Computing Science (CMPUT) 455

Search, Knowledge, and Simulations

James Wright

Department of Computing Science
University of Alberta
james.wright@ualberta.ca

Fall 2021

455 Today - Last Lecture

- Final exam preview
- Computer Poker - especially UofA Poker programs Cepheus and DeepStack
- Alpha Zero for chess and shogi
- Other applications of modern heuristic search
- Summary of the course
- Final exam study guide now available
- Coursework: Last Quiz 13 due **Mon Dec 13**
- Coursework: Assignment 4 due **Tue Dec 14**

Timeline - Last Steps

- **Thu Dec 9 (Day after tomorrow):** USRI closes
- **Mon Dec 13:** Quiz 13 due
- Tue Dec 14: Quiz 13 review posted
- Tue Dec 14: Assignment 4 deadline
- Thu Dec 16: Assignment 4 late submission deadline
- Dec 17–20: Finish Assignment 4 marking vs fixed opponents, then start 455 championship
- Mon Dec 20: Final exam on eClass

Final Exam

- 3 hours, on eClass, Dec 20
- Format: similar to midterm, but time for some more involved questions
- Have paper and pencil available, some questions will require you to do some work on paper
- Detailed study guide now on webpage, summary on next slide

Final Exam

- All new material after midterm cutoff (Lecture 11 and later)
- Old material is excluded
- This week's material:
  - Alpha Zero, Poker programs: included
  - Other heuristic search topics: excluded

Quiz 12 review

68 attempts, average 92%

Question 11 (extra): Do you have any follow-up questions or comments? What do you want to learn more about? How does the topic relate to other things you have learned? Which topics or articles do you want to read next? Etc.

Quiz 12 review

68 attempts, average 92%

Question 11 (extra): Do you have any follow-up questions or comments? What do you want to learn more about? How does the topic relate to other things you have learned? Which topics or articles do you want to read next? Etc.

1. What other games can alpha go/zero be applied to in the future?

Quiz 12 review

68 attempts, average 92%

Question 11 (extra): Do you have any follow-up questions or comments? What do you want to learn more about? How does the topic relate to other things you have learned? Which topics or articles do you want to read next? Etc.

1. What other games can alpha go/zero be applied to in the future?
2. I want to learn more about deep learning.

Quiz 12 review

68 attempts, average 92%

Question 11 (extra): Do you have any follow-up questions or comments? What do you want to learn more about? How does the topic relate to other things you have learned? Which topics or articles do you want to read next? Etc.

1. What other games can alpha go/zero be applied to in the future?
2. I want to learn more about deep learning.
3. How can I implement reinforcement learning in Gomoku?

Quiz 12 review

68 attempts, average 92%

Question 11 (extra): Do you have any follow-up questions or comments? What do you want to learn more about? How does the topic relate to other things you have learned? Which topics or articles do you want to read next? Etc.

1. What other games can alpha go/zero be applied to in the future?
2. I want to learn more about deep learning.
3. How can I implement reinforcement learning in Gomoku?
4. What is the central problem in reinforcement learning?

Quiz 12 review

68 attempts, average 92%

Question 11 (extra): Do you have any follow-up questions or comments? What do you want to learn more about? How does the topic relate to other things you have learned? Which topics or articles do you want to read next? Etc.

1. What other games can alpha go/zero be applied to in the future?
2. I want to learn more about deep learning.
3. How can I implement reinforcement learning in Gomoku?
4. What is the central problem in reinforcement learning?
5. Does adding more neurons [ensure] more [accuracy]?

Computer Poker

- Poker is a very popular family of card games
- Poker was one of the main models for development of mathematical game theory in the 1930s and 1940s
- Imperfect information game - players do not know opponent's cards, or cards that will be drawn in the future
- Our university is one of two major centers of computer poker research
- Carnegie-Mellon University (CMU) is the other
  - Programs: Tartanian, Claudico, Libratus

Recent Poker Publications involving U. of Alberta

- Bowling, Burch, Johanson and Tammelin. Heads-up limit hold'em poker is solved. Science, 2015
  - `http://poker.srv.ualberta.ca`
- Moravcik, Schmid, Burch, Lisy, Morrill, Bard, Davis, Waugh, Johanson and Bowling. DeepStack: Expert-level artificial intelligence in heads-up no-limit poker. Science, 2017
  - `https://www.deepstack.ai`

