Vyhľadávanie najlepšieho ťahu v šachu*

Martin Kubiš
Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
xkubis@stuba.sk

30. september 2023

Abstrakt

Šach je strategická stolová hra, ktorá sa skladá z šachovnice o rozmeroch 8x8 a 6 typov figúriek vo dvoch farbách. Aj keď sa na prvý pohľad zdá ako jednoduchá hra, neprestáva trápiť mozgy hráčov jak začiatočníkov, tak profesionálov. Od 50. rokov 20 storočia, kedy sa vyvinul prvý šachový robot, sa vývoj nezastavil a počítače už sú na míle ďaleko pred ľudskými hráčmi. Cielom článku je priblíženie čitateľovi metódy a techniky šachových algoritmov, ich optimalizácia, takisto aj porovnanie medzi klasickými šachovými algoritmami a algoritmami obohatené o umelú inteligenciu a ako vyhľadávajú najefektívnejší možný ťah. ...

1 Úvod

Šach je jednou z najstarších a najpopulárnejších hier, ktorá sa v priebehu storočí stala symbolom intelektu, stratégie a výzvy pre ľudský mozog.

Hranie šachu si vyžaduje veľa zručností, strategické myslenie, plánovanie určitých ťahov, schopnosť predvídať a dokonca byť nepredvídateľný. Aby sa hráči zlepšili, či už na profesionálnej alebo amatérskej úrovni, musia rozvíjať svoje zručnosti a znalosti o šachových algoritmoch a stratégiach, ktoré sú kľúčom k úspechu v tejto komplexnej hre.

Tento článok sa zameria na šachové algoritmy, aké sú stratégie určené na zlepšenie schopností hráča, pochopenie hry a vyhladanie čo najlepších možných žahov na prekonanie súpera. Šachové algoritmy sú rozhodujúce pre šachové programy, ktoré sú čoraz výkonnejšie a každým dňom umožňujú posúvať hranice toho, čo sme si mysleli, že je možné v oblasti rýchlosti a presnosti ťahov.

V tomto článku preskúmame históriu šachových algoritmov, pričom sa zameriame na najdôležitejšie algoritmy ako sú minimax algoritmus s alfa-beta pruningom, monte carlo tree search algoritmus, alebo využitie deep learningu, neurálnych sietí a umelej inteligencie a poukážeme na využitie týchto algoritmov v modernom šachovom svete. Ďalej sa budeme venovať technickým výzvam spojeným s vývojom šachových programov a ich schopnosťou hrať na úrovni šachových majstrov.

*Semestrálny projekt v predmete Metódy inžinierskej práce, ak. rok 2023/24, vedenie: Vladimír Mlynarovič

Nakoniec sa dotkneme etických a filozofických otázok spojených s používaním šachových algoritmov a umelej inteligencie vo svete šachu.

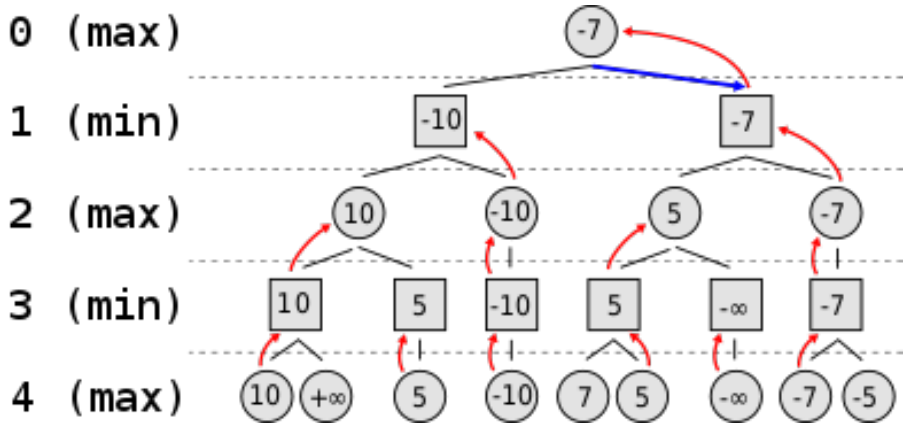
2 Minimax algoritmus

Minimax algoritmus je rozhodovací algoritmus používaný v hrách na určenie najlepšieho možného ťahu pre hráča v hre pre dvoch hráčov s nulovým súčtom, ako je napríklad šach alebo piškôrky. V hre s nulovým súčtom sú zisky jedného hráča presne vyvážené stratami druhého hráča, takže súčet ziskov a strát hráčov je vždy nula.

