AUTONOMOUS AIRCRAFT SEQUENCING AND SEPARATION USING HIERARCHICAL DEEP REINFORCEMENT LEARNING

Marc Brittain

Advisor: Peng Wei
Intelligent Aerospace Systems Lab
Iowa State University

June 29, 2018
OUTLINE

• Motivation

• Background

• Problem Formulation

• Solution Approach

• Results

• Summary & Future Work
RESEARCH MOTIVATION

• Recent proposals for low-altitude airspace operations
  • UAS Traffic Management (UTM)
    • For remote-piloted or unmanned autonomous drone operations
  • Urban Air Mobility (UAM)
    • For vertical takeoff and landing (VTOL) personal air travel or urban air taxi operation
RESEARCH MOTIVATION

• According to [1, 2, 3], an autonomous air traffic control system for automated sequencing and separation is expected to:
  • Increase airspace capacity
  • Enhance operation safety
RESEARCH MOTIVATION

• Can we design a framework to enable an artificial intelligence (AI) agent to perform sequencing and separation tasks?
PREVIOUS WORK

• Original proposal of autonomous air traffic control system was from Heinz Erzberger and his colleagues at NASA [1, 2, 3]

• Believed to be the ultimate solution to accommodate dense, complex, and dynamic air traffic

• Developed a core algorithm called the Autoresolver that detects and resolves conflicts between aircraft
PREVIOUS WORK

- Autoresolver
  - Iterative Approach - sequentially computes and evaluates candidate trajectories
  - Output is a trajectory that satisfies all resolution conditions
  - Physics based approach that involves separate components of conflict detection and conflict resolution
  - Does not integrate conflict detection and conflict resolution
PROBLEM FORMULATION

• We propose a hierarchical deep reinforcement learning framework to enable autonomous air traffic sequencing and separation
REINFORCEMENT LEARNING

• Involves an agent who takes sequential actions within an environment to maximize a cumulative reward function

• Example: Pac-man

\[
\begin{align*}
R(s) &= 1 \text{ (grab pellet)} \\
R(s) &= 2 \text{ (eat ghost)} \\
R(s) &= -5 \text{ (collision)}
\end{align*}
\]
REINFORCEMENT LEARNING

- Artificial Intelligence (AI) agent AlphaGo built by DeepMind defeated the world champion Lee Sedol in Go (March 2016) [4]
REINFORCEMENT LEARNING

• AI agent developed by OpenAI is able to defeat the world’s top professionals in DOTA 2 [5]

• Requires players to develop intuitions about their opponents and plan accordingly
PROBLEM FORMULATION

• Formulate the sequencing and separation as a reinforcement learning problem and solve using hierarchical deep reinforcement learning

• Deep reinforcement learning excels in environments where there is a large state space

• Uses a neural network to determine the optimal action given a particular state
PROBLEM FORMULATION

• Assumptions

  • Finite number of aircraft
  • Deterministic environment
  • Agent can only select the action to change route during the first time-step
  • 2-D environment
SIMULATOR

- NASA Sector 33
- Air traffic simulator developed by NASA
- Objective is to have all aircraft arrive at goal position before time runs out
- Many different problems ranging in difficulty from 2 aircraft to 5 aircraft
- Difficulty of a problem is based on number of aircraft and number of routes each aircraft can take
6 speed changes

Goal positions

Route change
REINFORCEMENT LEARNING FORMULATION

- State
  - Parent Agent - Image of the game screen
  - Child Agent - Position & Speed of aircraft

- Action
  - Parent Agent: During the first time-step \((t = 0)\), the parent agent determines which aircraft should change route
    \[
    A_P = [C_1, C_2, C_3, C_4]
    \]
    where \(C\) is a given route combination for all aircraft and \(n\) is the number of unique aircraft route combinations.
  - Child Agent - During each time-step where \(t > 0\), the agent determines the speed for all aircraft
    \[
    A_C = (600,600), (540,600), \ldots, (300,300)
    \]
REINFORCEMENT LEARNING FORMULATION

- Reward

- Parent Agent

\[ r_P = \frac{1}{\sum_{i=1}^{N} |g_{x_i} - x_i|} \]

- Child Agent

\[ r_C = 0.001 \sum_{i=1}^{N} v_i - 0.6 \]

\( g_{x_i} \) is the location of the goal position for aircraft i
REINFORCEMENT LEARNING FORMULATION

• Episode
  • One entire play through the environment
  • Concludes when terminal state is reached

• Terminal states
  • +10 all aircraft arrived at goal position
  • -5 aircraft did not make it to goal position
  • -10 aircraft collided
EXAMPLE OF REINFORCEMENT LEARNING STATE

• Assume image has \( m \times m \) pixels

• Parent state:
  \[
  S_P = (p_1, p_2, p_3, \ldots, p_{m \times m}),
  \]
  where \( p_k \) represents the intensity tuple of pixel \( k \) (R, G, B)

• Child state:
  \[
  S_C = (x_1, y_1, x_2, y_2, v_1, v_2, C_j)
  \]
OVERALL SYSTEM ARCHITECTURE
REINFORCEMENT LEARNING FORMULATION

• State transition:
  
  • Parent agent’s and Child agent’s state transition at different temporal levels
  
  • For a given child agent state \( s \), the next state, \( s' \) will be determined based on how the image was updated after the action was selected
  
  • Allowed 4 in-game seconds between selecting action and determining next state
  
  • Aircraft update position based on an unknown dynamics model built in the simulator
WHY DO WE NEED HIERARCHICAL REINFORCEMENT LEARNING?
HIERARCHICAL REINFORCEMENT LEARNING

• Allow for non-Markovian environments to be expressed as a hierarchy of agents

• These agents individually satisfy Markovian property:

\[ (s_t | s_{0:t-1}) = (s_t | s_{t-1}) \]