Game Theory Background

- In perfect information games (e.g. Go), solving a game includes:
  - Computing its *minimax value*
  - Finding a winning strategy
  - Strategy includes one move at OR nodes (your turn), all moves at AND nodes (opponent's turn)
- Imperfect information games (e.g. poker):
  - For games with more than two players, there is no analog to minimax value
    - One player's result depends on what all other players do
  - The most important general concept is the *Nash equilibrium*
  - A player's strategy in general needs to be *mixed*

Pure vs Mixed Strategies

- Pure strategy: in the same state,
  always choose the same action
- Mixed strategy: choose action
  according to a probability distribution
- Example in poker
  Call 80% of the time
  Raise 20% of the time
- In imperfect information games,
  sometimes mixed strategy needed for optimal play
  (e.g. Rock-Paper-Scissors)
- In complete information games such as Go,
  an optimal pure strategy always exists. No need to mix

Mixed Strategies vs Probabilistic Simulation Policies

- In both cases, we pick an action according to a probability distribution
- However, the meaning behind it is very different
- In simulation policy, we want to sample from a huge game tree, get better exploration
- In mixed strategy, we want to avoid being predictable and being exploited by the opponent
- We need to protect our hidden information (e.g. cards) from being inferred too easily

Nash Equilibrium

- Defined for multi-player, non-cooperative games
  - Two-player games included as special case
- A set of strategies for all players
- Assuming they behave “rationally” in a certain sense:
- In Nash equilibrium, no single player can profit from changing their strategy unilaterally
- Many Nash equilibria may exist, and they can be quite different from each other in a multiplayer (more than 2) setting

Imperfect Information Games - Two Player Zero-sum Case

Special case - only two players

- *Minimax value* still exists (in expectation)
- Minimax value is achieved by playing a Nash equilibrium strategy
- May *have* to use a mixed strategy to achieve it
- Example: Rock-Paper-Scissors
  - Choose action randomly, each with probability $1/3$
  - You will get an even result in the long run, no matter what the opponent does
  - Any other strategy could be exploited by the opponent

Imperfect Information and Beliefs

- Because of unknown information, reasoning about players' beliefs is very important
- Try to infer opponent's cards **and** beliefs from their actions
- Try to hide own cards and beliefs
- Search becomes much more complex
- Search now needs to reason over player beliefs, not just over completely known search states

Heads-up Limit Texas Hold'em Poker

- Limit on size of bets
- State space - 10^{14} decision points
- UofA program Cepheus "essentially solved" the game
- Main progress: different approach to satisficing than in previous poker programs
- Remember Herb Simon's quote from many weeks ago:

Herb Simon on Satisficing

"Decision makers can satisfice either by finding optimum solutions for a simplified world, or by finding satisfactory solutions for a more realistic world.
Neither approach, in general, dominates the other, and both have continued to co-exist..."

Cepheus vs Previous Approaches

- Previous poker programs: finding (close to) optimum solutions for a simplified world
- Used abstraction to group many different poker states into the same abstract state
- Example: pair of kings treated the same as pair of aces
- They are both very strong hands. But in some important situations, they are very different
- Cepheus does not do that.
  - It approximates the full game of poker directly

Cepheus Approach

- Learned a full strategy for all 10^{14} decision points
- Used a new approximation method called CFR+
- Used 900 CPU years of (parallel) computation
- The final result is basically one huge lookup table
  - Contains probability of each action in each state
- Error under 1 milli big blind per hand
- Note log scales in picture...

