DIP-IT:
Digital Improvement of Propeller Inspection Throughput

Manasa Akella

Undergraduate Intern
Mechanical Engineering
Georgia Institute of Technology

Rafael Piloto

Undergraduate Interns
Computer Science
Emory University

Other Team Members: Connor Guerin (GaTech), Prof. Ai-Ping Hu (GaTech), Andrew Gillman (CIV), Dan Berrigan (CIV)

Funding Sources: RT Core Funding (Transition: Robins AFB Investment)

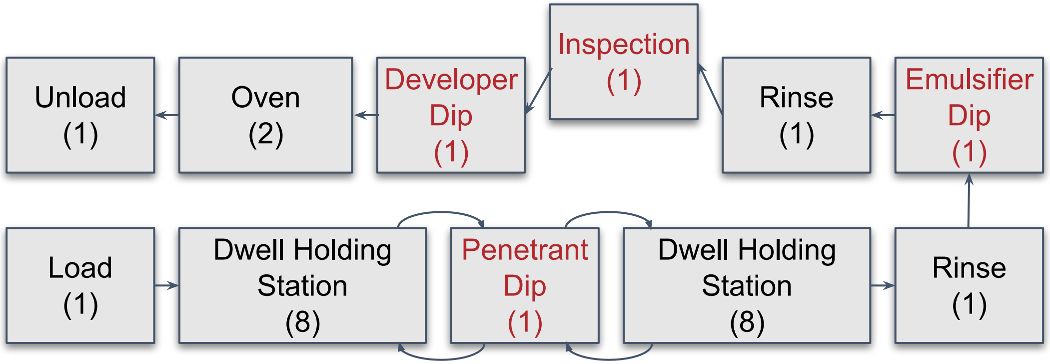
Fluorescent Dye Penetration Process

Robins AFB

FPI Robot

TODAY

FPI Process Flow



A single robot arm (above) applies fluorescent dye to C-130 blades for inspection at a rate of **3 blades per day (avg.)**.

CHALLENGES

- A pre-programmed methodology ***cannot*** adapt to new inputs on-the-fly.
- **High non-recurring engineering cost to adapt** to changes in process requirements.
- **No process model** to determine future investment ROI (e.g., add robot, dwell station)

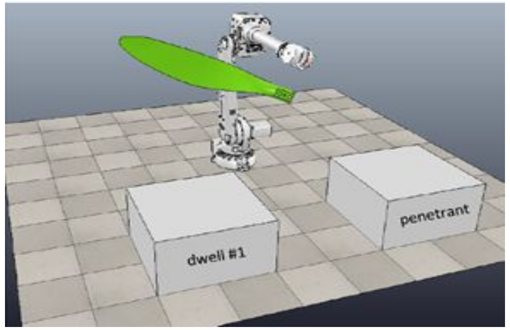
GOALS

- **Provide agility and model-based guidance** on future process improvement
- Enhance C-130 propeller blade throughput **to >8 blades in single day**
- Create FPI processing data stream for **linking process to outcome**

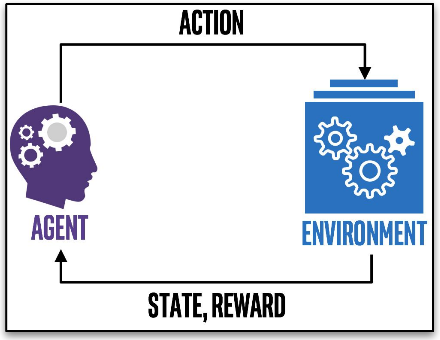
Where We Were - June 2020

Algorithm Development

Built digital twin of process



Developed RL Approach (Q Learning)



Visualization model for validation

Model-based integration of process timing

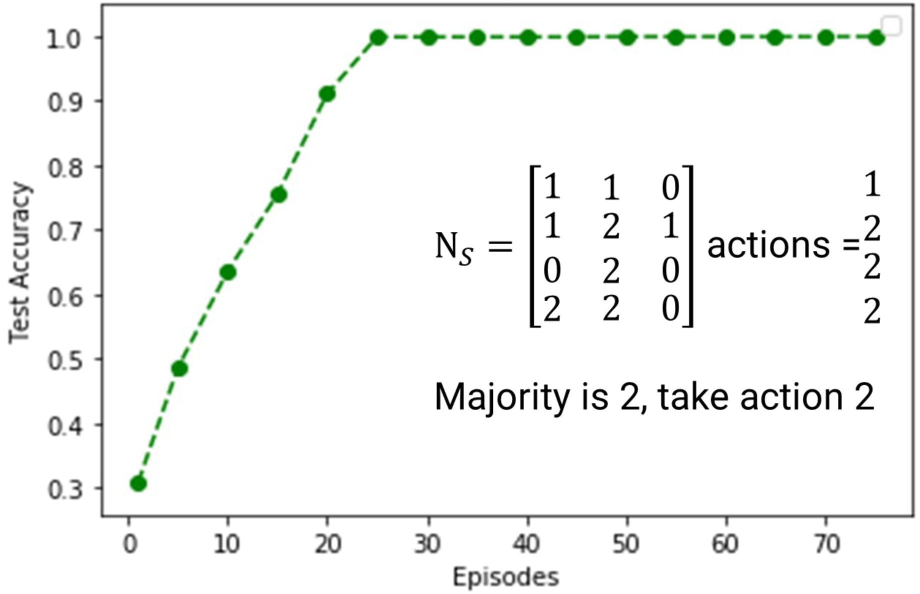
- General design for integration of process req's
- Tailored reward structure for timings violated
- Enables tractable dimensionality of state space

