THE AIR FORCE RESEARCH LABORATORY
LEAD | DISCOVER | DEVELOP | DELIVER

Predictive Simulation For UAV Flight Planning

- **ATR Center Summer Workshop**
- **9,10 August 2017**

University of Pennsylvania

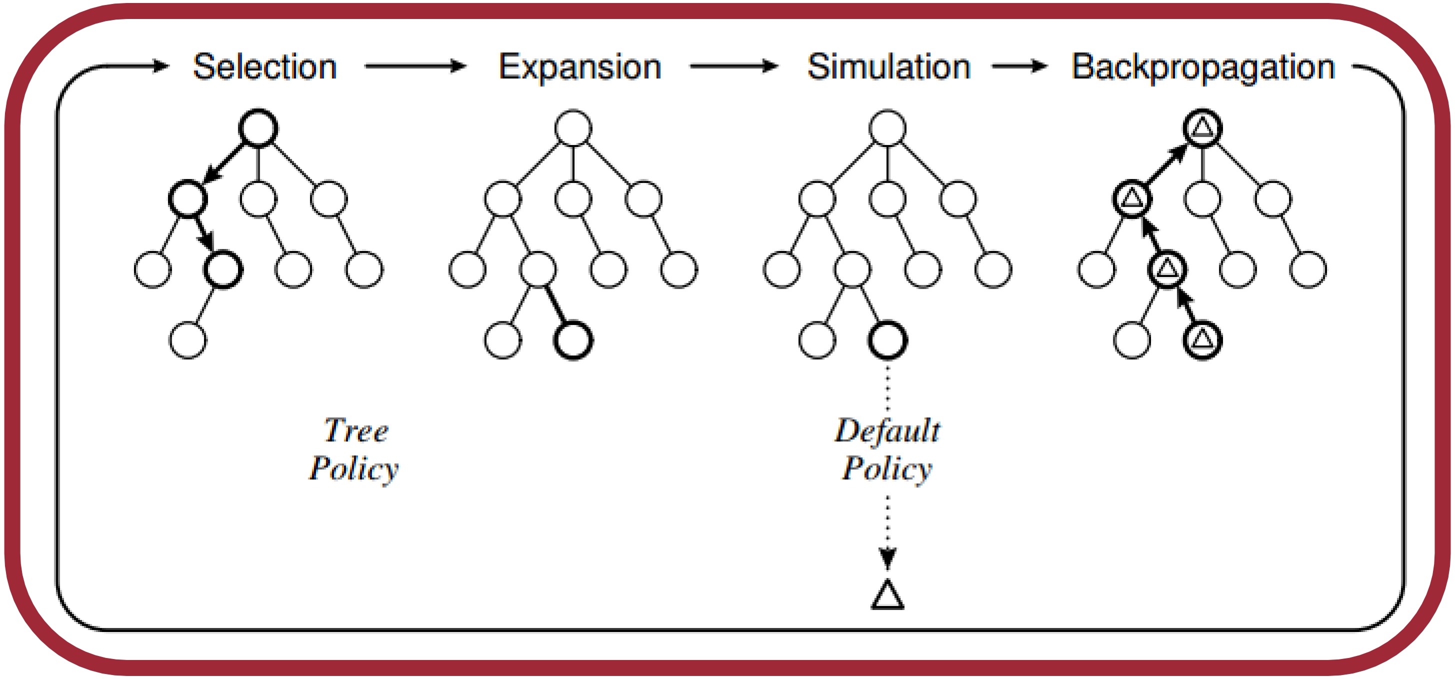
Saniyah Shaikh, University of Pennsylvania
Mentor: Dr. Clark Taylor, AFRL / RYAT