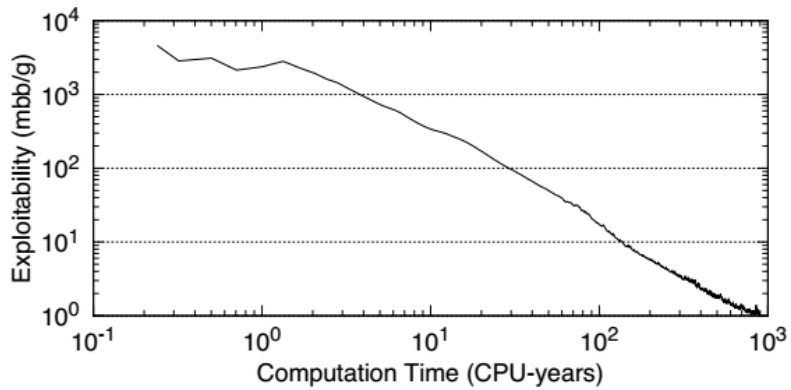


Figure 3: **Exploitability of the approximate solution with increasing computation.** The exploitability, measured in milli-big-blinds per game (mbb/g), is that of the current strategy measured after each iteration of CFR$^+$. After 1579 iterations or 900 core-years of computation, it reaches an exploitability of 0.986 mbb/g.

DeepStack and Heads-up No-Limit Texas Hold'em Poker

- No limit poker - any size of bet is allowed
- Much larger state space, about 10^{160} decision points
- Program DeepStack announced in March 2017
- Strong heuristic player, beat professionals in a set of matches
- Uses techniques much more similar to heuristic search than previous poker programs

DeepStack Approach

- Activity: watch Mike Bowling's talk:
  `https://www.youtube.com/watch?v=qndXrHcV1sM`
- Previous poker programs reasoned over the whole state space at once
- In complete information games (e.g. Go) you can limit your search to a state and its subtree
- The DeepStack team found a way to decompose the poker computation in a similar way
- "Until DeepStack, no theoretically sound application of heuristic search was known in imperfect information games."

Exploitability

Table 1. Exploitability bounds from Local Best Response. For all listed programs, the value reported is the largest estimated exploitability when applying LBR using a variety of different action sets. Table S2 gives a more complete presentation of these results (*10*).

Program	LBR (mbb/g)
Hyperborean (2014)	4675
Slumbot (2016)	4020
Act1 (2016)	3302
Always Fold	750
DeepStack	0*

*LBR was unable to identify a positive lower bound for DeepStack's exploitability.

Image source: DeepStack paper

DeepStack Approach

- Summarize history in two vectors, one for each player
  - Beliefs represented by “Range” of cards we could hold, plus “counterfactual values” for the opponent
- Depth-limited forward search using these vectors as state information
- Selective lookahead using only some of the possible bets
- At depth limit, use a deep neural network to evaluate states

DeepStack Approach (2)

- Continuous re-solving of subtree with different vectors - one traversal of search tree is not enough
- Standard fully connected network, trained on 10 million random poker games, starting from the end of game, with random belief vectors
- Output of network is vector of values, not just single evaluation

