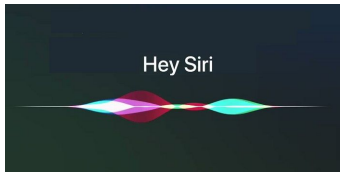
Proving the Convergence of Monte Carlo Tree Search to Brownian Motion

Elana Kozak
United States Naval Academy

Motivation- Machine Learning



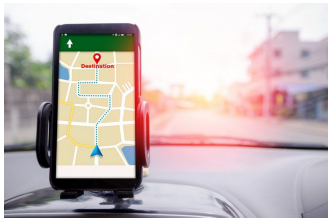
Have you ever played a game against a computer?

Have you ever talked to Siri or Alexa?

Have you ever used GPS to estimate travel time?

Has Facebook ever suggested new friends for you?

Has Amazon ever suggested a new product for you?



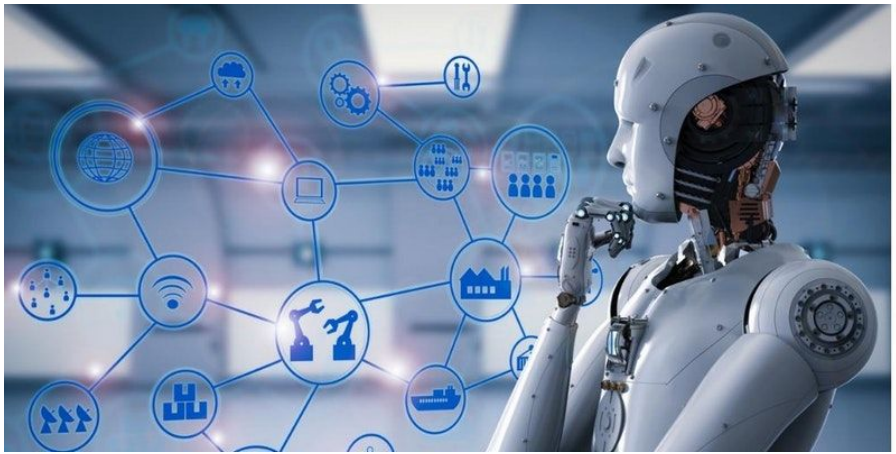
Military Applications

- Autonomous warfare platforms
- Cybersecurity programs
- Logistics and transportation
- Target recognition
- Combat simulation and training
- ISR missions
- Data processing
- Search and rescue