Cleared for Public Release: 88ABW-2017-6180

Motivation & Objective

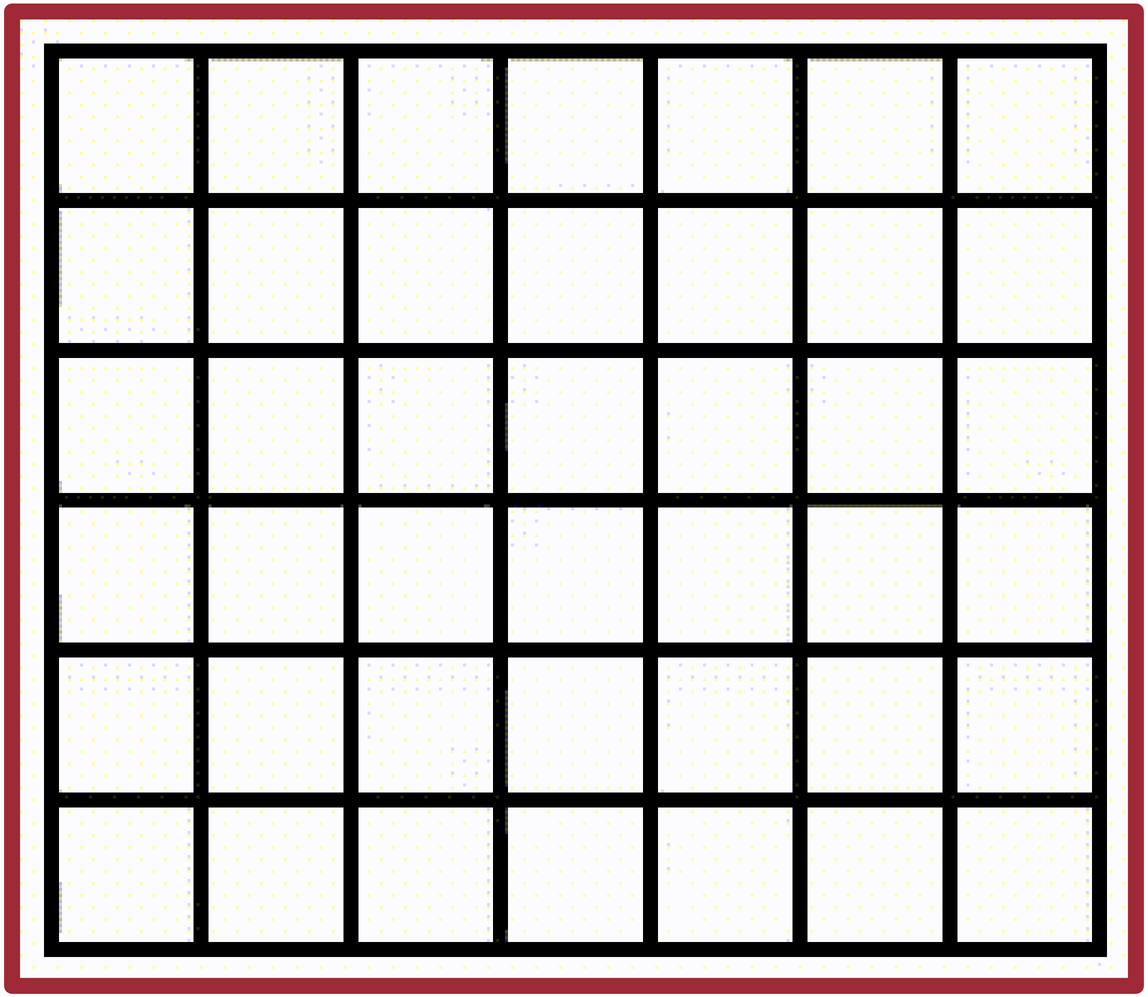
- **Motivation**
  - Google DeepMind’s AlphaGo defeated a human professional player at the game of Go
- **Objective**
  - Explore the modifications made to Monte Carlo Tree Search (MCTS) in AlphaGo to determine if any such algorithm could be viable when applied to flying one or many Unmanned Aerial Vehicles (UAVs)

Monte Carlo Tree Search (MCTS)



For each iteration of the MCTS algorithm, a tree policy (in this case, Upper Confidence Bounds for Trees, or UCT) is used to **select** a non-terminal node in the current tree. The node is **expanded** by adding a random child, a **simulation** is run from the child to a terminal state to obtain a reward, and **backpropogation** updates the tree with the reward. Each such iteration is called a playout.