DeepStack Neural Net

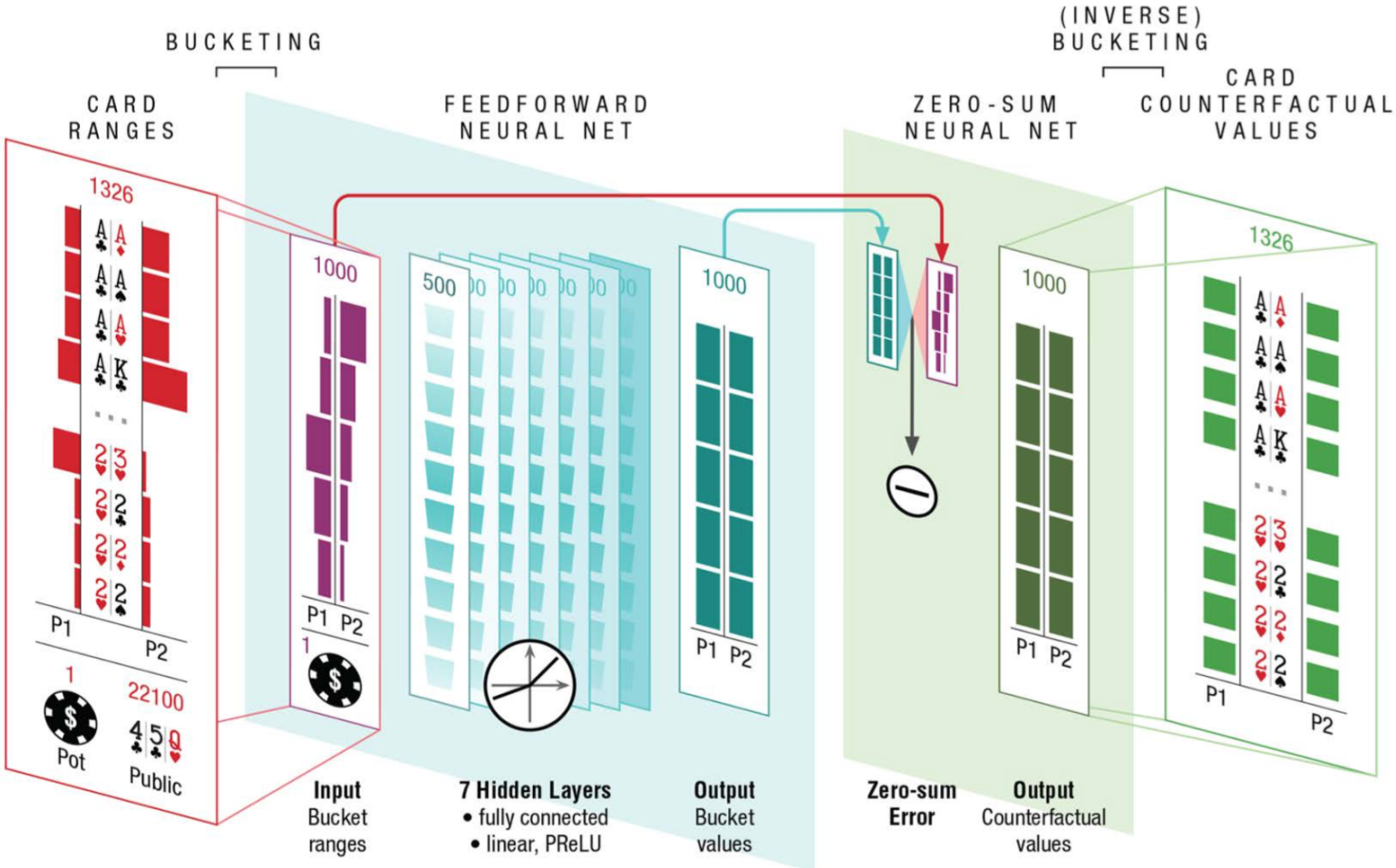


Fig. 3. Deep counterfactual value network. The inputs to the network are the pot size, public cards, and the player ranges, which are first processed into hand clusters. The output from the seven fully connected hidden layers is postprocessed to guarantee the values satisfy the zero-sum constraint, and then mapped back into a vector of counterfactual values.

Image source: DeepStack paper

DeepStack Approach (3) and Results

- Many other technical innovations to make heuristic search work in imperfect information games
- Amazing body of research, well worth deeper study
- Beat many human professionals over 3000 hand series

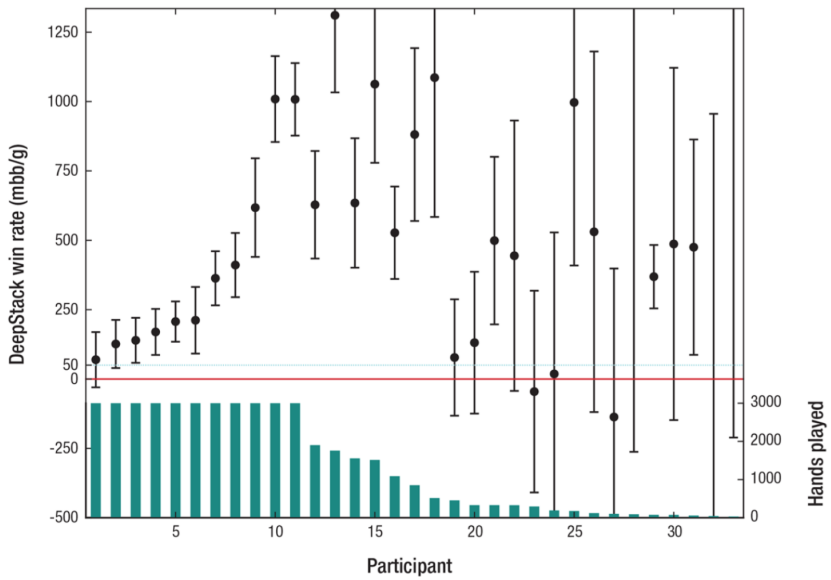


Fig. 4. Performance of professional poker players against DeepStack. Performance estimated with AIVAT along with a 95% confidence interval. The solid bars at the bottom show the number of games the participant completed.

