Here are some problems I've thought about in the past few years. You are also free to choose some other problem in (classical or combinatorial) game theory.

suggested reading

- Polya *How to solve it*
  `https://en.wikipedia.org/wiki/How_to_Solve_It`
- Ryan Hayward and Bjarne Toft *hex, the full story*
  `https://search.library.ualberta.ca/catalog/8574310`

1 geo-y

- geo-y environment for CMPUT 355
- solver strong enough for any base 3 position (done by Jacob Garber)
- solver strong enough for any base 4 position (almost done)
- solver strong enough for any base 5 position

2 logical staircase for learning hex (or other game)

do humans learn with discrete steps? if yes, what would those steps be when learning hex? perhaps first learn how to extend your groups, then learn how to look at 1 opponent response, then learn how to look at 4-ply sequences?

would this help solving? solve assuming weak counter-strategy, then stronger, then stronger, etc.

3 hex: our solver

- modify to work better with human guidance

- add button: automatically back up k moves, store results in TT at each step
- move ordering: beyond PNS (PNS + MCTS?)
- player+solver: integrate more solver direction (PNS?) into MCTS
- add button: 1-step k-opponent-move refutation

  e.g. allow white to play 3 moves in a row, make a winning black move, back up and put each white move in TT as loss
- canonical form for dead-capture-permanently-inferior fillin?

  this might make TT more effective
- allow decomposition into two (or more) subgames,

  solve each independently, and save the result
- threat search?

4 hex: best wins

- implement shortest-win solver
- confirm Weaver values for 6x6 7x7
- implement smallest-cell-set solver
- implement smallest-proof solver
- use any of these metrics with ML

5 hex: compact notation

- write a simple solver that finds solution in and/or notation
- convert from and/or notation to JingYang (Pankratz) notation

6 hex: solve 10x10, 11x11

- 10x10 Jakub found 2 non-iso winning openings (2.2, center)
- strategy-stealing proof: 2 non-iso losing openings (acute corner and its player-side neighbor)
- goal: find a verifiable strategy for some winning opening
- for center, I found — by hand plus solver plus mohex — a winstrat for 3 or 4 hardest openings, took about 120 hours

  main ideas:

  - induction/recursion?
  - Noshita connections (win or connect)?
  - Weaver swivel
  - subgame contracts (exploring bridge subgame contracts would be a nice project)
  - pseudo-solver/ML
  - when opponent probes side connection, sometimes you just repair the connection, but sometimes you want to swivel/flare
  - when oppo. blocks side weak connection in splitting manner, sometimes best move is tenuki, and play on other side of board where both possible ladders you could start will eventually end up

7 hex: human-concept strategies

- roughly, you can describe Weaver's 6x6 strategy (2nd player moves in paren) like this: centre (choke), push (block), flare (cut), jump (cut), ladder (ladder).
- each of these 8 terms is a different kind of move. Mohex does not recognize any of them. Mohex never plays the flare, but it will play all 7 others.

- maybe there's a way to train a new RAVE to see the importance of the flare. the point is that after the block, 1st player has 2 ladder options that require jumps, and if you look ahead, the 2 pvs from these 2 choices meet roughly where flare is played, which is why this gives a shortest win. the 2 ladder options also win, they just take more moves.

8 blend mohex-solver

- both programs are old and need rethinks, in particular, on hard problems, they both would benefit from the help of the other at the start.
- allow player to peek at solver hash table, see if any nodes are solved
- machine learning from pn/dn scores?
- add progressive bias from solver
- recognize ladder patterns, and do local explorations
- probable pruning: does pruning *probably inferior* cells help?
- provable pruning: does pruning inferior 432plus cells help?
- 1-sided miai probes

9 simple hex program

- suitable for CMPUT 355 or 497/670
- python? start with union-find move-to-front miai connectivity roll-outs, use all-shortest-paths to find live cells
- compare hex evaluation functions

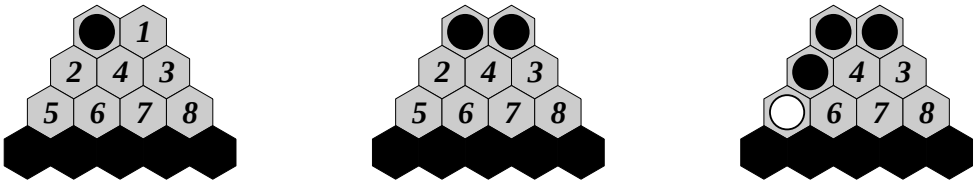

Figure 1: (from left) 432 conjecture: 4-8 inferior? 432+1: our inferior cell engine knows 4 inferior, but 6-7 also inferior, what about 5,8? 432+12(5): engine does not realize that 6,8 inferior

10 hex: better player

- build a stronger 11x11 player
- bootstrap from MoHex-3CNN? pick one opening, learn it well

11 analyze-bot

- automatic game analyzer: generate post-match game report
- for each move, run long-time search to get score
- report score drop/boost with each move
- from end, backup, solve etc
- blunders? weak moves?