Definition of state vector:

$$S_i = [\underbrace{s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8, s_9, s_{10}}_{\text{Station information}}, \underbrace{s_{11}, s_{12}, s_{13}}_{\text{Timing Indicators}}]$$

0-empty station
1-blades in station
2-station full

State estimation strategy to improve learning rate



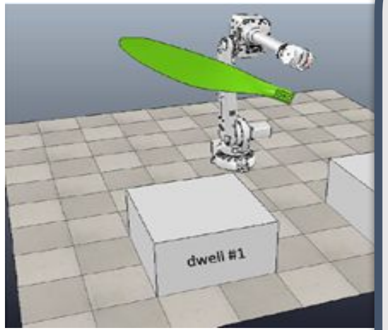
Where We Were - June 2020

Algorithm Development

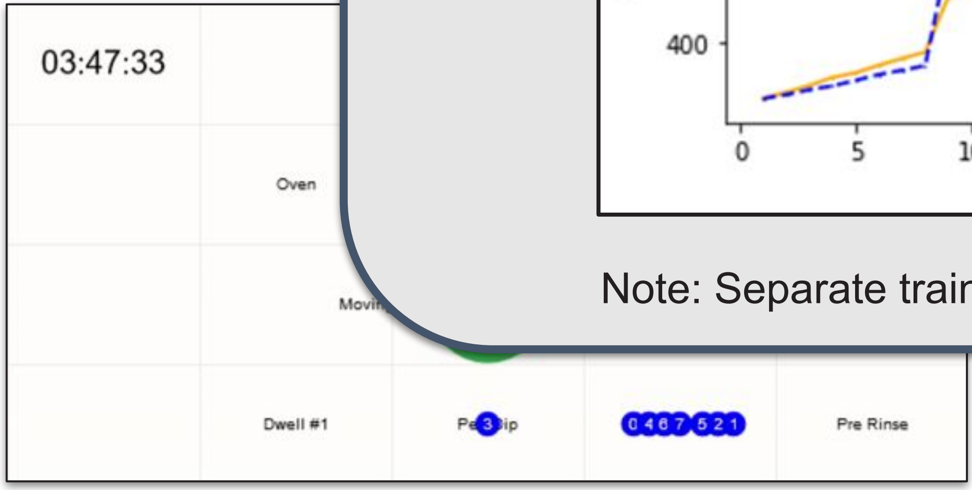
Model-based integration of process timing

- General design for integration of process req's
- timings violated
- lity of state space