Monte Carlo Tree Search (MCTS)



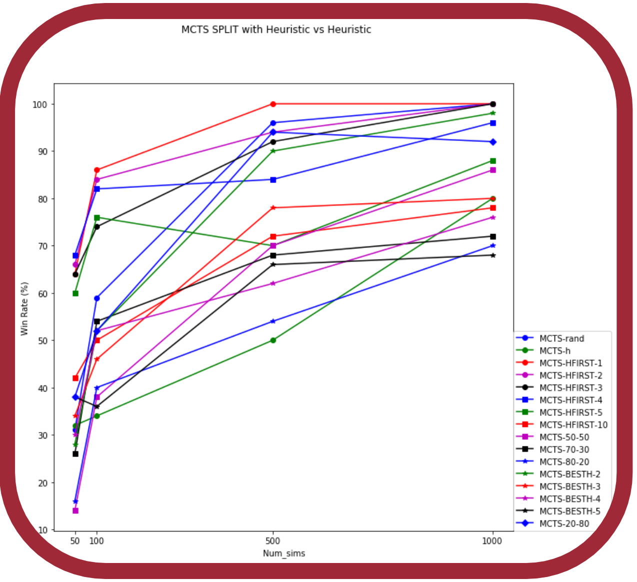
MCTS (Y) vs Human Player (R)
1000 playouts

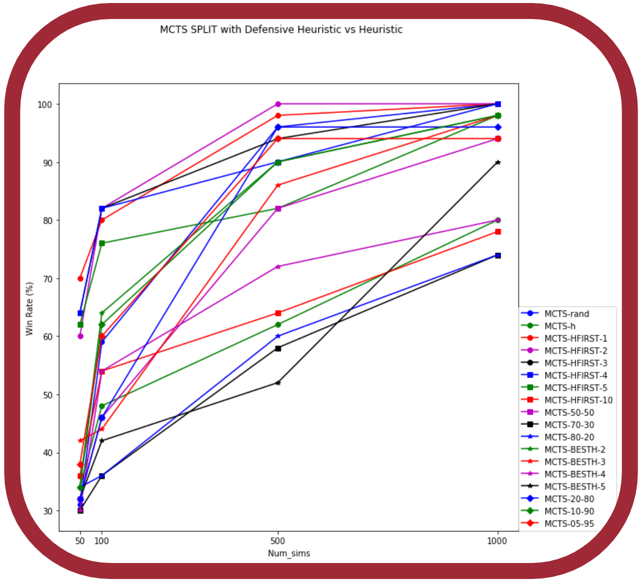
Playout Policies & Heuristics

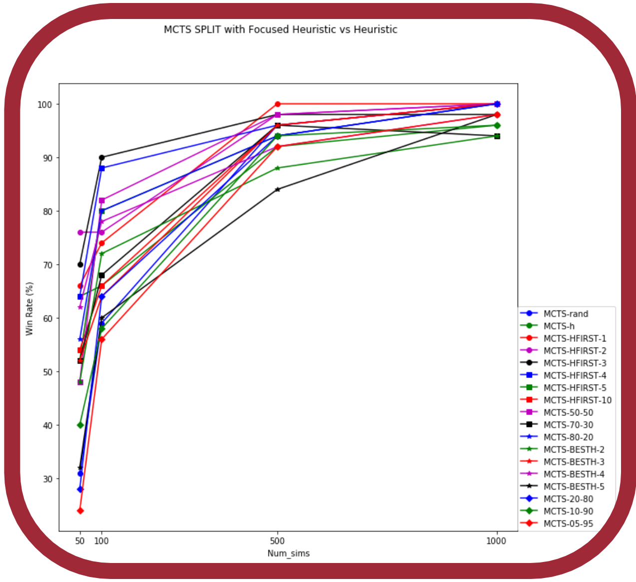
	Description
HPOLICY	Uses the heuristic policy to select moves until the board state is terminal, then returns the reward
HFIRST POLICY(n)	Uses heuristic move selection for first n moves, then uses random moves to a terminal board state, and returns the reward
BESTH POLICY(n)	For each move in the rollout, generates a set of n random moves of the current set of possible moves, then chooses the best one using the heuristic. Continues this selection process until a terminal state is reached, then returns the reward
PERC PLAYOUTS(n)	Applies HPOLICY for n percent of the playouts, uses DEFAULTPOLICY the rest of the time

	Regular Heuristic	Defensive Heuristic	Focused Heuristic
Description	Picks a move by attempting to win, attempting to block, then selecting randomly, close to the center	Picks a move by attempting to block, attempting to win, then selecting randomly, close to the center	Picks a move by selecting randomly, close to the center