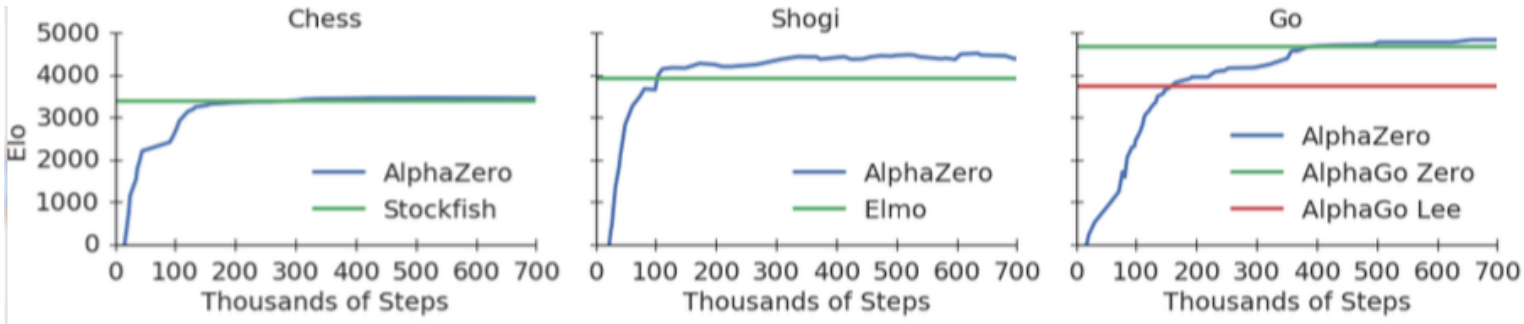
Alpha Zero: Chess and Shogi

- Paper: Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm
- Preprint on ArXiv, December 2017
- Final version in Science, December 2018
- Part of our readings
- Main ideas:
  - Generalize, simplify AlphaGo Zero approach
  - Apply to other games - chess and shogi (Japanese chess)

