# Dark Chess

Cs 695: Decision Making & Reinforcement Learning

Billy Ermlick

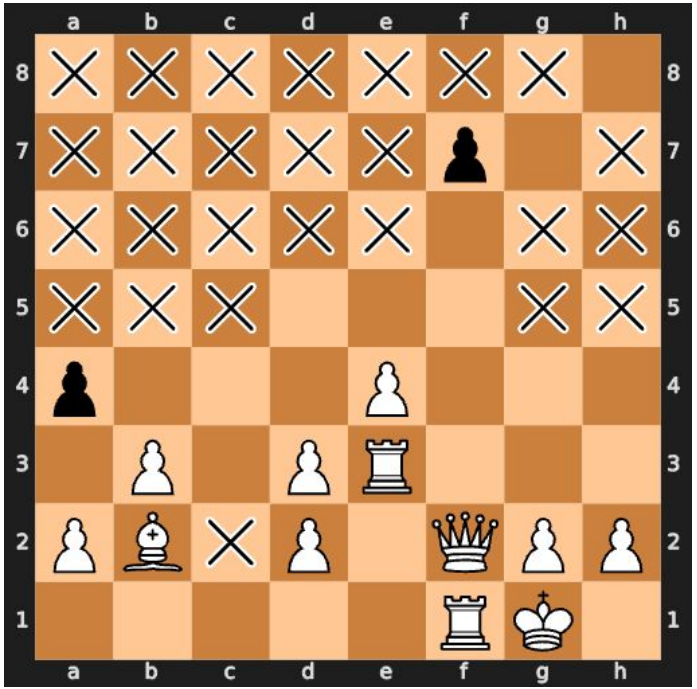
# What is Dark Chess?

**Chess Variant Proposed in 1989: “fog of war” chess**

**Online at chess.com:**
**https://www.chess.com/terms/fog-of-war-chess**

**Rules:**

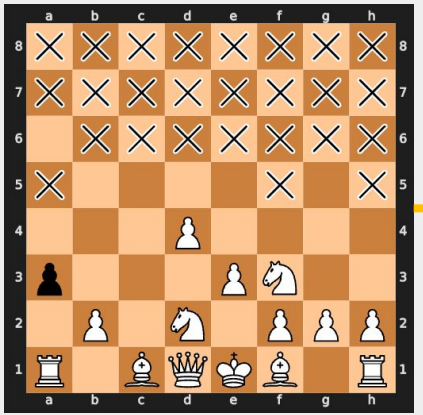
- **Capture the King to win**
- **No Checks**
- **Can only see where your pieces can move**



# Observation - Action - Observation

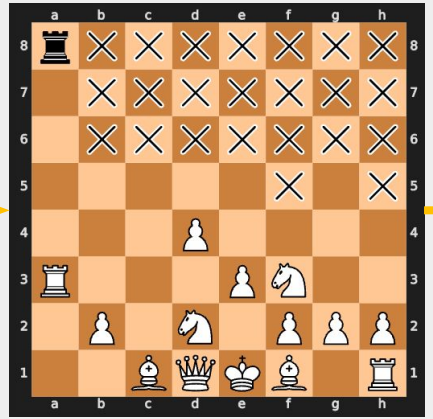
Play for each side follows this O-A-O pattern each turn