From MarketResearch.com

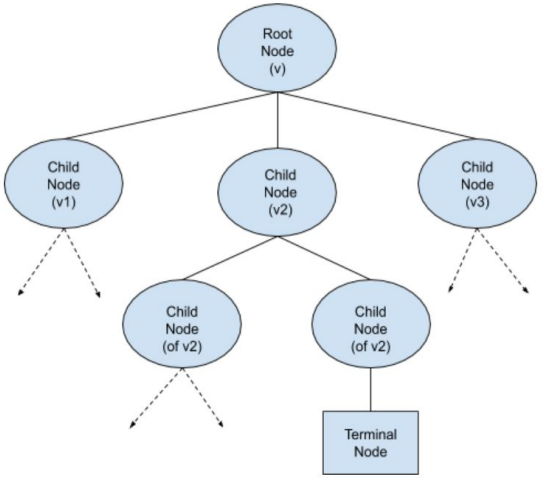
AI Decision Methods

- Random
- Cheat
- Script
- **Monte Carlo Tree Search**



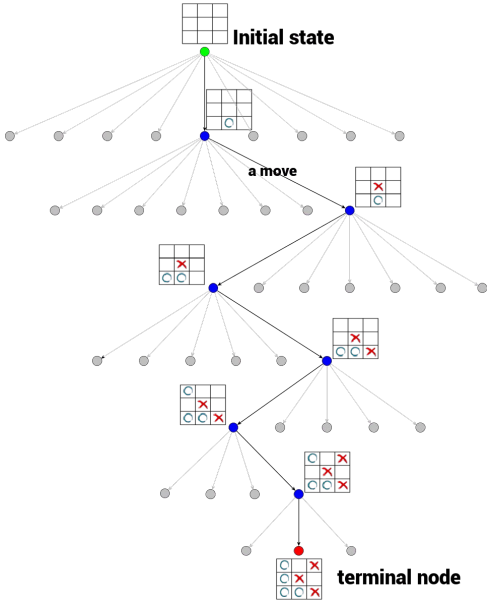
From oreilly.com

“Game” or Decision Tree



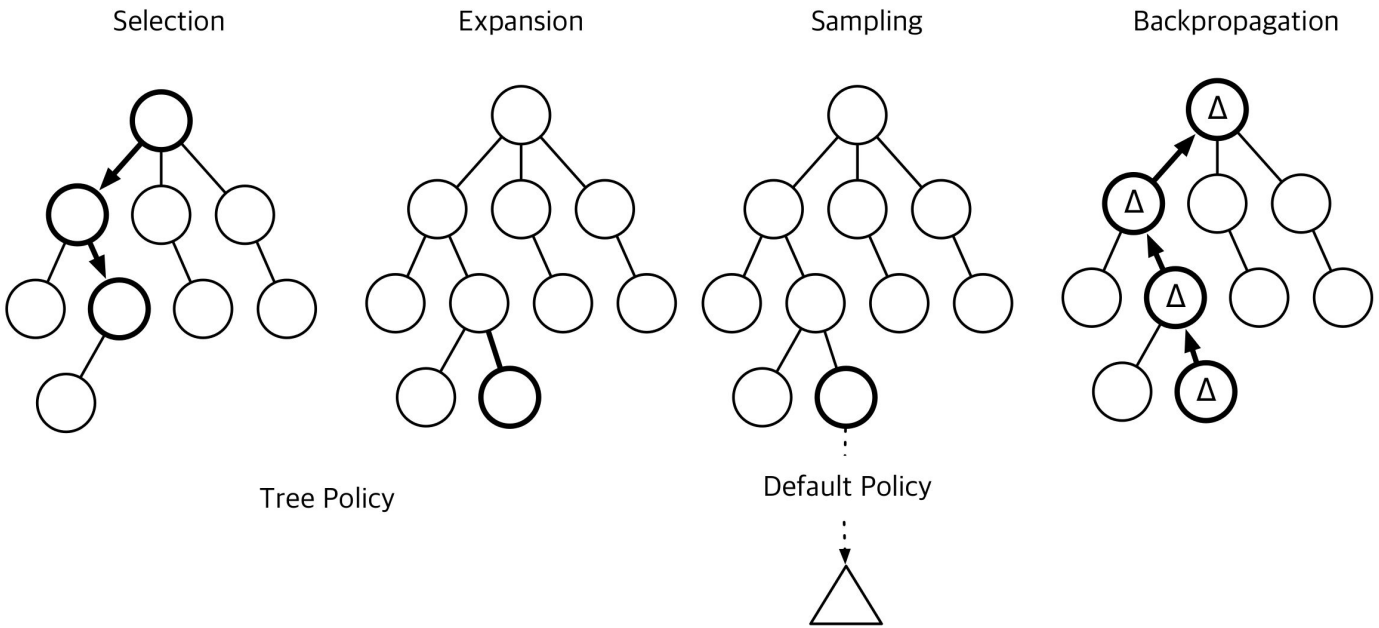
Generic Tree

Game state
Root node (v)
Child nodes (v_i)
Terminal node



Tic-Tac-Toe Example

MCTS Steps



From Kelly and Churchill, 2017

Upper Confidence Bound (UCB1)

aka Upper Confidence Bound for Trees (UCT)