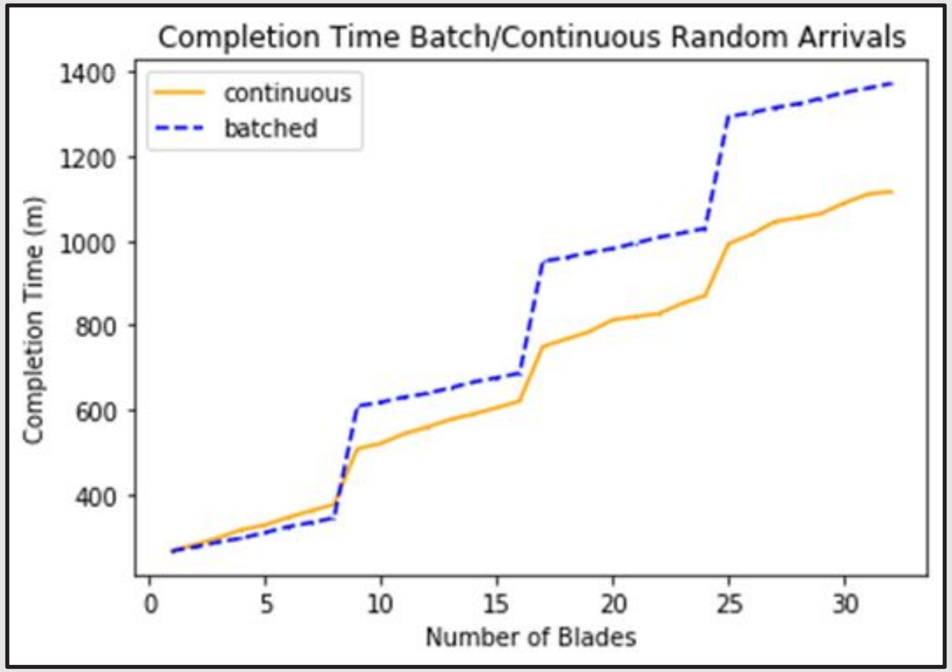
Built digital twin of proc



Visualization model f



Resulting Improvement

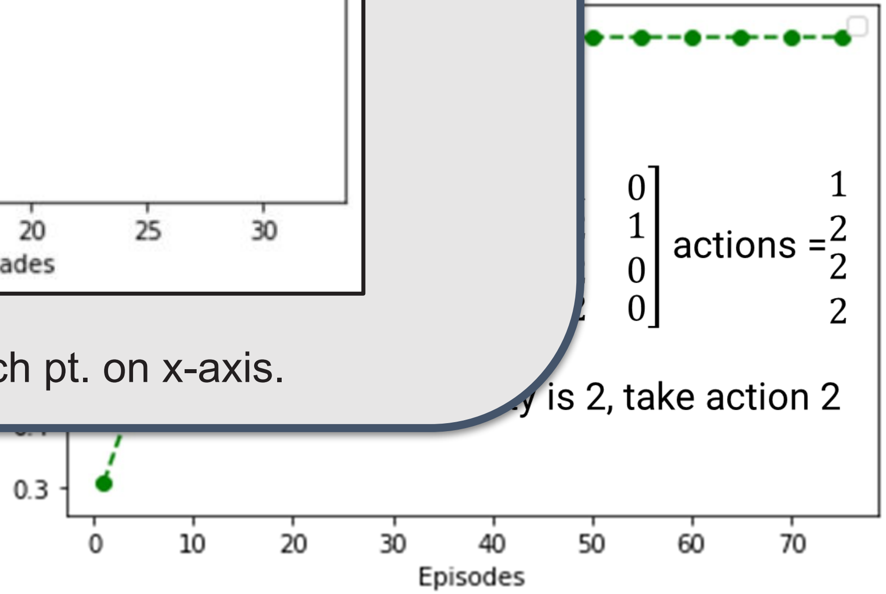


Note: Separate training for each pt. on x-axis.

$, s_{10}, s_{11}, s_{12}, s_{13}]$

n   Timing Indicators

earning rate



0 1 0 0   actions = 1 2 2 2

y is 2, take action 2

Where We Were - June 2020

Successful Demonstration

Trained RL algorithm with policy

Action number to perform from policy element corresponding to S

S = [s1, s2,...,s13]

Convert to Usable input for Robot Controller

Convert to state array

Command for performing robot action

Robot Controller

Sensor Readings

Tracked blade times

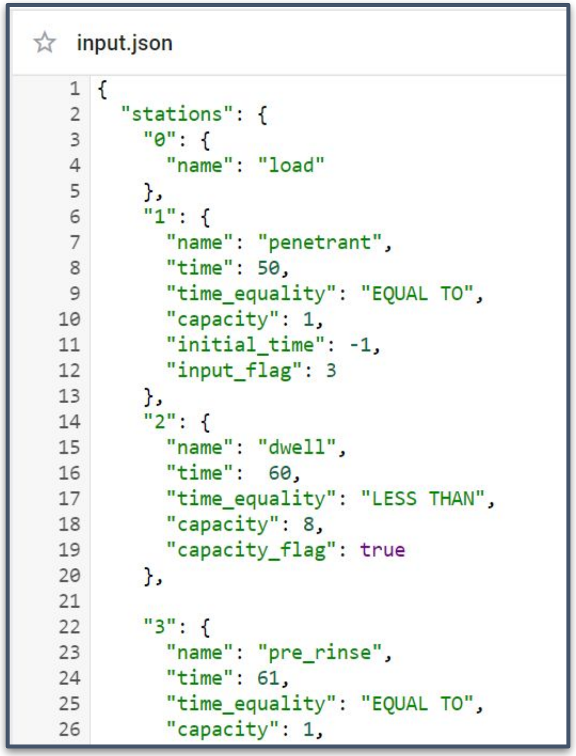
Execute action and read inputs

Summary of Key Updates

Main Goals: Generalizing Policy & Abstracting Code

Code Generalization & Clean-up (Rafael & Manasa)

- Pulled out constants and hyperparameters with first pass at generalizing input process
  - Single input file (json)
  - Updated Documentation
- Link to Google Cloud Storage