White Observation



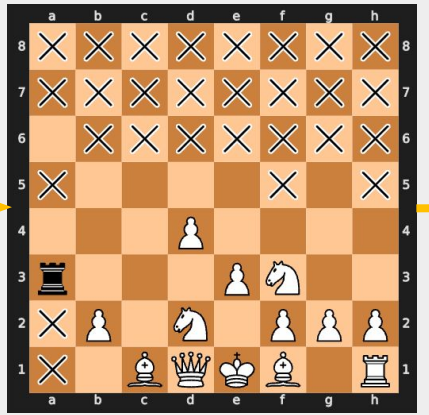
White Action

White Observation



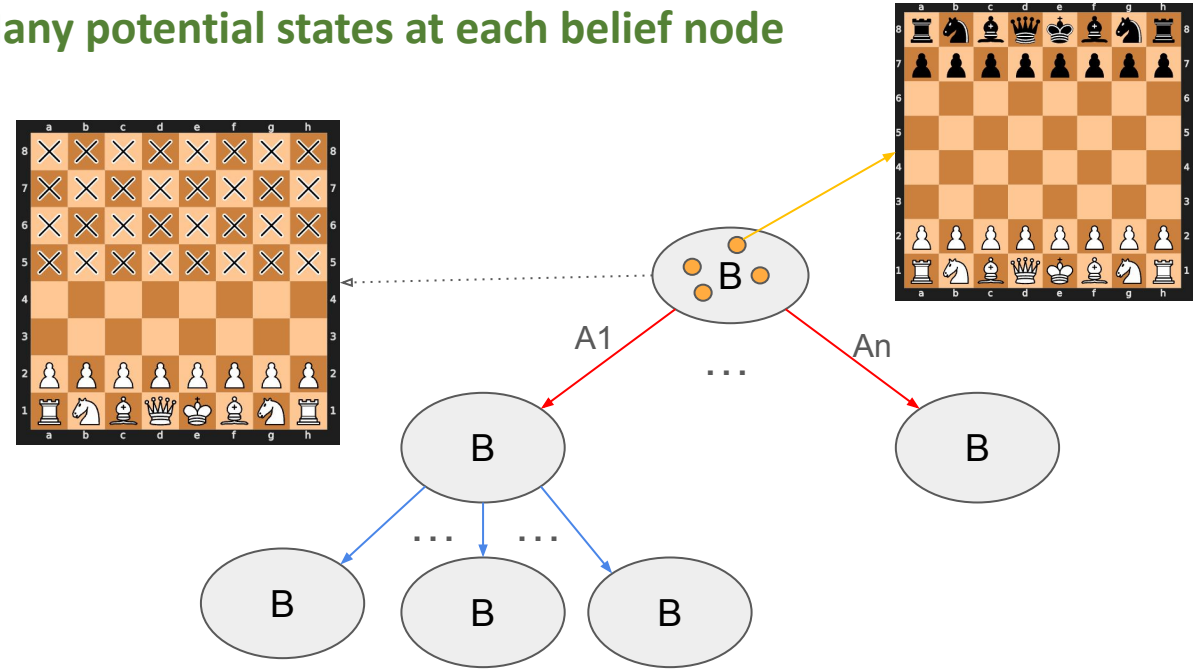
Black Action

White Observation



# Belief Tree Expands Quickly

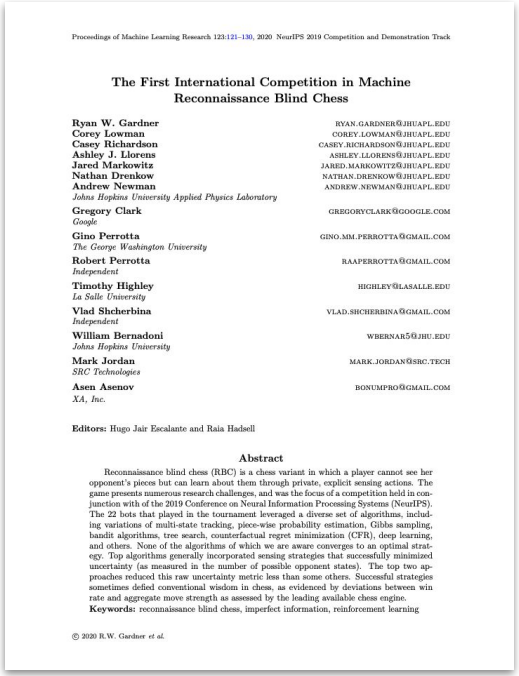
- Due to large amount of unobserved squares and large action space tree expands quickly
- Many potential states at each belief node



# Belief Tree Expands Quickly

- Partially Observable Markov Decision Process (POMDP)
- Chess has between ~10^40 legal board states…
- Many squares are unobserved
  → Lots of uncertainty in belief space

- Reconnaissance Blind Chess is a similar game
  - Competition hosted by NeurIPS 2019
  - Separate sensing action
  - Hosted Platform



Proceedings of Machine Learning Research 123:121–130, 2020 NeurIPS 2019 Competition and Demonstration Track

