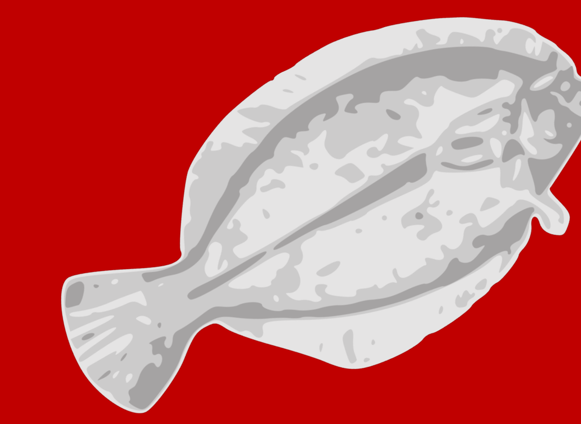
Flounder: an RL Chess Agent

Travis Geis, Andrew Bartolo, Varun Vijay
Department of Computer Science, Stanford University



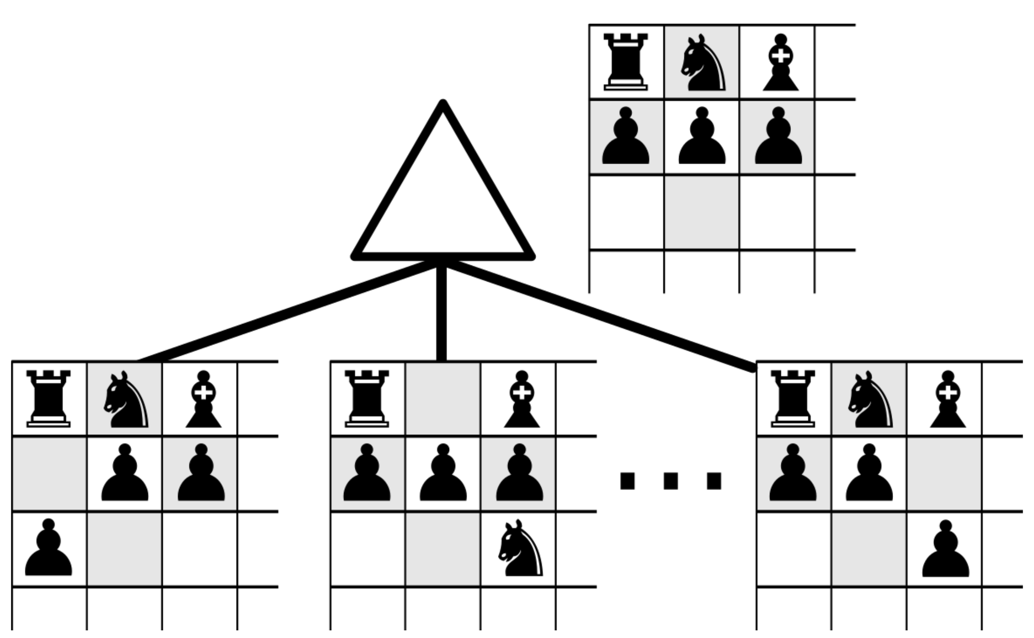
Motivation

Chess agents provide a good testbed for developing search algorithms and evaluation functions.
We built a chess engine that uses **MTD(bi) search** and a **trained evaluation function** to play chess more effectively.

Challenges

Chess's high branching factor (~35 moves per position):

- Some form of minimax tree pruning is needed
- Most states will never be explored, so we must approximate their value
- Endgame states may produce too-sparse features
- TD-learning needs large histories of data to learn good weights