Alpha Zero Approach

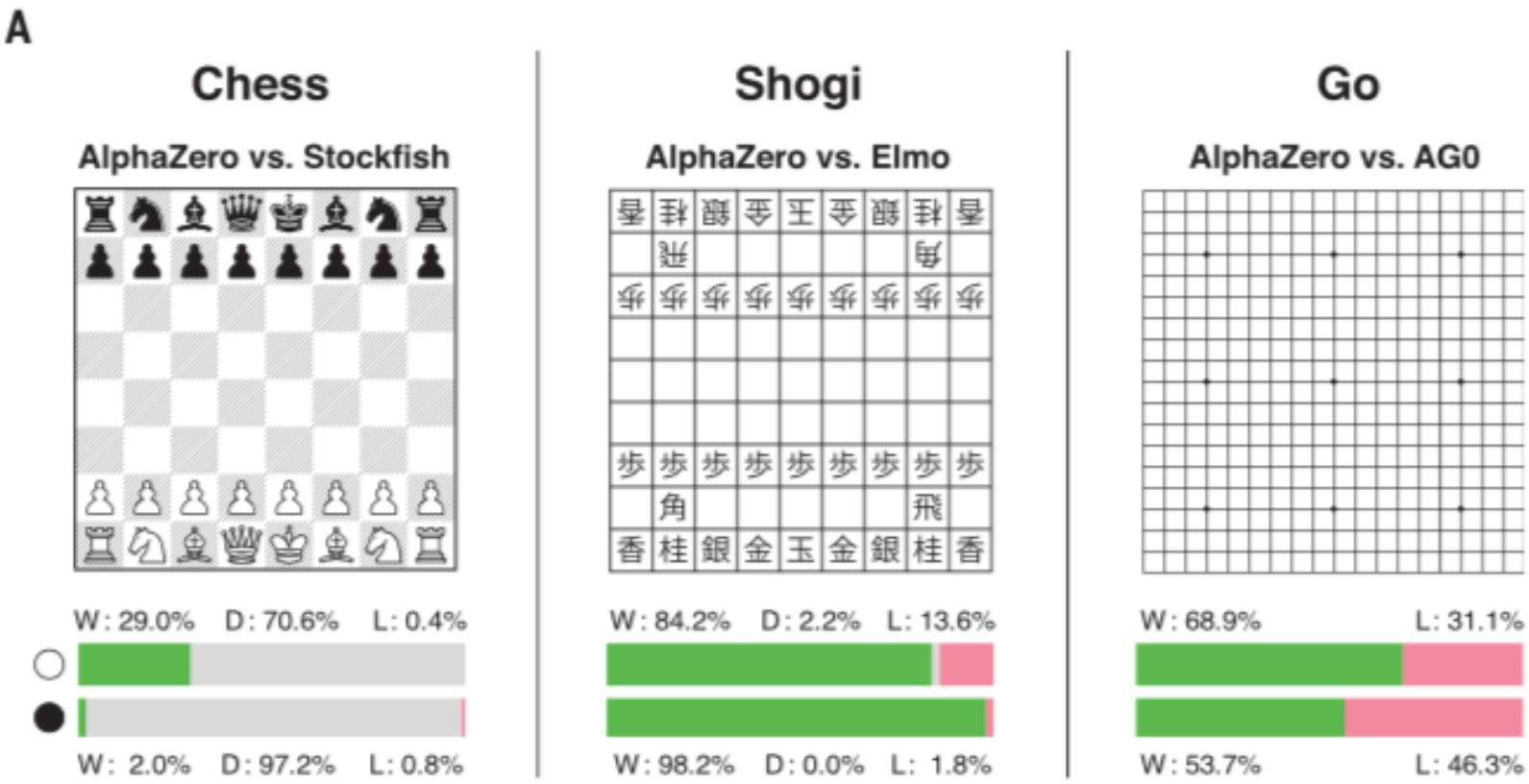
- Same as in Alpha**Go** Zero:
  - Two-head deep network, with policy and value heads
  - $(p, v) = f_\theta(s)$
  - MCTS for learning from self-play and for playing
- Different from Alpha**Go** Zero:
  - Learns expected outcome, not winning probability
    - Chess and shogi have draws, needs to learn to estimate those
  - AlphaGo Zero training and evaluation took advantage of board symmetries
    - AlphaZero does not
  - Learns by continuous updates to a single network
    - Alpha**Go** Zero learned its networks in generations
    - Each network used games from the previous best net
    - Alpha Zero learns and updates the same single net

Alpha Zero: Go, Chess and Shogi Learning



- Can learn Go, chess, shogi from scratch
- Beat top programs in matches
- In 2018 version, careful results against many versions of top other programs
- Alpha Zero wins even with large time handicap
- Hardware hard to compare - TPU vs parallel CPU

Alpha Zero: Go, Chess and Shogi Results Summary



Alpha Zero Summary and Discussion

- Very strong result
- Generalizes work on Go to other classical board games
- Stronger than other top chess and shogi programs
- Approach often copied by others
- Examples: five in a row (gomoku), connect 4, other games

Course Summary, Final Exam, Wrap-Up

- Three components of modern heuristic search: search, knowledge, simulations
- Fourth component - machine learning. Learn knowledge, simulation policy, in-tree guidance
- Recent strong focus on knowledge - deep learning, DCNN
- The combination of search and knowledge is much stronger than knowledge alone
  - E.g. first AlphaGo paper - Value net alone = 1600 Elo (weak amateur), Value net + policy net + simulations = 2800 Elo (strong professional) - 1200 Elo difference
  - AlphaGo Zero: 2000 Elo difference between full system and network-only
  - In AlphaGo Zero and Alpha Zero, search improves knowledge, and knowledge improves search - virtuous cycle

If This Course Was Twice as Long ... Some Pointers for Further Studies

- In Cmput 455 I have given only a high-level overview of modern heuristic search
- I needed to skip over many very interesting and important topics
- I will now present some of these with one or two slides per topic
- Single agent search, planning, MCTS improvements, tuning, parallel search, Atari games, ...