**The First International Competition in Machine Reconnaissance Blind Chess**

**Ryan W. Gardner** RYAN.GARDNER@JHUAPL.EDU
**Corey Lowman** COREY.LOWMAN@JHUAPL.EDU
**Casey Richardson** CASEY.RICHARDSON@JHUAPL.EDU
**Ashley J. Llorens** ASHLEY.LLORENS@JHUAPL.EDU
**Jared Markowitz** JARED.MARKOWITZ@JHUAPL.EDU
**Nathan Drenkow** NATHAN.DRENKOW@JHUAPL.EDU
**Andrew Newman** ANDREW.NEWMAN@JHUAPL.EDU
*Johns Hopkins University Applied Physics Laboratory*
**Gregory Clark** GREGORYCLARK@GOOGLE.COM
*Google*
**Gino Perrotta** GINO.MM.PERROTTA@GMAIL.COM
*The George Washington University*
**Robert Perrotta** RAAPERROTTA@GMAIL.COM
*Independent*
**Timothy Highley** HIGHLEY@LASALLE.EDU
*La Salle University*
**Vlad Shcherbina** VLAD.SHCHERBINA@GMAIL.COM
*Independent*
**William Bernadoni** WBERNAR5@JHU.EDU
*Johns Hopkins University*
**Mark Jordan** MARK.JORDAN@SRC.TECH
*SRC Technologies*
**Asen Asenov** BONUMPRO@GMAIL.COM
*XA, Inc.*

**Editors:** Hugo Jair Escalante and Raia Hadsell

**Abstract**

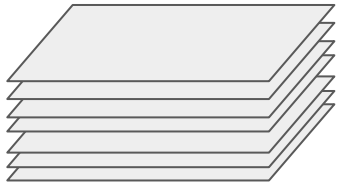
Reconnaissance blind chess (RBC) is a chess variant in which a player cannot see her opponent's pieces but can learn about them through private, explicit sensing actions. The game presents numerous research challenges, and was the focus of a competition held in conjunction with of the 2019 Conference on Neural Information Processing Systems (NeurIPS). The 22 bots that played in the tournament leveraged a diverse set of algorithms, including variations of multi-state tracking, piece-wise probability estimation, Gibbs sampling, bandit algorithms, tree search, counterfactual regret minimization (CFR), deep learning, and others. None of the algorithms of which we are aware converges to an optimal strategy. Top algorithms generally incorporated sensing strategies that successfully minimized uncertainty (as measured in the number of possible opponent states). The top two approaches reduced this raw uncertainty metric less than some others. Successful strategies sometimes defied conventional wisdom in chess, as evidenced by deviations between win rate and aggregate move strength as assessed by the leading available chess engine.

**Keywords:** reconnaissance blind chess, imperfect information, reinforcement learning

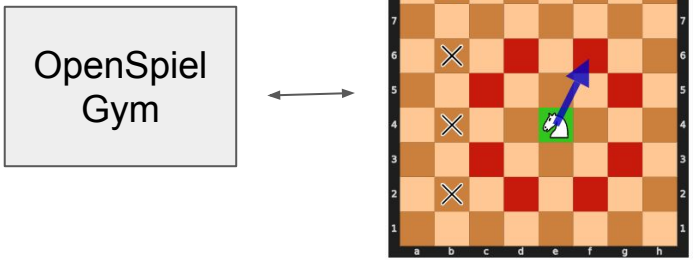
# Environment Setup

- Few poorly made environments out there
- OpenSpiel had a C++ version implemented with poor state management
- Needed ways to compute legal chess moves/captures + also have a tensor based state for quick comparisons and belief tracking

Numpy-array - 8x8x23



- Used a combination of
  - Numpy arrays → custom board + belief/observation states
  - Python-chess → visualizations and candidate move evaluation
  - OpenSpiel → base game rules and environment advancement



# Four Dark Chess Agents

1) Random Agent

- Just make a random move

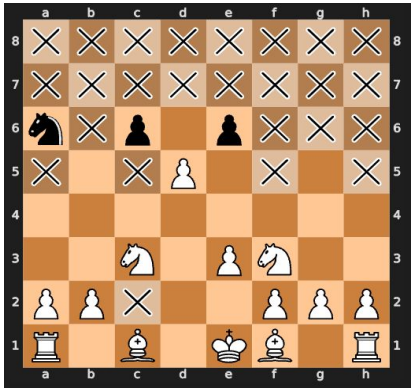
2) Greedy Agent

- If they can take a piece, take it
  - Takes highest worth piece capturable
- Conservative Variant - only take the piece if it is worth more than your piece
- Preservative Variant - more heavily consider taking if your piece is attacked as well