input.json

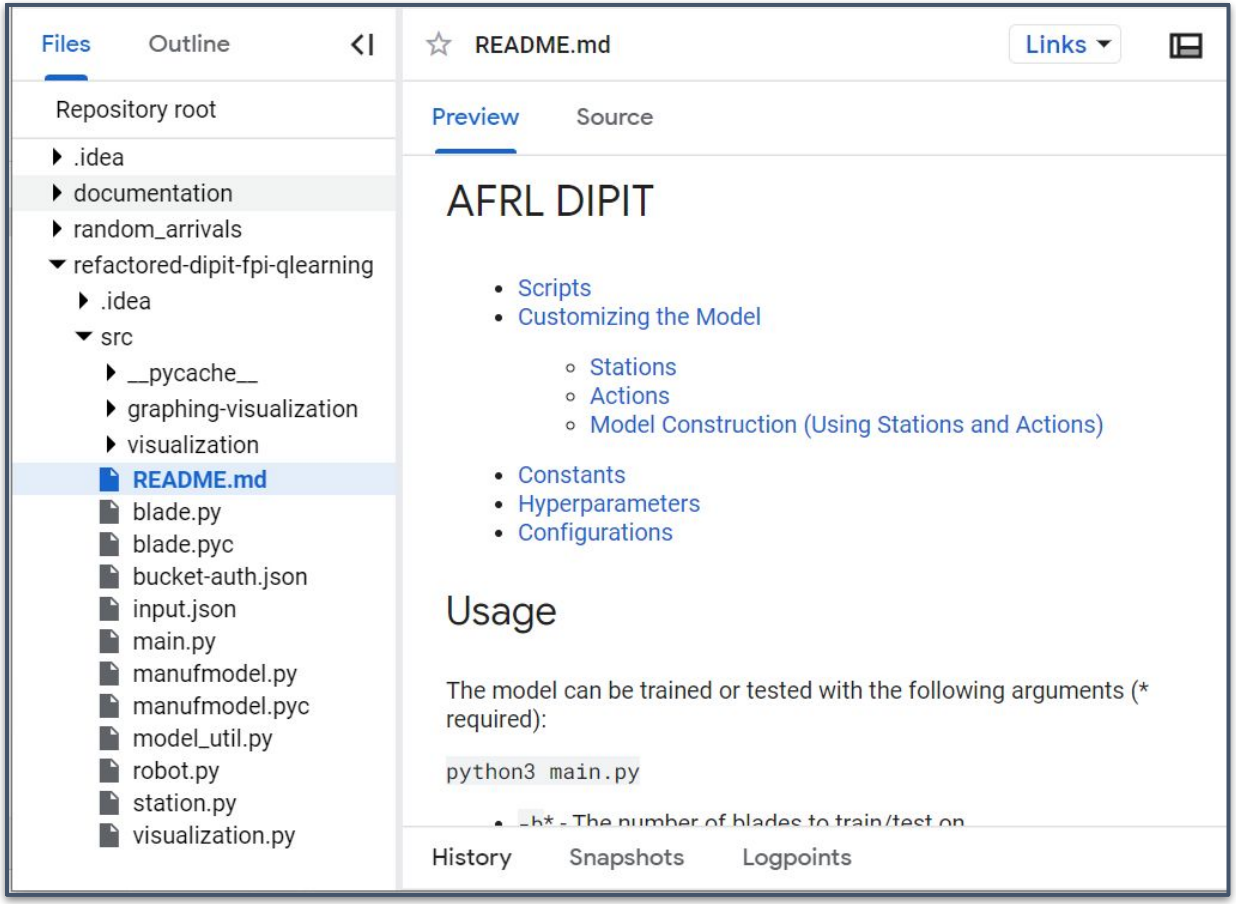
```
{
  "stations": {
    "0": {
      "name": "load"
    },
    "1": {
      "name": "penetrant",
      "time": 50,
      "time_equality": "EQUAL TO",
      "capacity": 1,
      "initial_time": -1,
      "input_flag": 3
    },
    "2": {
      "name": "dwell",
      "time":  60,
      "time_equality": "LESS THAN",
      "capacity": 8,
      "capacity_flag": true
    },

    "3": {
      "name": "pre_rinse",
      "time": 61,
      "time_equality": "EQUAL TO",
      "capacity": 1,
```

input.json

```
  "constants": {
    "MAX_DWELL_TIME" : 3600,
    "MIN_COOK_TIME" : 900,
    "REQUIRED_DIP_TIME" : 14400,
    "PASS_RATE" : 0.95,
    "MOVEMENT_TIME" : 10,
    "BUFFER" : 240,
    "FLAG_COUNT" : 2,
    "RANDOM_SEED": 0
  },
  "hyperparameters": {
    "discount": 0.2,
    "learning_rate": 0.2,
    "exploration_rate": 0.1,
    "decay_factor": 0
  },
  "configuration": {
    "blades": 9,
    "random_arrivals": false,
    "rate": 20,
    "is_test": false,
    "episodes": 2,
    "estimate-unvisited": false,

    "save_to_bucket": true,
    "bucket_name": "rafael-test-bucket",
    "storage_client_auth_path": "./bucket-auth.json"
  }
```