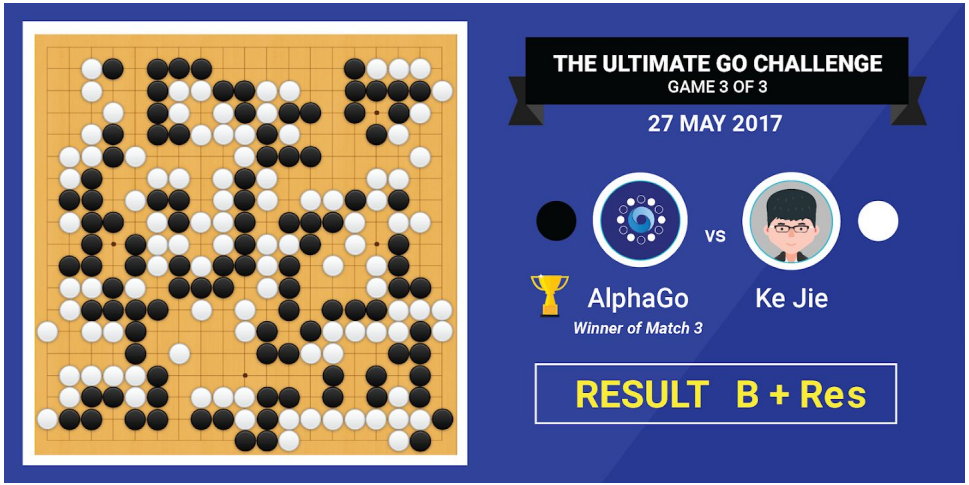
$$\mathbb{UCT}(v_i, v) = \frac{Q(v_i)}{N(v_i)} + c\sqrt{\frac{\log(N(v))}{N(v_i)}}$$

V_i: node
V: parent node
Q: win count
N: visit count
C: exploration constant

From int8.io

Current Applications and Advantages

- Artificial Intelligence (AI) game players
  - Chess
  - Go
  - Tic-Tac-Toe
  - And more…
- Adjustable Computation
  - No initial strategy
  - Only stores end state
  - Set time limit
- But… not always accurate
  - Inherent randomness
  - Doesn't cover all paths

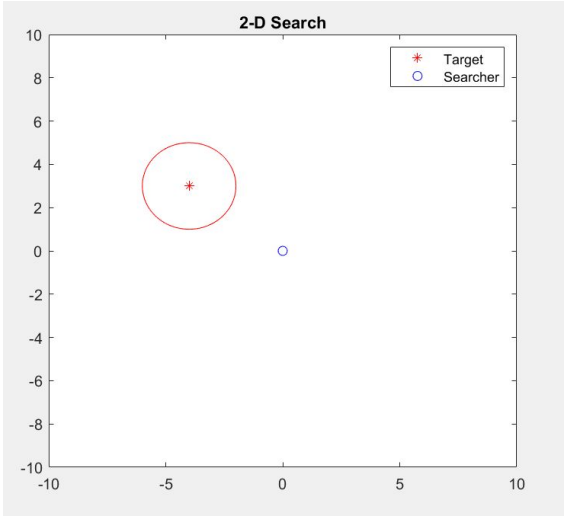


Can we apply MCTS to search and detection?

YES!

Imagine a game…

Moves = up, down, left, right
Goal = find the target



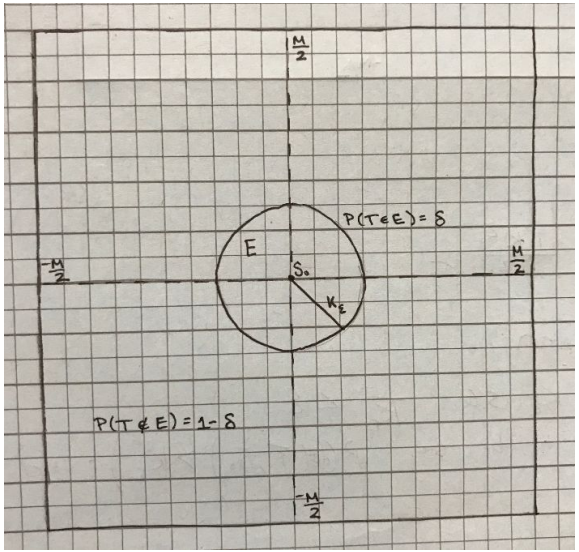
Our question: how does this method behave?

