Predictive Simulation for UAV Flight Planning

Saniyah Shaikh (University of Pennsylvania, Sophomore)
Mentors: Dr. Clark Taylor (AFRL / RYAT)

Motivation

Google DeepMind recently created a program called AlphaGo, which employed deep neural networks with a Monte Carlo Tree Search (MCTS) algorithm, to defeat a human professional player at the game of Go (Silver et al. 2016). We wanted to explore the modifications made to MCTS in AlphaGo to determine if any such algorithm could be viable when applied to flying one or many Unmanned Aerial Vehicles (UAVs).

Phase I: MCTS Algorithm

Objective: Implement a general form of the MCTS algorithm

Approach: Using the pseudocode provided by Browne et al. 2012, created a generic Python 3.6 file that would return a next move given some board object by using the Upper Confidence Bounds for Trees (UCT) method of selection.

Challenges: The pass-by-object-reference nature of Python caused some issues with modifying the boards of the tree in the expand and playout steps.

Result: Tic Tac Toe and Connect Four games with MCTS AI

Phase II: Heuristic Playouts

Objective: Implement a variety of heuristics as playouts in the MCTS algorithm and evaluate their effectiveness

Approach: Coded three heuristics (regular, defensive, and focused) for each game, and created a variety of playout policies that use the heuristics in different ways (inspired by Drake and Uurtamo 2007). Created data-gathering code to evaluate the effectiveness of each heuristic or playout policy by playing many games with the same heuristic, playout policy, number of playouts, and type of opponent.

Challenges: During the simulations, after reaching a node which would guarantee a win for the opposing player due to the heuristic used, instead of halting exploration, the EXPAND function would expand the node by giving it a random child, which resulted in some playouts that were not very feasible for the given heuristic. This issue was fixed by redefining the definition of a terminal node in the MCTS algorithm, and by adding such nodes to the tree immediately. However, there was no significant difference in algorithm performance.

Result: Although the regular heuristic more accurately approximates a human playing the game, the focused heuristic yielded the highest win rates. The HFIRST playout policy seemed to win most, although it achieved maximal performance at different n values for each type of heuristic.

Phase III: UAV Flight Planning

Objective: Apply the MCTS algorithm to planning the flights of one or more UAVs while trying to capture intruders.

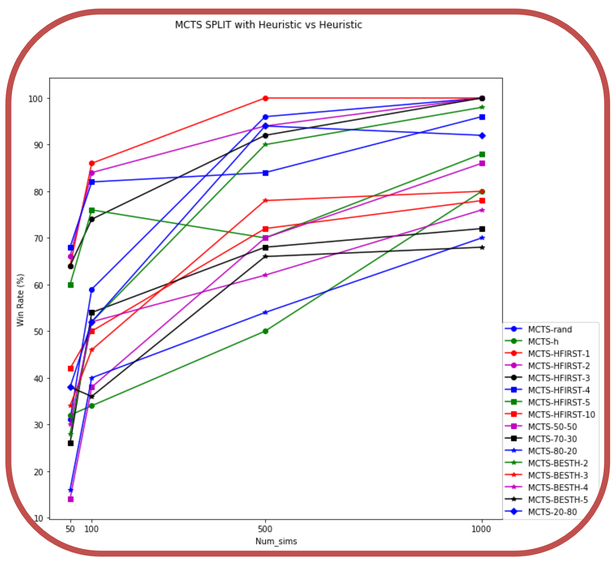
Approach: Created and coded a game called 'Defend the Base' to simulate defending a location from a number of intruders. The MCTS player wins if all intruders are captured by being seen in a UAV scan while on the perimeter fence, and the attacking player wins if the intruders are on the fence and outside of any UAV's scan range.

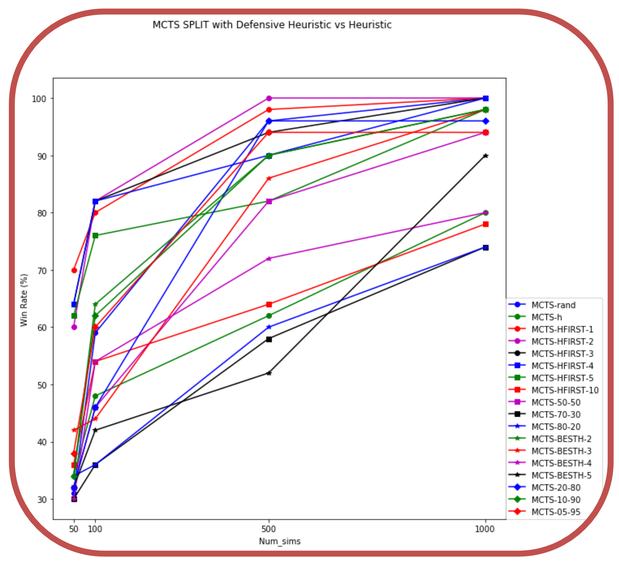
Challenges: Intruders were disappearing from the board just before being captured and were not reflected in the score.