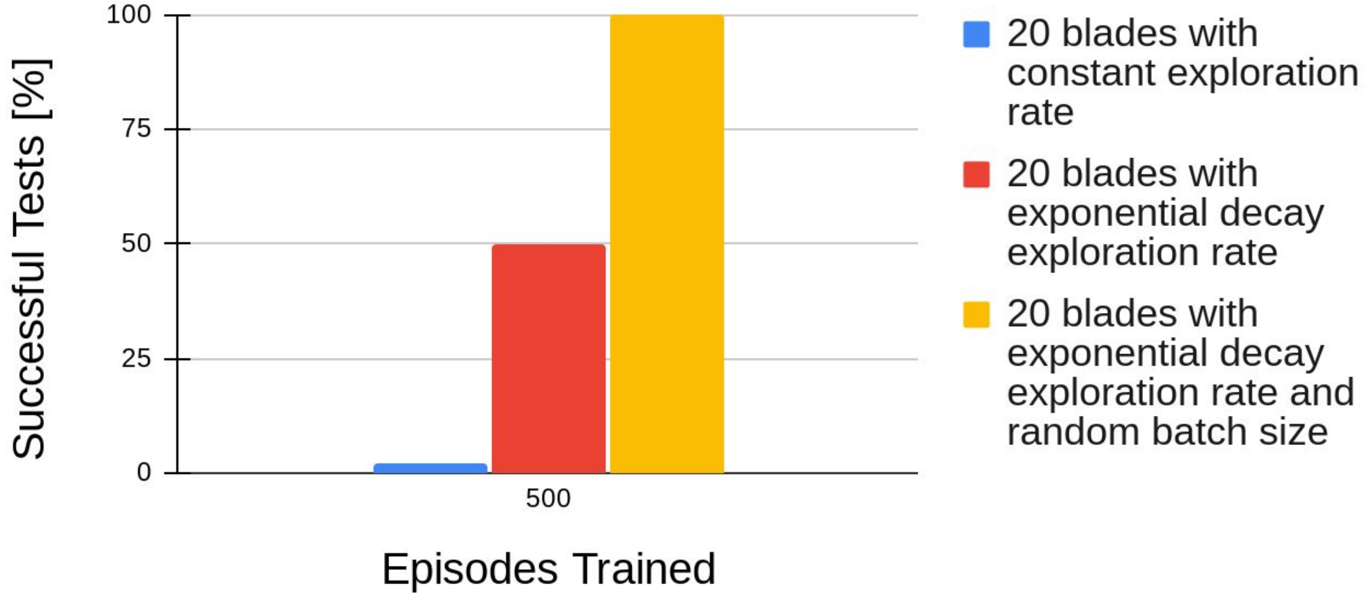


Summary of Key Updates

Main Goals: Generalizing Policy & Abstracting Code

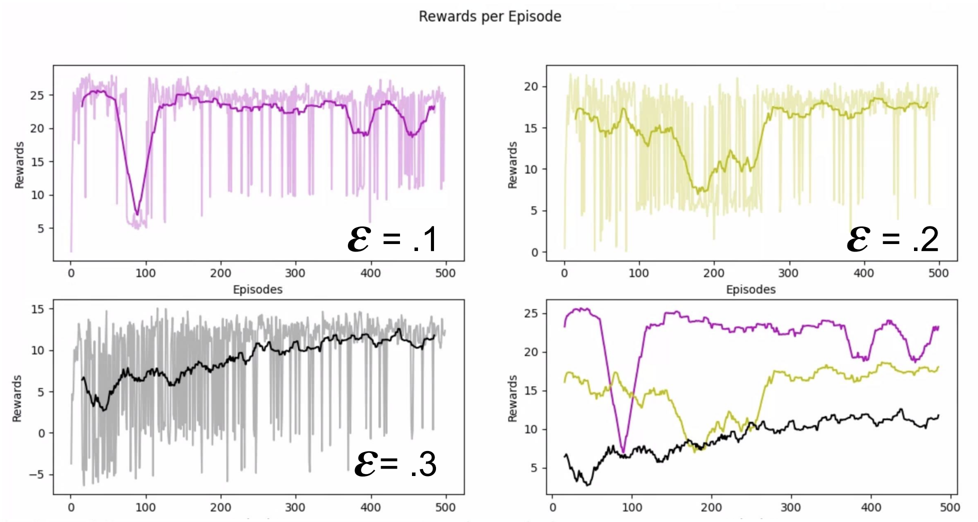
Returning a General Policy (Manasa)

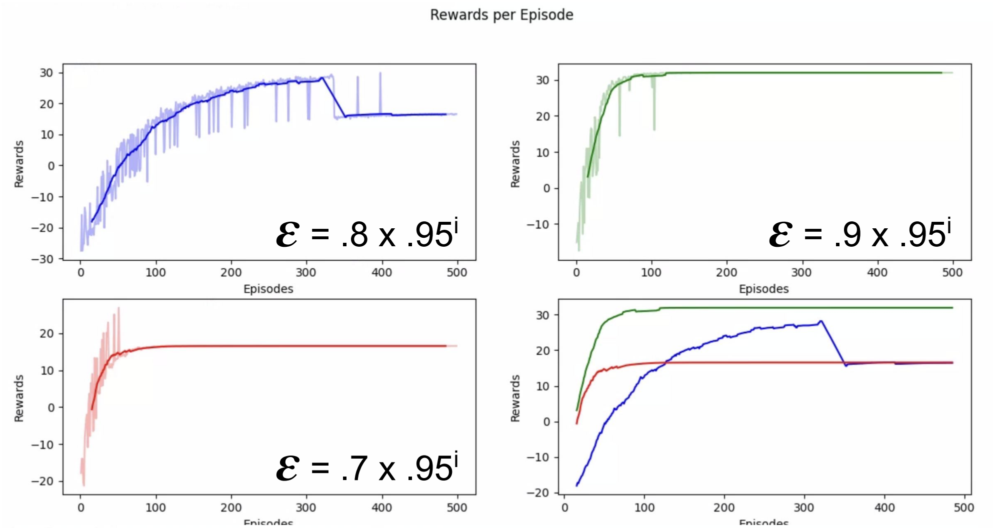
- Moved from *single* number of blades considered in each episode during training to *range*
  - Signs of improving agility of policy
- Explored constant and variable exploration rates
- Generalized to random arrivals as well but tabled this for now until we understand agile batch arrival
- Remaining Question: How would traditional infinite supply scheduling optimization formulation compare?