Single-Agent Heuristic Search

- We only reviewed the basics of “traditional” single-agent search
  - Search only, or “blind search”
  - Search plus knowledge
- There are many exciting new single-agent methods that combine all three pillars - search, knowledge, simulations
- There are also exciting applications of machine learning for single-agent search

Single-Agent Heuristic Search with Simulations

- Single-agent MCTS
  - Used for many optimization problems, even for drug discovery in Chemistry
- Nested Monte Carlo Search (Cazenave)
  - Use rollouts, plus recursion to define more powerful rollouts which themselves are Monte Carlo searches
- Nested Rollout Policy Adaptation (Rosin)
  - Online machine learning to improve the simulation policy
  - Holds current world records for several difficult puzzles

Machine Learning for Single-Agent Search

- Often, many different problem instances from same domain
  - Example: same airport, different airplanes landing and taking off at diffrent times
- General idea for many learning approaches:
  - Solve small problems by a (quick) search
  - Learn generally good actions or action sequences from the solutions for these small problems
  - Use the learned knowledge to solve similar, but larger problems

Search Improvements, and Other Applications of Modern Heuristic Search

- Better algorithms for MCTS and for solving games
- Other games
- Performance optimization

MCTS Enhancements

- Many improvements for MCTS have been developed
- Adaptive simulation policies
- More statistics over all simulation moves (RAVE), over single points on the board (territory map)
- Integrating an exact minimax solver within MCTS
- Adding 1 or 2-ply lookahead to simulations
- Many more...

Other Games where MCTS is Successful

- Hex - multiple World Champion MoHex developed in Prof. Hayward's group
- Amazons - most strong bots use MCTS
  - Interesting case - short 5 move simulations, then call an evaluation function works best
- General Game Playing - all strong programs use MCTS
- Havannah - best program by UofA graduate Timo Ewalds (now at DeepMind)
- Many more

Specialized Game Solvers

- There are algorithms that usually work better than alphabeta for solving games
- Example: Proof-number search, df-pn
- They try to find a small proof tree
- Sometimes, a narrow deep tree can be a smaller proof
- These algorithms are very good in finding such proofs

Tuning and Optimizing

- Tuning and optimization are important steps in building high performance heuristic search programs
  - Remember experiment: Fuego 500x faster than our Python demo Go programs
- Better algorithms
- Better caching and re-use of solved subproblems
- Low-level optimizations and tuning
- Running on parallel computers
- Offline pre-computations (e.g. opening books, endgame databases)

Atari Games and General Game-Playing

- Everything we did was for a single game - Go
- AlphaGo is very specialized: cannot play on a different board size or even with a different komi
- Can we solve more general tasks with heuristic search?
- Automated planning is one example - one planner, many different planning problems
- Atari games is a famous example for learning to play many different games
  - ALE - Arcade Learning environment - simulator for Atari 2600 games console, developed in our dept.
  - Deep learning can learn to play many games from raw pixels, surpass human skills
- Many other learning platforms for developing “general AI”, often using games

Implementation of Neural Networks

- Details, hands-on training and evaluation
- Algorithms for GPU and TPU
- Using software packages such as Tensorflow
- Creating, cleaning and preparing data for training and test
- Choices of architecture and input features

Parallel Heuristic Search

- Parallel MCTS on computer clusters
- How to scale to really big searches?
- Example: Fuego ran MCTS on 2000 cores on Hungabee system
- Example: AlphaGo Fan ran on 1,202 CPUs and 176 GPU, “similar hardware” plus TPU for AlphaGo Lee

Some Final Words of Wisdom...

- Many of you will graduate soon
- This is a time of unprecedented importance of AI, ML, heuristic search in real world applications
- 99.99% of users will not understand the potential and the limitations of the new technology. That is scary.
- Remember garbage in - garbage out principle
- Your responsibility is to use your understanding to be a leader in guiding applications of this technology

Finally...

- Thank you for taking 455 with me
- Thanks to the TAs: Abbas and Amir, for their work
- Please finish your quiz 13 and your USRI
- Good luck for all your final exams!