Policy & Heuristic Comparisons







Connect Four Playout Policy Comparisons:

MCTS with Regular Heuristic – HFIRST-1

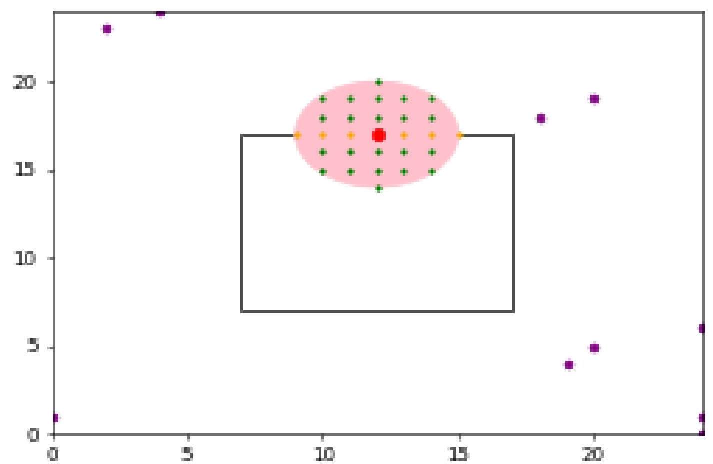
MCTS with Defensive Heuristic – HFIRST-2

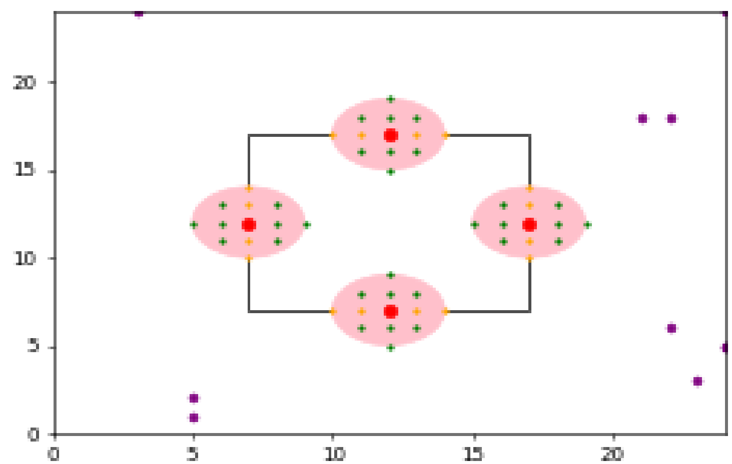
MCTS with Focused Heuristic – HFIRST-3, HFIRST-1

Focused Heuristic has best performance alone

UAV Flight Planning

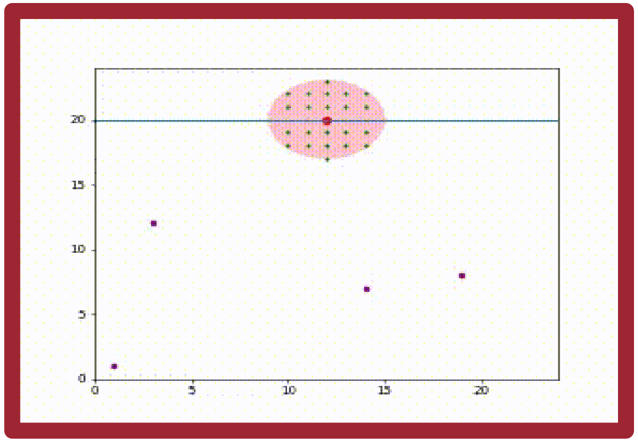
- Created a game to simulate a base being attacked
- Rewards determined by how many intruders were caught before an intruder reached the perimeter fence
- Made improvements by allowing multiple UAVs and preventing UAVs from having overlapping scanning zones