• Hierarchical agents can operate at different temporal domains
HIERARCHICAL REINFORCEMENT LEARNING

- Each aircraft is able to change route until the “20” mark
- The action to change route is time-sensitive
- Action to change speed of aircraft is not
HIERARCHICAL REINFORCEMENT LEARNING

- Given two action sets (change route and change speed), assign each set to an agent in hierarchy

\[
A_P = [C_1, C_2, C_3, C_4] \quad A_C = [(600,600), \ldots (300,300)]
\]
HIERARCHICAL REINFORCEMENT LEARNING

• Example of hierarchical reinforcement learning episode progression

1. Initialize state, $s_p$

2. Parent agent selects action to change route

3. Parent agent propagates this information to child agent by state alteration

$$S_C = [S_p, a],$$

where $a$ is the action the parent agent selected

4. Until terminal, child agent chooses an action to change speed at each time step
HIERARCHICAL REINFORCEMENT LEARNING

\[ S_C = [S_P, a] \]

- By adding the action to the state of the child agent, there are many computational benefits

- Avoid training many additional neural networks

- Provides for a more robust model by having the ability to learn different policies based on parent action
CASE STUDY

• In this work, we examine a two aircraft problem (2-3), where each aircraft can take two routes

• Added a feature to the simulator to terminate an episode if the aircraft collide

• Simulator is available on IOS, android, and web-based application

• We trained our model using the web-based application
CASE STUDY

• Optimal Solution

• All aircraft had to be on goal position when time remaining = 0

• Agent had to maintain safe separation and avoid aircraft collision

• Agent had to choose the correct route combination

• **Agent had to select the correct speed for both aircraft at every time-step**
CASE STUDY

• Feasible Solution

• All aircraft avoided conflicts but did not make it to goal position

• Aircraft could arrive to goal position if given more time
ENVIRONMENT SETUP

• Needed to retrieve and update information from the agent to the online game

• Difficult because we do not know the underlying model for the game

• Solution:
  • Map each action from the agent to a sequence of automated mouse-clicks
ENVIRONMENT SETUP

- Python package: pyautogui used for automated mouse clicks
- Input was pixel location of mouse click
- Start, pause, restart, speed values, and route changes could be calculated one time and stored
- Opening the speed menu had to be calculated at every time step
ENVIRONMENT SETUP

- Determining aircraft position
  
  - Python’s OpenCV module
    
    - Allowed us to use template matching to determine where the most probable position of the aircraft is
    
    - Template matching required 2 images
      
      1. Background Image (Sector 33)
      
      2. Image to find (aircraft)
ENVIRONMENT SETUP

- Image to find

- Background Image

AAL12
600 kts
ENVIROMENT SETUP

• Result of Template Matching
ENVIRONMENT SETUP

• Result of Template Matching
ENVIRONMENT SETUP

• Result of Template Matching

Probable Locations
ENVIRONMENT SETUP

• Reinforcement learning checklist…

1. State - aircraft position and speed
2. Action - agent can change route and speed
3. Reward - penalty assigned for each action and state
4. Agent - uses state, action, and reward to learn the optimal policy
ENVIRONMENT SETUP

- Reinforcement learning checklist...

1. State - aircraft position and speed
2. Action - agent can change route and speed
3. Reward - penalty assigned for each action and state
4. Agent - uses state, action, and reward to learn the optimal policy
ENVIRONMENT SETUP

• Reinforcement learning checklist…

1. State - aircraft position and speed
2. Action - agent can change route and speed
3. Reward - penalty assigned for each action and state
4. Agent - uses state, action, and reward to learn the optimal policy
ENVIRONMENT SETUP

• Reinforcement learning checklist…

1. State - aircraft position and speed
2. Action - agent can change route and speed
3. Reward - penalty assigned for each action and state
4. Agent - uses state, action, and reward to learn the optimal policy
ENVIRONMENT SETUP

• Reinforcement learning checklist…

  1. State - aircraft position and speed
  2. Action - agent can change route and speed
  3. Reward - penalty assigned for each action and state
  4. Agent - uses state, action, and reward to learn the optimal policy
HIERARCHICAL DEEP AGENT ALGORITHM

Algorithm 1 Hierarchical Deep Agent

Initialize: Parent Agent
Initialize: Child Agent
Initialize: $s_P$
reward = 0
number of episodes = $n$

for $i = 1 \text{ to } n$ do
    $a_P = \text{ChooseAction}(\text{ParentAgent})$
    $s_C = [s_P, a_C]$
    repeat
        $a_C = \text{ChooseAction}(\text{ChildAgent})$
        $s', r_C = \text{SimulateEnvironment}$
        receive parent agent reward = $r_P(s')$
        reward = reward + $r$
        update($\text{ChildAgent}$)
    until Terminal
    update($\text{ParentAgent}$)
end for
RESULTS

- Agent converged to optimal solution:
  - aircraft arrived at goal position
  - maintained safe separation and sequencing
  - chose optimal route combination
  - chose optimal aircraft speed
RESULTS

• Plot of the agent’s score throughout the training process

• Can see that the agent is exploring the environment by the change in score

• Agent converges based on epsilon greedy search strategy to the optimal solution
RESULTS

• Convergence time was around 7k episodes

• Final policy proved optimal as shown after 7k episodes
SUMMARY & FUTURE WORK

• Novel hierarchical deep reinforcement learning algorithm introduced is capable of solving complex problems

• Investigated the feasibility and performance of autonomous aircraft sequencing and separation using deep reinforcement learning

• Provides a potential solution framework to enable autonomous sequencing and separation

• Plan to extend this work to more advanced simulators that can model operational uncertainty and allow for more aircraft
REFERENCES