Theorem 1

A 2-D Monte Carlo Tree Search that uses the UCT selection policy and a uniformly random, unknown target will converge to a symmetric random walk as M, the size of the search lattice, goes to infinity.

Proof

- Let ε>0 and choose K(ε) such that (1/K(ε)) < ε as the radius of a region E around the origin
  - Thus K(ε) is the minimum number of steps required to exit this region
- Choose M as the dimension of the square grid such that P(dist(T, S(0))> K(ε)) = 1- δ
- Q = 1/k represents the success rate
  - On average, k >> K(ε) so Q < 1/K(ε) < ε



Recall:

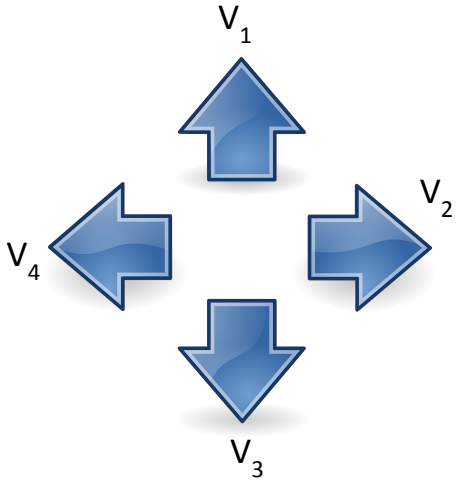
$$UCT(v_i, v) = \frac{Q(v_i)}{N(v_i)} + c\sqrt{\frac{log(N(v))}{N(v_i)}}$$

Proof (continued)

- N(v) is the same for all v_i

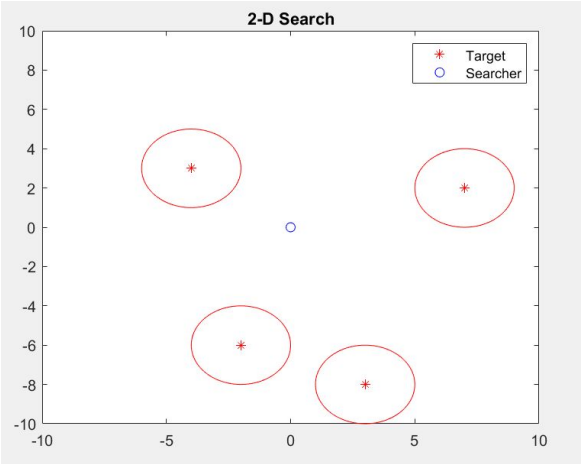
1. First four trials pick *i* randomly, then UCT is equal for all *i*
2. Visited nodes have a lower UCT, so next move is chosen randomly from remaining nodes
3. Process repeats, randomly cycling through the moves since UCT is always equal

Recall: $$UCT(v_i, v) = \frac{Q(v_i)}{N(v_i)} + c\sqrt{\frac{log(N(v))}{N(v_i)}}$$



Future Work

- Theorem 2: When a stationary target is known, a 2-D Monte Carlo Tree Search will converge to an optimal “straight” line path as the number of iterations goes to infinity.
- Test MCTS in more complex scenarios
  - More targets
  - More searchers
  - Different distributions
- How does MCTS compare to other search methods?
  - Time, accuracy, computational complexity, etc.
- What real-world scenarios can we apply MCTS to?
  - Search and rescue
  - Animal foraging
  - Submarine detection



Thank You