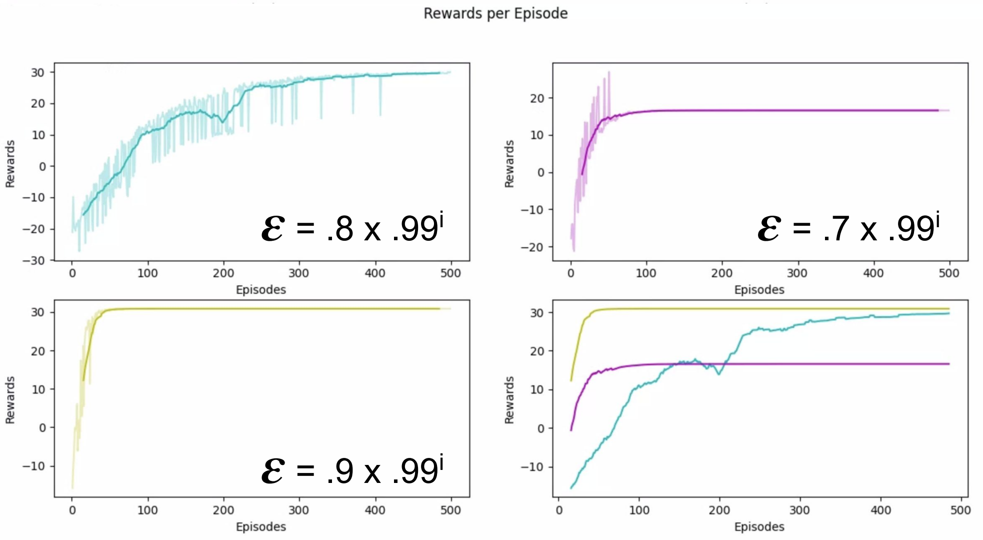


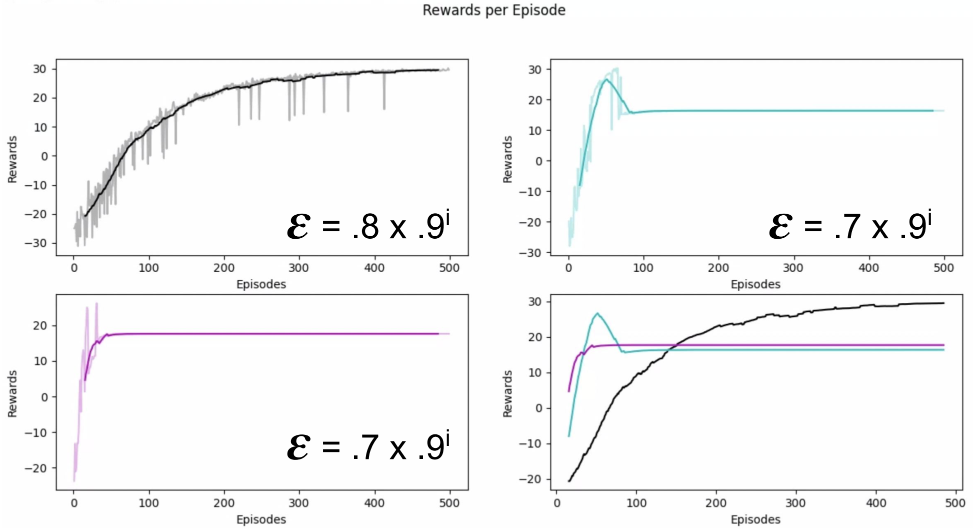
Recent Results

Systematic study of *exploitation v. exploration* and *sources of uncertainty* on RL training