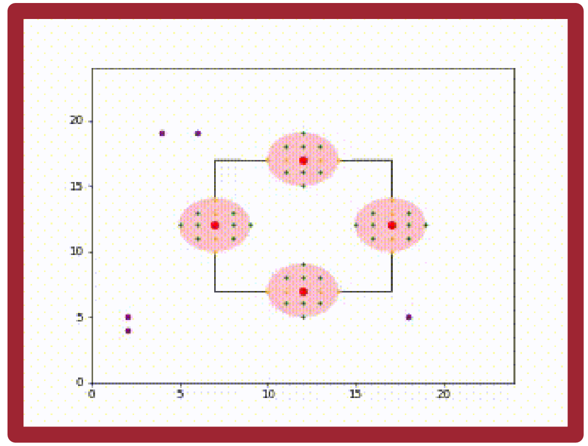


UAV Flight Planning

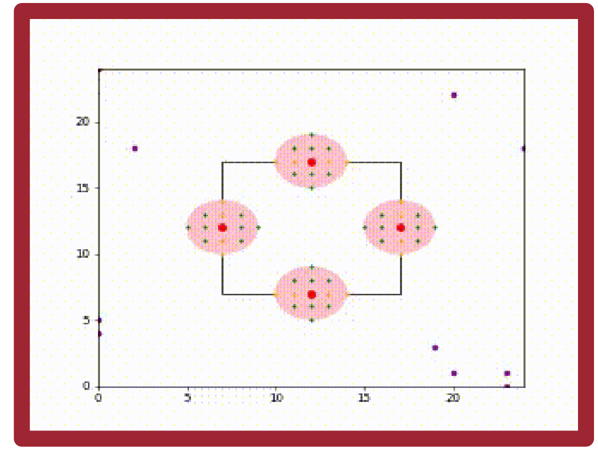
UAV Collisions Permitted:



1000 playouts, 4 opponents

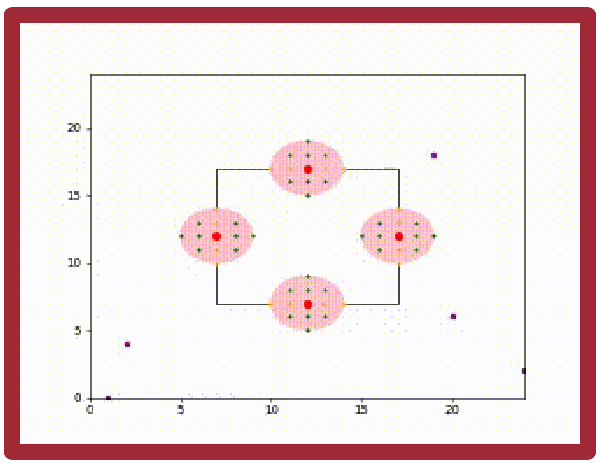


800 playouts, 5 opponents

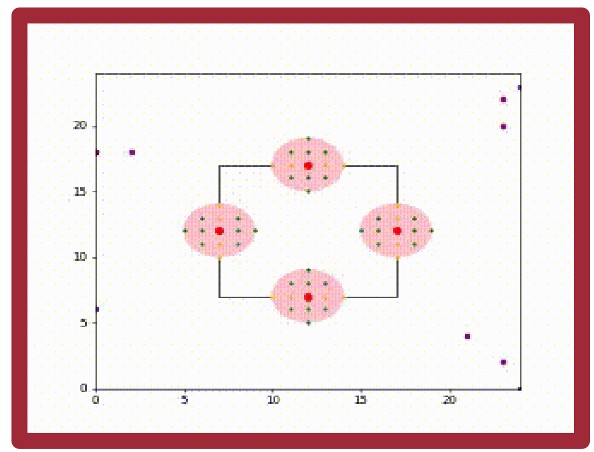


1000 playouts, 10 opponents

UAV Collisions Prohibited:



800 playouts, 5 opponents



800 playouts, 10 opponents

Contributions & Future Work

- **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
- **Future Work**
  - 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

AFRL

THE AIR FORCE RESEARCH LABORATORY

LEAD | DISCOVER | DEVELOP | DELIVER

References

- Browne, Cameron B., Edward Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. "A survey of monte carlo tree search methods." *IEEE Transactions on Computational Intelligence and AI in games* 4, no. 1 (2012): 1-43.
- Drake, Peter, and Steve Uurtamo. "Move ordering vs heavy playouts: Where should heuristics be applied in Monte Carlo Go,"." In *Proceedings of the 3rd North American Game-On Conference*, pp. 171-175. 2007.
- Silver, David, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser et al. "Mastering the game of Go with deep neural networks and tree search." *Nature* 529, no. 7587 (2016): 484-489.