Programowanie silników szachowych – część 2 (przegląd silników i technik ich implementacji)

1. Traditional chess engines

"Significant among the early ideas in computer chess is Claude Shannon's 1949-50 distinction between a brute force (type-A) strategy for looking at every combination of moves, and the use of chess knowledge to select and examine only a subset of the available moves (type-B strategy). Although some electro-mechanical systems to play a subset of chess had been built prior to Shannon's work, it was the programming of his ideas that led to the development of today's computer chess machines." …

A three-stage tree model is popular with computer chess programmers. The first stage uses a brute force (Shannon type-A) approach, the second a selective (type-B) search, and the third a strategy known as a quiescence search, designed to resolve the problems and conflicts that remain.

(from https://webdocs.cs.ualberta.ca/~tony/ICCA/anatomy.html)

Kasparov vs. Deep Blue (https://pl.wikipedia.org/wiki/Deep_Blue) – mecz w 1997 r. zakończył się pierwszym w historii zwycięstwem maszyny w meczu z mistrzem świata

Stockfish: https://stockfishchess.org/, https://github.com/official-stockfish/Stockfish, https://www.chessprogramming.org/Stockfish - aktualnie jeden z najsilniejszych klasycznych silników szachowych (GPL)

Houdini: http://www.cruxis.com/chess/houdini.htm (darmowa wersja Houdini 1.5a) – w ścisłej czołówce

Komodo https://komodochess.com/store/pages.php?cmsid=14 (darmowa wersja Komodo 9.02) – również wśród najlepszych silników szachowych

TCEC (Top Chess Engine Competition) Chess Championship: https://en.wikipedia.org/wiki/Top_Chess_Engine_Championship - cyklicznie rozgrywane nieoficjalne mistrzostwa świata silników szachowych

Przeszukiwanie drzewa gry – wg strategii *depth-first search*:

- **tablice transpozycji** (https://en.wikipedia.org/wiki/Transposition_table) – przechowują rozpatrywane dotychczas pozycje (wraz z ewaluacjami) – implementowane jako **tablice haszujące (hash tables)**; ograniczony rozmiar (cache) – usuwane są rzadziej używane pozycje; zysk – uniknięcie oceny poddrzewa drzewa gry; bardziej wartościowe są pozycje na mniejszej głębokości (większe poddrzewo); po n posunięciach – maks. $(n!)^2$ transpozycji

- **minimax** algorithm (https://en.wikipedia.org/wiki/Minimax) – generuje pełne drzewo gry do zadanej głębokości

- negamax algorithm (https://en.wikipedia.org/wiki/Negamax) – równoważny, ale łatwiejsza implementacja

- minimax/negamax with alpha–beta pruning (https://en.wikipedia.org/wiki/Alpha-beta_pruning) – zwraca ten sam ruch co minima/negamax, ale nie rozpatruje gałęzi, które nie mogą poprawić oceny pozycji (algorytm typu B&B); algorytm używany np. przez Stockfish'a

- problem horyzontu przeszukiwania, podejście **quiescence search w celu oceny pozycji końcowej** (np. sprawdzenie bić i promocji; https://en.wikipedia.org/wiki/Quiescence_search, https://www.chessprogramming.org/Quiescence_Search)

- iterative deepening – strategia wykorzystywana w silnikach aby w każdej chwili móc zwrócić ruch (gra na czas); kolejne tury przeszukiwania drzewa gry, na coraz większej głębokości; ponieważ efektywność przeszukiwania alfa-beta zależy od kolejności rozważania posunięć, stosuje się **najpierw płytsze przeszukiwanie w celu ustalenia kolejności rozpatrywania posunięć przy głębszym przeszukiwaniu**.

- move reordering (https://webdocs.cs.ualberta.ca/~tony/ICCA/anatomy.html) - ustalanie **kolejności rozpatrywania ruchów** w sposób pozwalający szybciej redukować drzewo gry

- null move pruning (https://www.chessprogramming.org/Null_Move_Pruning) – heurystyka przyspieszająca przeszukiwanie drzewa gry – "assumes a pass move should be worse than any variation, in positions that are unlikely to be in zugzwang, as determined by simple heuristics"

Opening books – książki otwarć – używane do wyboru posunięć w początkowej fazie partii

Endgame tablebases (https://en.wikipedia.org/wiki/Endgame_tablebase) – determinują wynik partii dla wszystkich możliwych pozycji o zadanej liczbie figur (maks. 7), pokazując jednocześnie najlepszą grę obu stron:

