimized scheduling algorithm(s) to increase blade throughput for the Warner Robins AFB blade inspection process. We formulated our approach with the aim to study and optimize a real world manufacturing process - a penetrant dye quality control inspection process in the defense industry. The inspection is centered around a rigidly programmed robotic arm, which moves components through stations according to various precedence and time based rules. There is significant loss of productivity time and throughput due to frequent changes in the process requirements, leading to a cycle of constant reengineering and reprogramming of the robot arm. The technical goals for this quarter included engaging in conversation with a contractor to develop a front end app engine for the DIPIT software to allow for easy usage by the Warner Robins floor engineers and continue testing of the software to test it’s functionality and optimize the algorithm as well as obtain data necessary for a publication.

Summary of Major Technical Accomplishments

- Develop method for integration of RL algorithm with an App Engine
- Determine hyperparameter settings for optimal learning and reward gain
- Track and measure behavior of RL algorithm over different parameters

Technical Discussion

Background

Manufacturing processes often require agility to adapt to ever changing demand and requirements. This presents a challenge if these processes must also adhere to strict time and spatial constraints. With the need for agile process control under manufacturing uncertainties, reinforcement learning can be a viable approach for implementing intelligent manufacturing systems. In this paper we present a method for implementing reinforcement learning frameworks on manufacturing systems with time based and precedence constraints in order to optimize throughput in a real world quality control process. We test our approach on a simulation of a system in question as well as a small scale physical version of the manufacturing process to validate our results. The reinforcement learning problem typically consists of an agent and an environment, where at each time step, the environment is in a well-defined state, and the agent selects an action from a finite set of possible actions. The agent receives a reward for the action chosen in this state and the chosen action causes the environment to change to a new state at the next time state. The objective of the problem is to maximize the long term expected reward by determining the optimal policy for choosing decisions based on the states experienced. If the desirable outcomes of the system are easily identifiable, but the system is too complex to directly determine an optimal solution, we can employ reinforcement learning to learn an optimal decision-making policy for the process.

Methods, Assumptions and Procedures

The inspection process itself is very straightforward, it is centered around a Fanuc R-2000iC robot. The blades are dipped into the penetrant dip, the emulsifier dip, and the final developer, in that order, and then baked. The process gets complicated because of the many hard timing and operational rules. For example the penetrant dip cannot touch the emulsifier dip and vice versa. The current method allows for a 3 blade throughput per day which is 9 blades less than the eventual goal of a 12 blade daily throughput.

The current program is rigid, as most robot programming is, but the input stream of blades is largely unpredictable. Therefore a maintainer has to choose between stopping the robot to reset for a new batch or waiting until the current batch is finished. This is what I mean when describing blades arriving in a batched fashion or a random fashion. When the blades arrive in a batched fashion the blades are all already in the queue when the process starts and in a random process the blades arrive whenever they arrive.

A type of RL known as Q-learning is being used for this project. This is used when all of the different states are not known.

In the past few weeks the focus has been on the reward signal and maximizing the reward gained by the algorithm. The reward signal defines the goal. On each time step, the environment sends a single number called the reward to the reinforcement learning agent. The agent’s objective is to maximize the total reward that it receives over the long run. The reward signal is used to alter the policy. So in the case of the DIPIT algorithm the agent is given a positive 10 reward every time it dips the blades before their hour is up and 100 reward for every blade exits the cycle. The agent is penalized with negative reward when it takes the wrong action or an action that leads to a terminal state(invalid action) or breaks one of the timing or operational rules.

In order to increase the reward earned during a training episode, the algorithm needs to learn more efficiently. One of the first ways to approach that is by tuning the hyperparameters. The initial tests included changing the exploration rate(one of the hyperparameters that controls the rate at which the algorithm learns and the ratio of exploring to exploiting) from a constant value to an exponential decay. The rewards grew immensely as can be seen in fig. 1