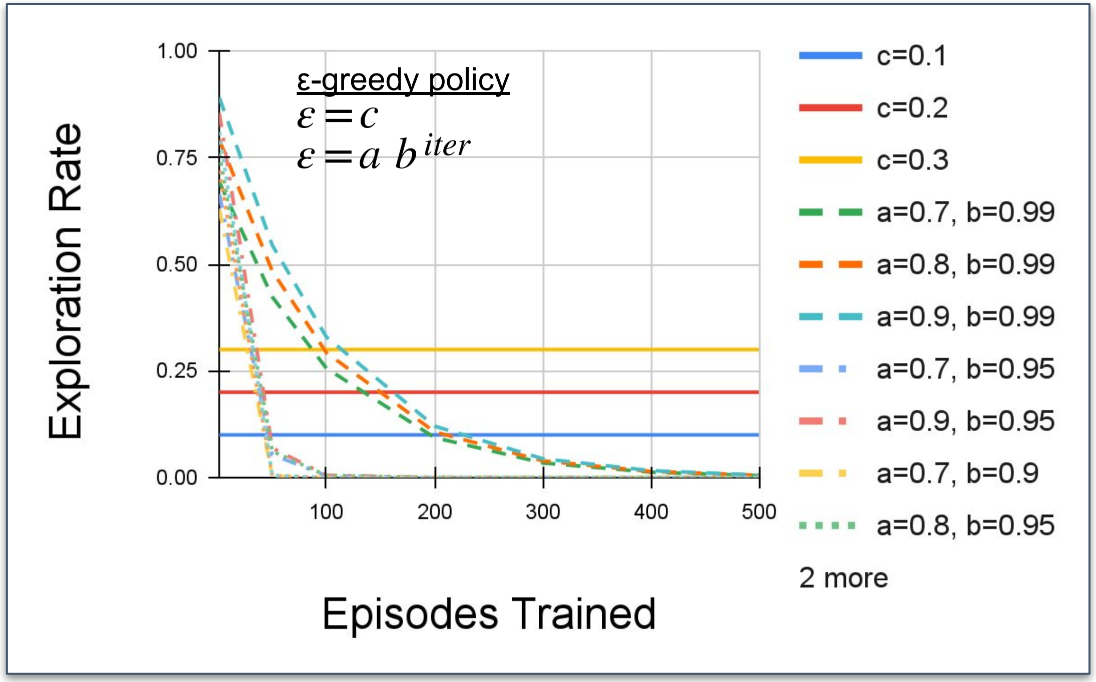


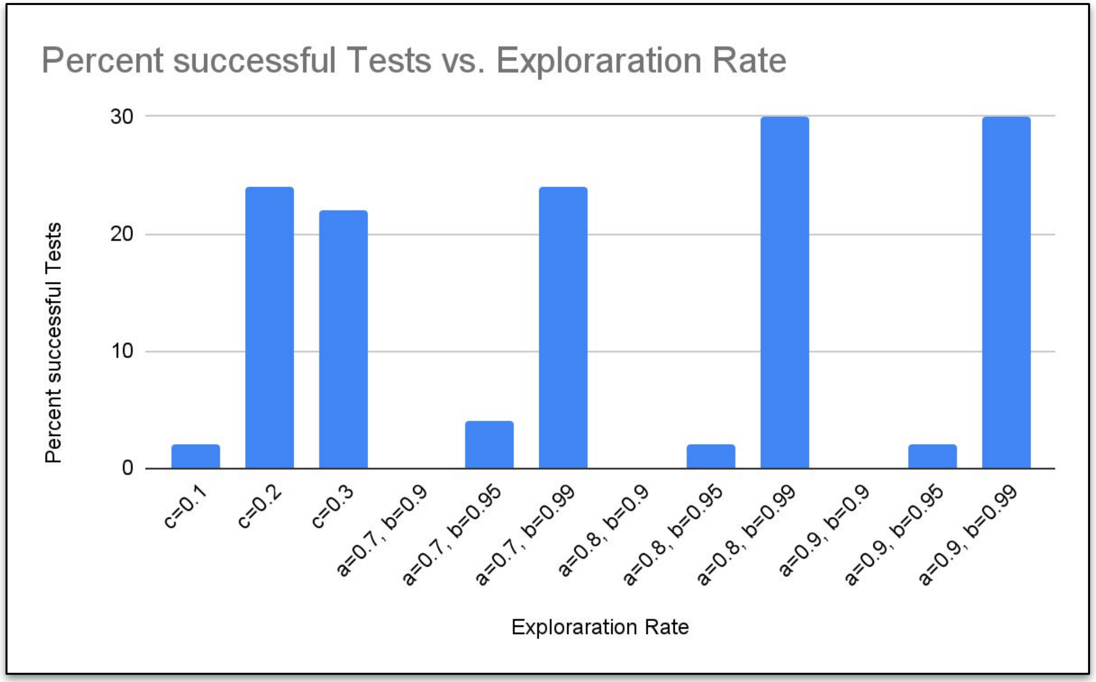




Recent Results

Systematic study of *exploitation v. exploration*
and *sources of uncertainty* on RL training
Note: Need to study plot on right a little further given how we define a successful run