2.1 Princíp fungovania Minimax algoritmu

Minimax algoritmus funguje tak, že predpokladá, že obaja hráči budú hrať optimálne, a potom určí najlepší ťah pre hráča, ktorý práve robí ťah.

Robí to tak, že zvažuje všetky možné ťahy, ktoré by hráč mohol urobiť, a potom hodnotí každý z týchto ťahov simulovaním zvyšku hry za predpokladu, že súper urobí svoj najlepší možný ťah. Algoritmus potom vyberie ťah, ktorý vedie k najlepšiemu výsledku pre hráča, za predpokladu, že súper ako odpoveď urobí svoj najlepší možný ťah.



Obr. 1: obr1: Jednoduchá ukážka minimax algoritmu

2.2 Výhody a obmedzenia Minimax algoritmu

Minimax algoritmus je výkonný a teoretický prístup k rozhodovaniu o hrách, ktorý poskytuje optimálne riešenia v zmysle teórie hier. Tento algoritmus má však niekoľko obmedzení. Prvým je jeho výpočtová náročnosť, najmä v hrách s veľkým rozhodovacím priestorom. Preto boli vyvinuté rôzne varianty a optimalizácie, ako napríklad:

- Alpha-Beta pruning
- Otváracie ťahy (Openings)

Vďaka týmto optimalizáciam sa zníži počet vetví, cez ktoré musí algoritmus prejsť, taktiež sa vylepší celkový výkon algoritmu.

Môže sa zdať, že problém vlastne nejestvuje [1], ale bolo dokázané, že to tak nie je [2, 3]. Napriek tomu, aj dnes na webe narazíme na všelijaké pochybné názory [4]. Dôležité veci možno *zdôrazniť kurzívou*.

3 Alpha-Beta pruning

V praxi väčšina AI šachových robotov používa optimalizovanú verziu minimax algoritmu známeho ako alfa-beta pruning algoritmus. Základnou myšlienkou alfa-beta pruningu je zníženie počtu vetiev, ktoré je potrebné hodnotiť „prerezávaním“(pruningom) konárov, ktoré istotne nepovedú k lepšiemu výsledku ako tie, ktoré už boli hodnotené.

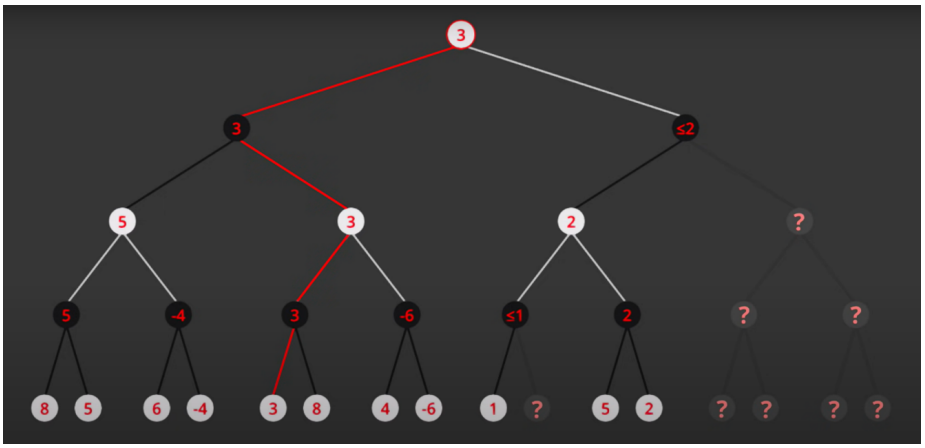
To sa dosiahne použitím dvoch hodnôt, alfa a beta, na sledovanie najlepšieho výsledku, ktorý bol doteraz zistený pre maximalizujúceho hráča (alfa) a najlepšieho výsledku, ktorý bol doteraz nájdený pre minimalizujúceho hráča (beta)

3.1 Princíp fungovania alpha-beta pruningu

Algoritmus funguje tak, že rekurzívne skúma strom hry a vyhodnocuje každý možný pohyb a výsledok pomocou algoritmu minimax. Pri skúmaní každej vetvy stromu aktualizuje hodnoty alfa a beta na základe najlepšieho výsledku, ktorý bol doteraz nájdený.

Ak nájde vetvu, v ktorej je výsledok pre minimalizujúceho hráča (beta) horší ako aktuálny najlepší výsledok pre maximalizujúceho hráča (alfa), prereže túto vetvu a prestane ju vyhodnocovať, pretože minimalizujúci hráč by tento ťah nikdy nezvolil.

Algoritmus postupuje rekurzívne nahor po strome, pričom neustále aktualizuje hodnoty alfa a beta, kým nevyhodnotí všetky možné pohyby a výsledky, alebo kým neoreže všetky vetvy, ktoré sú zaručene horšie ako aktuálny najlepší výsledok.