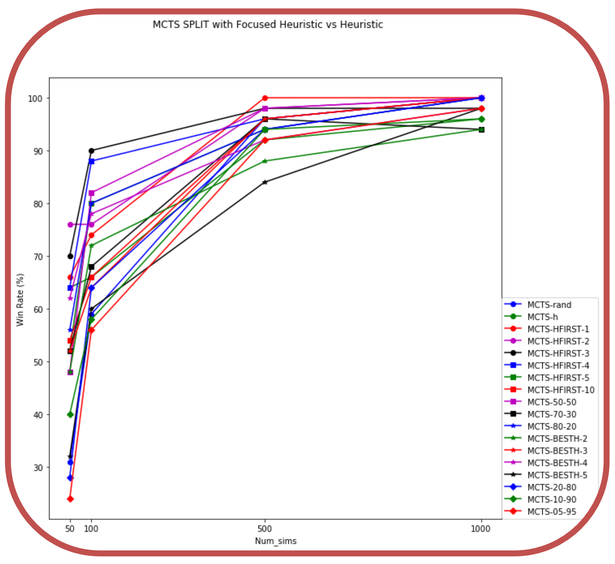
Result: Code that simulates the game between an MCTS-controlled UAV or swarm of UAVs, and a random set of intruders. The algorithm is able to win quickly when there fewer intruders, and code is currently being completed to prevent a swarm of UAVs from scanning the same area. Video of gameplay was produced.

Contributions

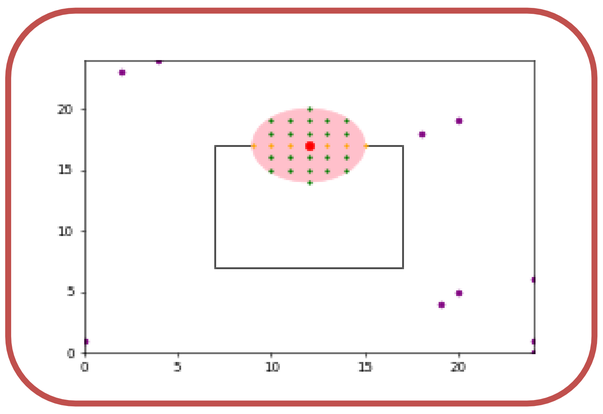
- Python codebase
  - General MCTS algorithm
  - Working Tic Tac Toe with MCTS AI
  - Working Connect Four with MCTS AI
  - Defend the Base simulation
    - Can change number of UAVs, UAV radius, base position, radius, and board size
  - Data-producing code to evaluate effectiveness of changes
- Data on effectiveness of different heuristics used in MCTS
- Videos, images, or text output from hundreds of games



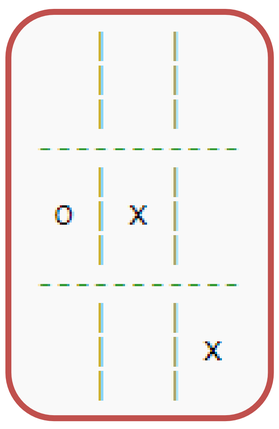




Connect Four Playout Policy Comparisons:
TL: MCTS with Regular Heuristic; TR: MCTS with Defensive Heuristic; BL: MCTS with Focused Heuristic

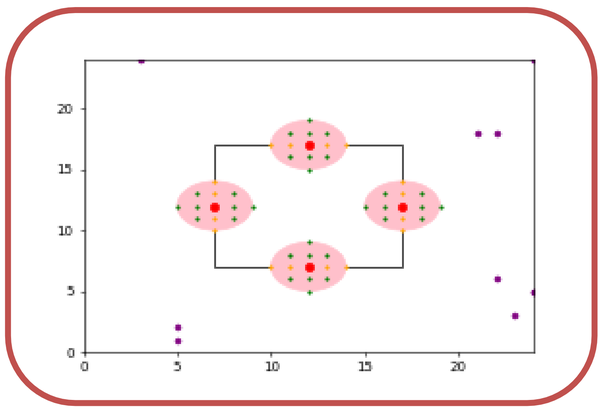


Single UAV vs 10 opponents



Original board state

Incorrect playout



4 UAVs vs 10 opponents

Future Challenges

- Evaluate capture rates of single UAV vs multiple on square, line, and circular boards
- Evaluate effects of changing board size or UAV radius on capture rate
- Adjust for uncertainty or partial information in intruder locations
- Real-life application:
  - Figure out how to coordinate movement among a swarm of UAVs using the MCTS algorithm
  - Test the algorithm with a swarm of UAVs

DISTRIBUTION STATEMENT C. Distribution authorized to U.S. Government agencies and their contractors.