12 hex: machine learning

- modify AlphaGo or AlphaZero or MuZero concept to work for Hex assuming that we also use solver
- implement some self-learning loop system
- repeat: select best player(s), improve them automatically (selfplay/generate, learn, solve)
- could be non-CNN approach
- modify KataGo so that it works for hex

13 hex/Y: virtual connections, beyond sets

- call 0-join vc, 1-join semi-c, k-join: after k moves, player has vc, store 2-joins where opponent has unique stopper?
- experiment with the smallest ones: eg, using 2-joins, prove weaver's 6x6 solution is valid (not nec shortest)
- e.g. 6x6 center opening, after white blocks 4.b6, black has 2-join c5,c6, so if black plays c5, white is forced to block c6. this starts a ladder ...
- investigate what connections this might find that current vc-builder does not find
- common miai substrategy: gather data
- explore? implement? is this related to lambda search? forcing sequences?
- embed in solver?
- fast playouts: does this help?
- generalize mustplay reasoning to allow for interfering substrategies. e.g. if a B-ladder running down W's right side runs into a B-stone that is part of a VC, and the other cells in that VC happen later in the ladder, then these connections have intersecting carriers but do not interfere. this requires replacing carrier sets with carrier ordered sets. find some small examples of this and see whether you can shorten any particular small-board win-proof

14 hex: finding traps

- how to find openings that are hard for the opponent?
- book building: how to ensure player can continue from book entries
- cache book
- puzzle generation (unique winning moves)

15 hex: subgame strategy contracts

- these could be connection contracts
- reasoning about contracts in a solver, e.g. learn which contracts are likely to be used, solve under those constraints, then if necessary solve without the constraints: is this faster?
- player? solver? rollouts?

16 hex: visualization

- our hex gui needs a score history visualizer, plotting estimated score after each move so far. our searches also need visualizers, it would be good to see scores at various depths of the pvs during search.
- our connection vis? other vis?
- a step-thru slo-mo vc/search vis tool

17 hexgui

- help button ? documentation?

18 hex: classic olympiad games

- top 10 ICGA hex games: what are they? why?
- can we use tools to estimate playing strength of past programs? or analyze past programs? eg use our strongest program to analyze past programs, and our solver, plot number of moves until win detected with uniform (300s?) search

19 hex: tutor

- build a hex tutor

- much of this material has been written for an informed specialist audience, but there is little that is precise and suitable for a general audience

20 hex: reversing

- explore reversible moves in hex
- when is it a good idea to reverse a reversible move? (not usuallly, but maybe when opponent makes a reversible move we should spend time looking for a most-punishing move)

21 432 and 432plus

- read Fabiano's paper: are there any interesting open problems?
- other problems probably more critical now
- 432 conjecture still open
- some resulting patterns, eg 432plus, have provably inferior cells that we are not pruning. investigate, and see whether provable pruning helps.
- investigate probable pruning
- write down what we can prove, gather data on what we can't
- problem seems to be that our engine does not recognize played cells that are killed; after move A, if reply B kills A, then A is inferior to B, our algorithms recognize this when B is simple kill, but not for some capture-kills

22 dark hex

- solve 3x4
- get bounds for larger boards

- talk to Dustin Morrill

23 other games

- write/improve player/solver for your favorite game
- Y
- irregular Y
- irregular hex, Vex
- any game from Browne's book
- 7x7 Rex? Rex player? Rex ladders?
- inf cell analysis for other games

24 solve small game, eg. 6x6 go

- someone is currently working on 6x6 go
- start with 4x4 etc
- data generation?
- leela0 approach

25 search algorithms

- mcts
  - replace UCB with Thompson sampling
  - shared-tree player-solver
- pns
- sibling conspiracy number search

26 random-v-perfect

- for a given position, compute the winning probability when a perfect player (or well-defined deterministic player) plays a uniform-random player? this gives a measure of strength of move: for a given position, the one that does best against a random player is arguably strongest

27 BoxOff

- NP-complete with ≥ 5 colors
- what if ≤ 2 colors?
- what if ≤ 2 rows?
- tutor? analyzer?
- Steven Meyers wants to know: 5 colors, 15x12 board, give bounds on probability that a random board (with 18 pairs of each of the 5 colors) is solvable (what can you do with fewer colors, or same number of colors but smaller board?)