# Four Dark Chess Agents

**3) Uniform Belief State Agent (UBSA)**

- Keep track of taken pieces
- Predict all unseen pieces to be uniformly distributed in unseen space
- Sample many possible board from belief space
- Best Variant) get best StockFish move for each sampled board and greedy move and get most common move, choose greedy if no consensus
    - Others variants considered board evaluation across all sampled board after making moves, but stockfish was too slow

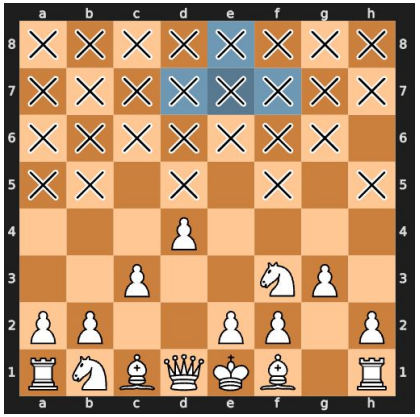


UBSA uniform prediction of enemy King location

# Four Dark Chess Agents

## 4) Progressive Belief State Agent (PBSA)

- Keep track of taken pieces
- Each observation, update the belief state with the new seen squares
- To update the belief state after opponents turn:
  - If we saw their move: update only the seen/unseen squares of the piece type that made the move
  - Else: sample many possible board based on last belief state and play random/greedy moves on each of them, use this distribution as new belief state
- Select action for belief state same as (UBSA) agent



PBSA prediction of enemy King location based on particle filter

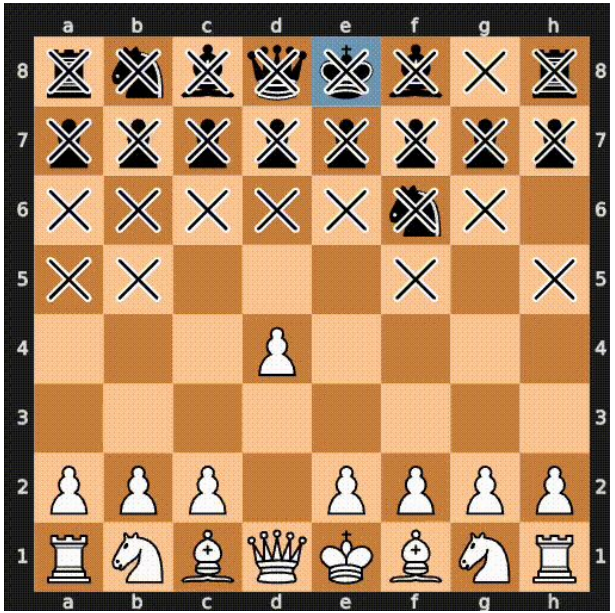
# Better Agent, Better Performance

**Compared agents using head to head performance over a set amount of games**

- **Random Agent v. Greedy Agent → Greedy agent wins 99.8% out of 1k games**
- **UBSA v. Greedy Agent → UBSA wins 70% of 100 games**
- **PBSA v. Greedy Agent → PBSA wins 93% of games of 50 games**
- **PBSA v. UBSA → PBSA wins 77% of 20 games**

# Takeaways

- Stockfish helps agents make moves that increase vision and keeps pieces protected, but definitely not the best move for dark chess
- Suicide attempts still happen even in PBSA
    - likely due to lack of simulations or the fact that we are taking the most commonly suggested stockfish move after the fact
- Greedy agent takes advantage of conservative version 70% win rate
- A conservative agent that doesn't take any pieces may have good chances – need to get rid of stockfish



# Conclusion / Next Steps

- Clean code & bugs
- Move on to POMCP & DESPOT Tree Based Planners
  - Started on this then realized I did not have the belief state updating mechanics in place
- Use learning to find common belief state transitions instead of particle sampling
  - Value and policy network like alpha zero
- State tensor could be expanded to include different planes for light and white square bishop
- Code is slow see why parallelization and C++ is used for these things
  - Currently stockfish evaluation of positions is major bottleneck & stockfish crashes a fair amount