Obr. 2: obr2: ukážka alpha-beta pruningu

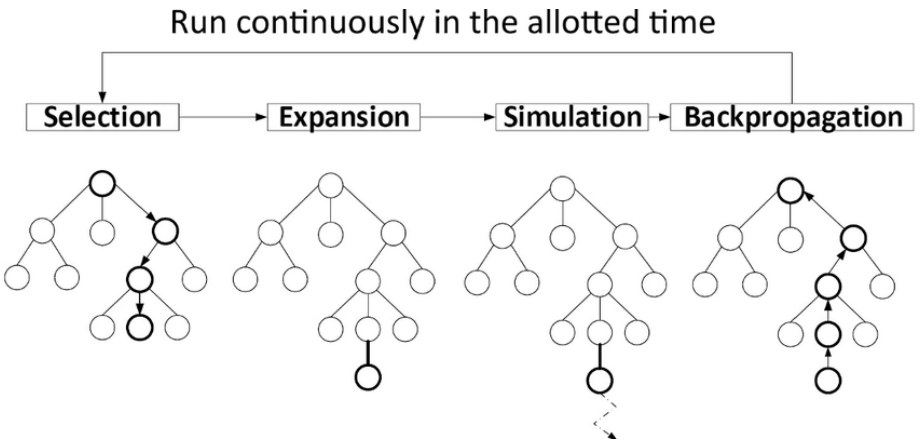
4 Monte Carlo Tree Search

Monte Carlo Tree Search (MCTS) je inovatívny a výkonný algoritmus v oblasti umelej inteligencie a teórie hier, ktorý bol využitý na rôzne stolové hry, ako je napríklad Go a šach. Tento algoritmus výrazne prekonáva tradičné techniky vyhľadávania v stromovom priestore, čím umožňuje získať vysokokvalitné výsledky v hrách s veľkými rozhodovacími priestormi.

5 Princíp fungovania MCTS

MCTS sa líši od predchádzajúcich algoritmov Minimax tým, že neprehľadáva celý stromový priestor. Namiesto toho sa používa metóda vzorkovania, pri ktorej sa ťahy vyberajú a simulujú až do konca hry, aby sa získal odhad aktuálneho stavu hry. Opakuje tento proces niekoľkokrát a na základe získaných hodnôt sa rozhodne, ktorý ťah je najlepší. V MCTS existujú štyri základné fázy:

1. výber (Selection)
2. šírenie (Expansion)
3. simulácia (Simulation)
4. spätné šírenie (Backpropagation)



Obr. 3: obr3: ukážka fungovania MCTS

Veľmi dôležitá poznámka. Niekedy je potrebné nadpisom označiť odsek. Text pokračuje hneď za nadpisom.

6 DeepLearning, Neural networks

DeepLearning a neurálne siete spôsobili revolúciu v spôsobe, akým počítače chápu a hrajú šach. AlphaZero, šachový program od DeepMind, využíva hlbokú neurónovú sieť na samoučenie. Program dokázal poraziť špičkových ľudských hráčov a dosiahnuť úroveň podobnú šachovým lídrom.

6.1 Role neurálnyh sietí v šachových algoritmoch

1. Vyhodnotenie pozície

   Neurónové siete možno trénovať na vyhodnotenie aktuálnej pozície v šachu. Toto hodnotenie možno použiť na výber najlepšieho ťahu a zlepšenie kvality rozhodovania na počítači.

2. Generovanie ťahov

   Neurónové siete možno použiť na generovanie a filtrovanie možných ťahov v hre. To urýchľuje výpočet a obmedzuje počet hľadaných mutácií.

3. Samo-učenie

   Neurónové siete môžu byť použité v počítačových šachových samoučiacich programoch. Napríklad AlphaZero nikdy pred tréningom nehral šach. Sieť sa poučila z každej hry a stala sa silnejším hráčom.

7 Záver

Literatúra

[1] James O. Coplien. *Multi-Paradigm Design for C++*. Addison-Wesley, 1999.

[2] Krzysztof Czarnecki, Simon Helsen, and Ulrich Eisenecker. Staged configuration through specialization and multi-level configuration of feature models. *Software Process: Improvement and Practice*, 10:143–169, April/June 2005.

[3] Krzysztof Czarnecki and Chang Hwan Peter Kim. Cardinality-based feature modeling and constraints: A progress report. In *International Workshop on Software Factories, OOPSLA 2005*, San Diego, USA, October 2005.

[4] Carnegie Mellon University Software Engineering Institute. A framework for software product line practice—version 5.0. `http://www.sei.cmu.edu/productlines/frame_report/`.