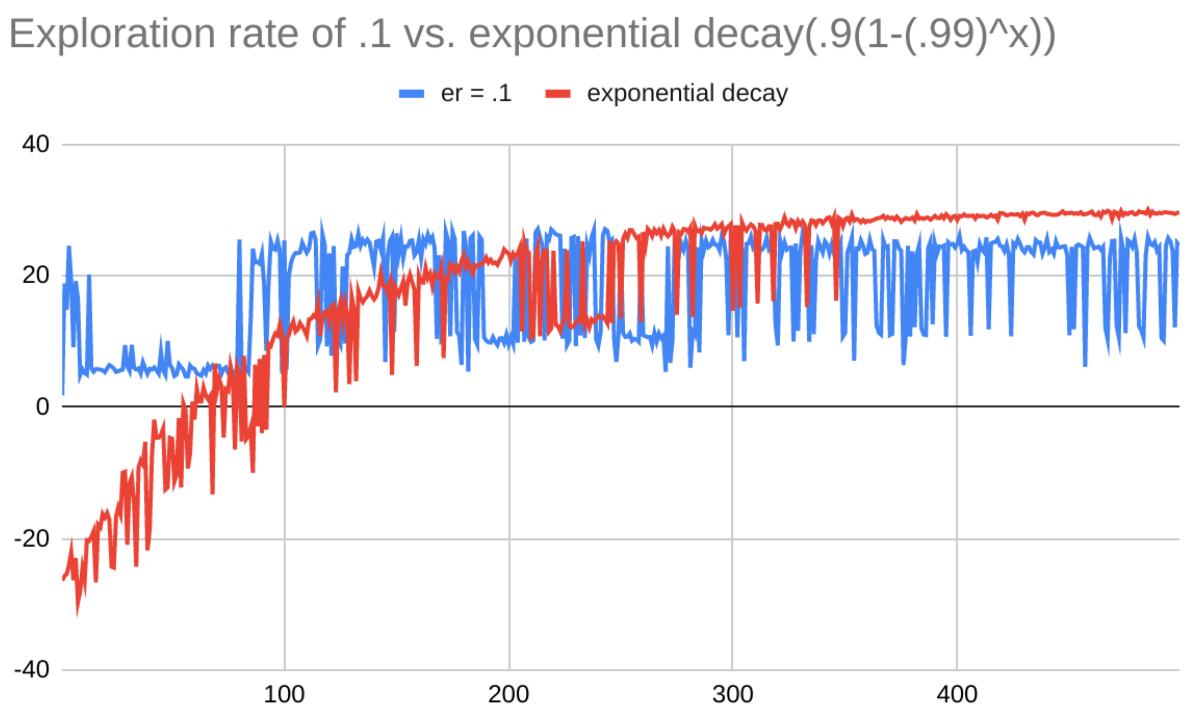

fig.1

Results and Discussion

As seen in the figure above an exponential decay function to represent the exploration rate had a much smoother learning curve and leveled off at a higher reward than a constant exploration rate. Because of this it is clear that much more tests need to be done to determine the most ideal hyperparameter set up for this algorithm. Thus next steps should include testing multiple exponential decay functions to determine the best for training. The data collected from this will also back the reasoning behind the hyperparameter structure in the paper.

The following data should be collected as the next best step:

1) 5 Exponential exploration rate(s) to create graph representing x(training episodes)/y(reward) with various exploration approaches:
   Constant rates: 0.1, 0.2, 0.3 (500 episodes) w/ 20 blades at start
   Variable Rate: a bx (500 episodes)
      a = 0.7, b = 0.99
      a = 0.8, b =0.99

a = 0.9, b = 0.99
a = 0.7, b = 0.95
a = 0.8, b =0.95
a = 0.9, b = 0.95
a = 0.7, b = 0.9
a = 0.8, b =0.9
a = 0.9, b = 0.9

2) Training method vs. % of successful policy tests out of 50 trials (1-50 blades). Conclusion from the figure is what exploration rate worked best for this scenario
3) Batch arrival w/ variable blades (trained on n number of blades for each episode) and 500 episodes with ‘best’ exponential rate from 1 & 2

   Study: Randomly selecting blades (upper bound changing)

   1 - 4
   1 - 8
   1 - 12
   1 - 16
   1 - 20

Conclusions and Recommendations

- Hyperparameter tuning greatly affected the policy functionality and will be looked into further for optimization of the RL algorithm
- Basic timeline has been detailed for App Engine project

Anticipated Technical Tasks for Next Quarter

- Further testing for Hyperparameter Tuning
- Further analysis of algorithm and policy functionality given different initial circumstances