- **Nalimov tablebases** (https://www.chessprogramming.org/Nalimov_Tablebases; również Readme.md w stockfish-10-src.zip) – „rozwiązane" wszystkie pozycje do 6 figur włącznie, włączając króle (pozycje z 6 figurami wyliczone w 2005 r.)
- **Lomonosov tablebases (**http://chessok.com/?page_id=27966) - 7 figur (bez 6 figur na samotnego króla), rozmiar 140 TB (pozycje wyliczone na superkomputerze Lomonosov w Moskwie w 2012 r.)
- **Syzygy tablebases** (https://www.chessprogramming.org/Syzygy_Bases; również Readme.md w stockfish-10-src.zip) – „rozwiązane" wszystkie pozycje do 6 figur włącznie (2013 r.) a następnie do 7 figur włącznie (2018 r.).
  „The Syzygy tablebases are "Distance to Zero" (DTZ) tablebases. This means that they report, in addition to a definitive win/draw/loss (WDL) score, the distance to the zeroing of the 50-move draw clock. Unlike other endgame tablebases which report Distance to Mate (DTM), the Syzygy bases won't always report the fastest win from a particular endgame position, but

you can be confident that the WDL score (and moves) provided by a tablebase probe are accurate."

2. AI (trained) chess engines

Alpha Zero (https://en.wikipedia.org/wiki/AlphaZero) – premiera w 2017 r.; niespotykana u programów szachowych umiejętność grania na długofalową rekompensatę za materiał oraz atakujący styl gry; możliwości – np. https://youtu.be/akgalUq5vew - AlphaZero vs Stockfish Chess Match: Game 3 – (zugzwang!); używa do oceny pozycji funkcji nieliniowej w postaci **deep learning neural network** (https://www.chessprogramming.org/Deep_Learning); uczenie ze wzmocnieniem (*tabula rasa reinforcement learning*) oraz Monte Carlo Tree Search (MCTS); działa na wielu dedykowanych **TPU** (https://en.wikipedia.org/wiki/Tensor_processing_unit)

- https://en.wikipedia.org/wiki/DeepMind
- https://en.wikipedia.org/wiki/AlphaZero
- https://en.wikipedia.org/wiki/Stockfish_(chess)#Stockfish_versus_AlphaZero

AlphaZero was trained solely via "self-play" using 5,000 first-generation TPUs to generate the games and 64 second-generation TPUs to train the neural networks, all in parallel, with no access to opening books or endgame tables. In parallel, the in-training AlphaZero was periodically matched against its benchmark (Stockfish, elmo, or AlphaGo Zero) in brief one-second-per-move games to determine how well the training was progressing. DeepMind judged that AlphaZero's performance exceeded the benchmark after around four hours of training for Stockfish, two hours for elmo, and eight hours for AlphaGo Zero. The trained algorithm played on a single machine with four TPUs and a 44-core CPU. (Wiki)

Leela Chess Zero (https://en.wikipedia.org/wiki/Leela_Chess_Zero, http://lczero.org, https://github.com/LeelaChessZero/lc0, https://github.com/LeelaChessZero/lc0/wiki/Technical-Explanation-of-Leela-Chess-Zero) – premiera w 2018 r. – open source chess engine; distributed computing project; najlepiej działa na GPU Nvidii (platforma CUDA)

LCZ, AZ – ewaluacja mniejszej liczby pozycji niż Stockfish (o kilka rzędów wielkości), Monte Carlo tree search (MCTS), głęboka sieć neuronowa (konwolucyjna), trening poprzez granie samemu ze sobą, bez książki otwarć czy baz końcówek (tylko na podstawie reguł gry)

LCZ blog, 2018-09-21 (http://blog.lczero.org/2018/09/guide-setting-up-leela-on-chess-gui.html):

„Leela Chess Zero is a project started before some months inspired by Deepmind's papers about AlphaGO Zero and AlphaZero, which is based on a **new paradigm of Chess engines by not using traditional AlphaBeta search with handcrafted evaluation function** but uses a variant of MCTS

search called PUCT (Predictor + Upper Confidence Bound tree search) and for evaluation function it uses a self-taught neural network that learns by deep learning methods by playing against itself million times."

Monte Carlo Tree Search (https://en.wikipedia.org/wiki/Monte_Carlo_tree_search; https://www.chessprogramming.org/Monte-Carlo_Tree_Search) –> *dodatkowe materiały*

Wiedza dziedzinowa (trudna do zakodowania w funkcji oceniającej; wygląd na to, że wyuczona przez Alpha Zero), np.:

- długofalowa aktywność

- identyfikacja kluczowych cech pozycji

- długoterminowa strategia gry

Dodatkowe referencje:

https://arxiv.org/abs/1712.01815 - **źródłowy artykuł (preprint) opisujący Alpha Zero**: „Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm", David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, Demis Hassabis, arXiv:1712.01815v1, **Submitted on 5 Dec 2017**

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, Demis Hassabis (**2018**). *A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play*. Science, Vol. 362, No. 6419 – **final peer-reviewed paper** (lepszy opis AlphaZero, nowsze wyniki) – [download Open Access version]

https://deepmind.com/blog/article/alphazero-shedding-new-light-grand-games-chess-shogi-and-go

https://www.chess.com/article/view/whats-inside-alphazeros-brain

https://en.chessbase.com/post/leela-chess-zero-alphazero-for-the-pc

Wcześniejsza praca nt. GO:

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. Nature, 550:354–359, 2017.