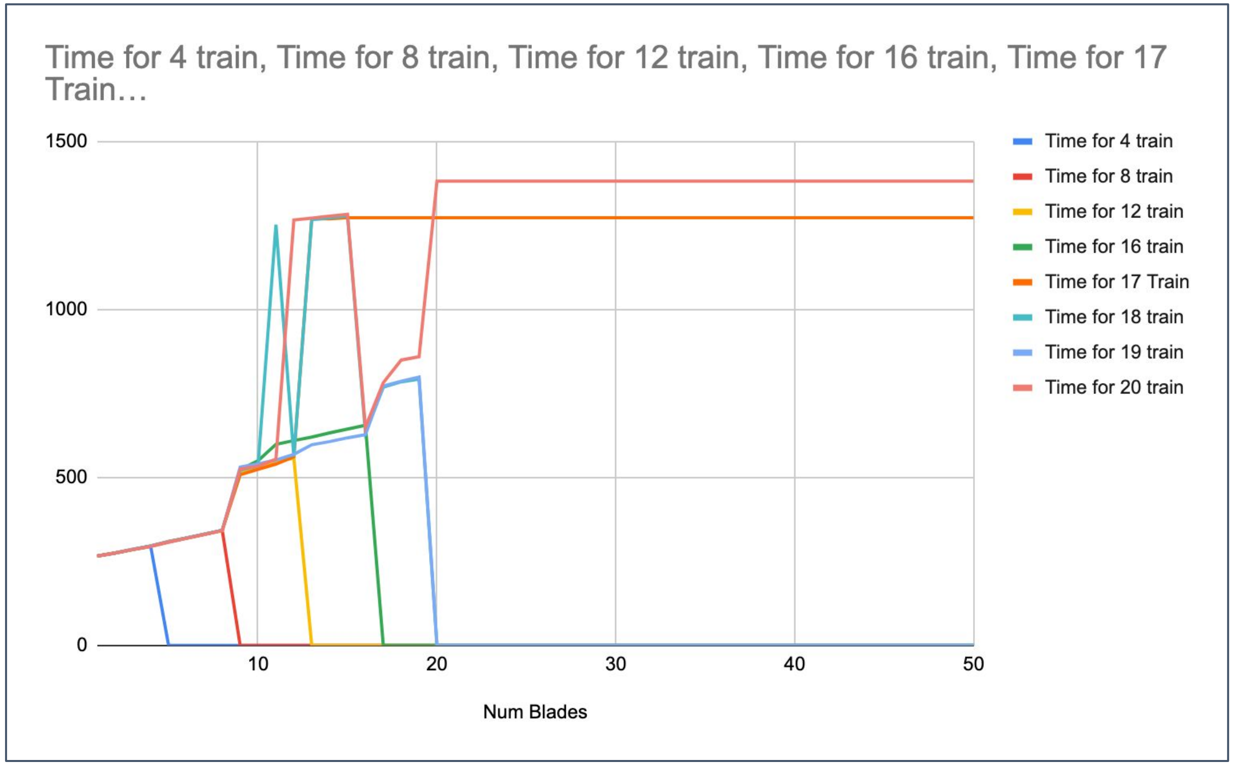


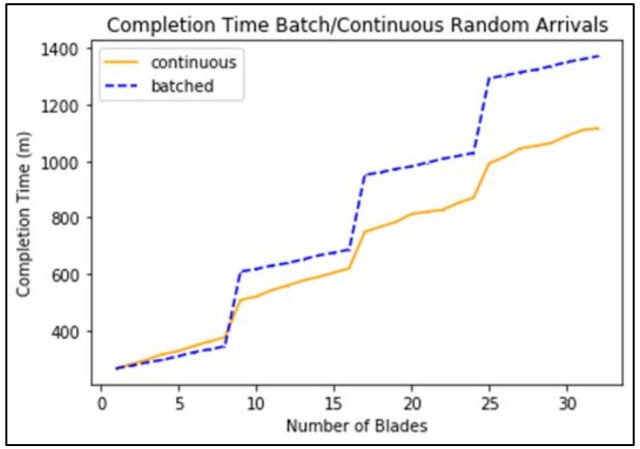
Recent Results

Agile policy for up to 16 blades achieved

Need for studying more carefully not just cycling all blades through or ending an episode without a negative reward

Num Blades	Time for 4 train	Time for 8 train	Time for 12 train	Time for 16 train	Time for 17 Trair	Time for 18 train	Time for 19 train	Time for 20 train
1	265.95	265.95	265.95	265.95	265.95	265.95	265.95	265.95
2	275.25	275.25	275.25	275.25	275.25	275.25	275.25	275.25
3	285.5333333	285.5333333	285.5333333	286.5166667	285.5333333	285.5333333	286.5166667	285.5333
4	294.8333333	295.8166667	295.8166667	295.8166667	295.8166667	295.8166667	295.8166667	294.8333333
5	0	309.4166667	309.4166667	308.25	308.25	308.25	308.25	307.2666667
6	0	319.7	319.7	319.7	319.7	319.7	319.7	318.7166667
7	0	331.15	331.15	331.15	331.15	331.15	331.15	330.1666667
8	0	342.6	342.6	342.6	342.6	342.6	342.6	341.6166667
9	0	0	517.1	523.1833333	508.25	522.0166667	531.05	524.9666667
10	0	0	532.9666667	550	524.4333333	534.0166667	540.35	533.65
11	0	0	547.1833333	598.6333333	539.8166667	1252.416667	551.8	554.3166667
12	0	0	557.4666667	609.9	561.7166667	562.0166667	568.9	1266.266667
13	0	0	0	620.1833333	1270.083333	1267.25	597.5	1271.933333
14	0	0	0	632.6166667	1270.266667	1272.916667	606.8	1277.616667
15	0	0	0	644.0666667	1272.916667	1278.6	618.0666667	1283.3
16	0	0	0	655.5166667	1272.916667	634.25	627.3666667	649.6166667
17	0	0	0	0	1272.916667	768.6166667	772.3666667	781.8333333
18	0	0	0	0	1272.916667	784.6166667	786.5833333	849.5166667
19	0	0	0	0	1272.916667	792.65	799.0166667	859.3666667
20	0	0	0	0	1272.916667	0	0	1382.133333
21	0	0	0	0	1272.916667	0	0	1382.133333
22	0	0	0	0	1272.916667	0	0	1382.133333
23	0	0	0	0	1272.916667	0	0	1382.133333
24	0	0	0	0	1272.916667	0	0	1382.133333
25	0	0	0	0	1272.916667	0	0	1382.133333
26	0	0	0	0	1272.916667	0	0	1382.133333
27	0	0	0	0	1272.916667	0	0	1382.133333
28	0	0	0	0	1272.916667	0	0	1382.133333
29	0	0	0	0	1272.916667	0	0	1382.133333
30	0	0	0	0	1272.916667	0	0	1382.133333





Future Goals

Key Generalization Problems:

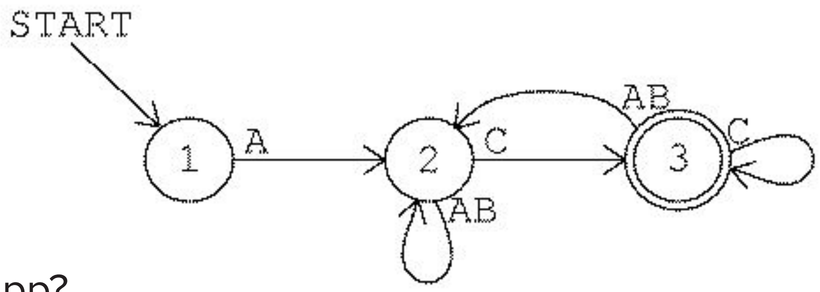
- Fundamental problem within the decision making process leading to the majority of tests being loops of positive or negative reward
- Revisit random arrivals

Potential Solutions:

- Retrain and retest the collected data using new method of determining a successful test
- Analyze reward structure
- Analyze timing rules and how they are implemented

Scalability & Abstraction

- Abstract model creation for policy training and testing
- Visually create dynamic models based on a user created directed-graph and specified rules (Automata)



User Interface Design

- Appetite for cloud interface? Web app?
- Determining relevant visual tools for user interaction