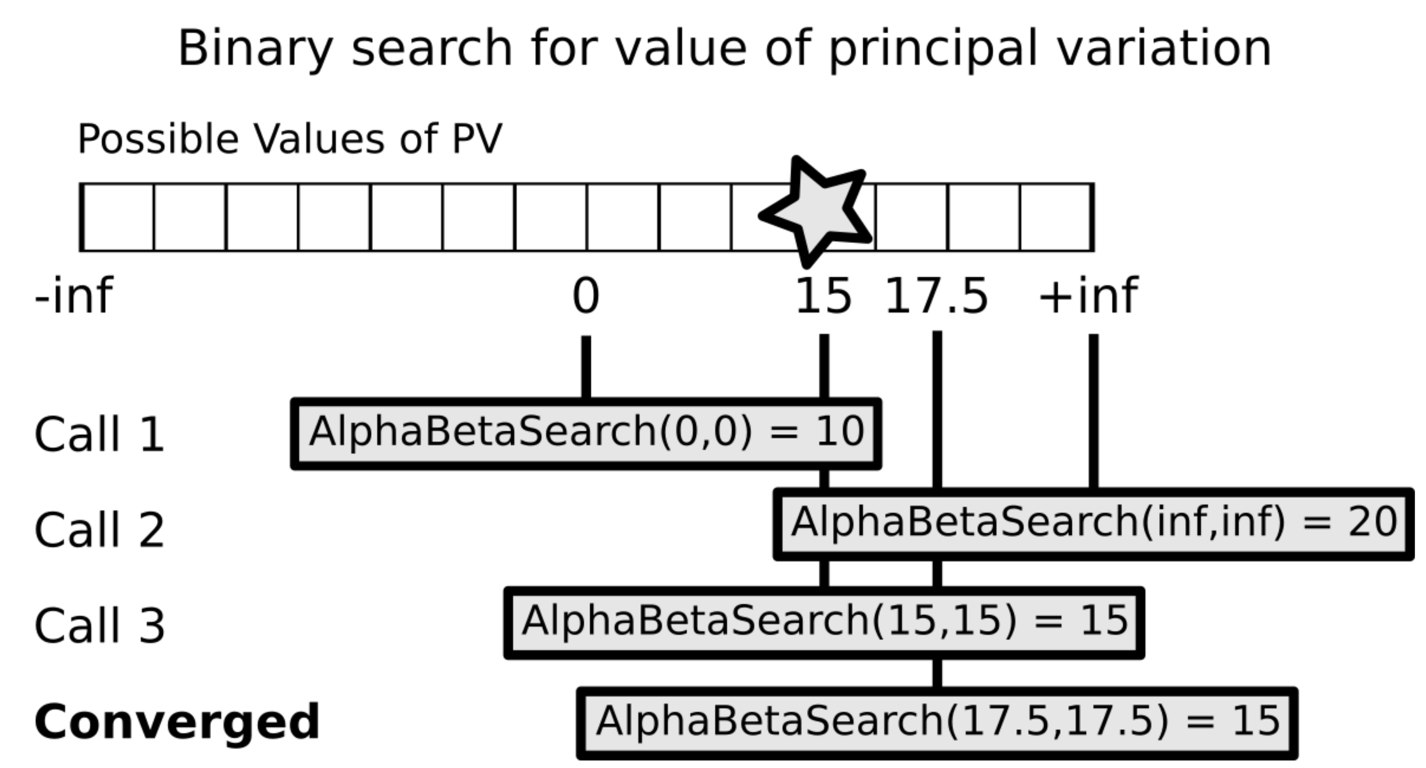
Maximizing over possible next moves.

Making minimax search faster

•Alpha-beta pruning: don't continue searching tree nodes when we have a higher-valued move
•LRU cache: keep commonly-searched states in a table
•Killer heuristic: when performing iterative deepening, use depth *d*'s best move as the starting point for depth *d+1*
•MTD(bi) search, modified to use real-valued evaluation function (which we learn)

Approaches

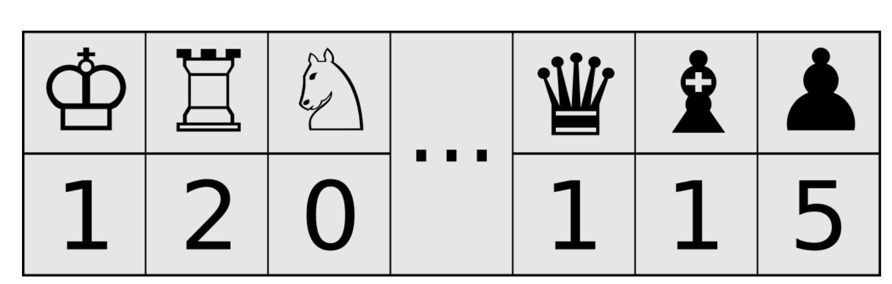
MTD(bi) search



- Uses zero-window alpha-beta search
- Tightens lower and upper bounds with successive calls to AlphaBetaSearch(α,β)
- Halves step size on each successive call (b-search)

State evaluation

Use a hand-crafted feature extractor, with features:
•White or Black move
•Material count – numbers of K, Q, R, B, N, P present for each player
•Piece "slots" – for each of the 32 individual pieces, store presence indicator, x- and y-coordinates, and least-valuable attacker (LVA)
•Initialize weights to predict game outcomes from board positions using SGD



The material count feature.

Results

- We benchmarked our oracle, baseline, and own engine (using table eval. fn.) against Strategic Test Suites:

Test Suite Name	Stockfish (500ms) (Oracle)	Sunfish (500ms) (Baseline)	Flounder
STS1 (Undermining)	89	13	2
STS2 (Open files and diagonals)	81	16	4
STS3 (Knight outposts)	77	23	4
STS4 (Square vacancy)	77	6	7
STS5 (Bishop vs. knight)	81	34	21
STS6 (Re-capturing)	78	21	19
STS7 (Offer of simplification)	75	9	8
STS8 (Advancement of f/g/h pawns)	77	9	1
STS9 (Advancement of a/b/c pawns)	75	7	4
STS10 (Simplification)	82	42	19
STS11 (Activity of the king)	74	7	2
STS12 (Center control)	79	13	1
STS13 (Pawn play in the center)	80	15	4

- Initialize with weights pretrained by SGD
- Train by playing 108 batches of 10 games against Stockfish

Choice of next move	Number of games played	Mean num. moves before loss by checkmate
TD-Learning	30	37.2
Random-valid-move	30	25.2

Future directions

- Augment feature extractor to better handle endgames
- Move away from industry-leading chess engines as benchmarks of success

References

•Ahle, Thomas. *Sunfish.* Computer software. *Sunfish: a Python Chess Engine in 111 lines of code.* 31 Aug 2016. Web. 4 Dec. 2016. <https://github.com/thomasahle/sunfish>.
•Bernhardsson, Erik. "Deep learning for… chess." 28 Nov 2014. Web. 4 Dec. 2016 <https://erikbern.com/2014/11/29/deep-learning-for-chess/>.
•Corbit, Dann and Swaminathan. *Strategic Test Suites.* 13 Jun. 2010. Web. 4 Dec. 2016 <https://sites.google.com/site/strategictestsuite/>
•Fiekas, Niklas. *Python-chess*. Computer software. *Python-chess*. Vers. 0.15.4. N.p., 2 Nov. 2016. Web. 4 Dec. 2016. <http://github.com/niklasf/python-chess>.
•Lai, Matthew. "Giraffe: Using Deep Reinforcement Learning to Play Chess." Imperial College London, Department of Computing, Sep 2015. Web. 4 Dec. 2016 <https://arxiv.org/abs/1509.01549>.
•Plaat, Aske. *Best-First Fixed-Depth Minimax Algorithms.* Erasmus University. 14 Dec. 1995. Web. 4 Dec. 2016. <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.93.8838>.
•Plaat, Aske. "MTD(f): A Minimax Algorithm faster than NegaScout." 3 Dec. 1997. Web. 4 Dec. 2016. <https://people.csail.mit.edu/plaat/mtdf.html>.
•Romstad, Tord, Marco Costalba, and Joona Kiiski. *Stockfish*. Computer Software. *Stockfish Chess.* Vers. 8. N.p., 1 Nov. 2016. Web. 4 Dec. 2016. <https://stockfishchess.org/>.
•Weill, Jean-Christophe. "Experiments With The NegaC* Search - An Alternative for Othello Endgame Search." Université Paris 8, Vincennes, Département d'informatique, Insititut d'Intelligence Artificielle. 1991. 4 Dec. 2016. <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.39.3189>