Modelovanie algoritmu pre šachový engine*

Martin Čajka
Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
xcajkam@stuba.sk

20. október 2021

Abstrakt

Cieľom článku je popísať a objasniť štruktúru algoritmu šachového enginu, vysvetliť myšlienkový postup za jej jednotlivými časťami a zanalyzovať univerzálne používané prístupy tvorenia algoritmu pri takej komplexnej hre, akou je šach. Článok sa taktiež zameriava na riešenie problémov a otázok s ktorými sa osoby podieľajúce na vývoji algoritmu stretnú a popisuje metódy ich riešenia, ako napríklad využívanie prvkov umelej inteligencie alebo ponáranie sa do princípov strojového učenia. K záveru sa článok vyjadrí k fungovaniu určitých algoritmov, hardvéru a softvéru potrebnému k ich chodu ako aj potencionálnej budúcnosti šachových programov (*subject to change).

1 Úvod

„Šach je všetkým: umením, vedou aj športom“ – vyhlásil ruský šachový veľmajster Anatolij Karpov. Za posledné dve dekády sa vďaka rapídnemu pokroku v oblastí umelej inteligencie tento jeho výrok aspoň z jednej časti potvrdil – šachu vládnu stroje, ktoré dominujú ľudských protivníkov na základe ich bleskových kalkulácii a schopnosti naozaj sa poučiť z vlastných chýb. Je fascinujúce ako rýchlo prebehla evolúcia v oblasti šachových strojov. Od čias šachových majstrov skrývajúcich sa v "Mechanickom Turkovi"to už je síce 250 rokov, prvé softvéry schopné naozaj hrať šach sa začali vyvíjať pred približne 50 rokmi.

2 Základy a materiálová hodnota

Aj napriek tomu, že šach má jasne stanované pravidlá a určené limity pre pohyb figúrok, stále tu je veľmi vysoká variabilita - každý jeden ťah má značný dopad na štádium hry. Existujú však isté faktory, ktoré napovedajú ako je na tom hra a ktorý hráč ma najlepšie predpoklady zvíťaziť. Jedným z najznačnejších je materiálová hodnota figúrok. Jednotlivé figúrky majú všeobecne známu priradenú hodnotu - napr. jazdec a strelec sú obidvaja známy pod hodnotou 3 body, no

*Semestrálny projekt v predmete Metódy inžinierskej práce, ak. rok 2021/22, vedenie: Vladimír Mlynarovič

nie je to také jednoduché. Ich hodnota taktiež závisí od viacerých faktorov - od pozície, teda ako dobre pokrývajú ostatné figúrky a zároveň v akej miere ohrozujú súperove figúrky a možné ťahy, ako aj od štádia hry, napríklad ku koncu hry kde je menej figúrok je dvojica strelcov nadradená dvojici jazdcov, nakoľko majú väčšiu mobilitu ako aj synergiu, keďže sú schopné voľnejšie sa hýbať po šachovnici a pokryť dve celé diagonály. Otázkou je teda ako nastaviť algoritmus tak, aby bol schopný robiť ťahy ktoré ho dovedú k lepšej pozícií a v konečnom dôsledku aj ku výhre?

Prirodzene, základné šachové pravidlá a pohyby sú do šachového programu vštiepené na začiatku jeho vývoja. Ďalším krokom je potrebné jasne definovať aké je v ktoromkoľvek momente rozloženie figúr, aké majú možné ťahy a ako sa navzájom ohrozujú alebo zakrývajú. Pre tento účel sa vo väčšine prípadov algoritmus modeluje nasledovne: Každé políčko na šachovnici reprezentuje vektor s dĺžkou 64 tak, že každý komponent vo vektore smeruje k možnej pozícii na šachovnici. Tieto komponenty neskôr môžu nabrať rôzne hodnoty ktoré budú reprezentovať buď figúrky daného hráča alebo prázdne polia [4]. Teraz ale už nastáva čas, aby program začal robiť svoje prvé vlastné ťahy.

3 Vyhodnocovanie a vyhľadávanie ťahov

Pri začiatočnej fáze hry je potrebné algoritmus nejakým spôsobom usmerniť ako má postupovať. Univerzálne sa aplikujú dva postupy. Menej sofistikované algoritmy sa držia nejakým všeobecným pravidlám ako rozvíjať svoje jednotlivé figúrky. Tieto pravidlá sa dajú vizualizovať ako mapy šachovnice pre, ktoré priťahujú figúrky do určitých zón. Pre kráľa je napríklad esenciálne aby so dostal do bezpečia, teda jeho sa odporúča aby smeroval do rohov šachovnice. Naopak zas figúrka jazdca má svoju komfortnú zónu bližšie k strede šachovnice, kde má k dispozícii viac možných ťahov ako na kraji. Práve kvôli tomuto princípu je potrebné cielene formovať pomocou pravidiel program tak, aby usmerňoval svoje otváracie štádium hry na to aby posilnil svoju kontrolu nad centrom šachovnice. Druhý postup ktorý už využívajú pokročilejšie šachové programy sa zakladá na využití princípoch strojového učenia. Pre algoritmus sa stiahne rozsiahla databáza najhrávenejších a najúspešnejších otvorení ako aj otvorenia odohraté na najvyššej úrovni. Z tejto databázi algoritmus čerpá a modifikuje svoje otvorenia, ale pri následnom vývine hry už prichádzajú do vyhodnocovania neurónové siete.

3.1 Neurónové siete

3.2 Prístupy

Táto citácia je dočasne potrebná na správny chod referencí. [**?**]

- Quiescence Search
- Alfa-Beta Prunning
- Monte Carlo Tree Search

4 Testovanie

5 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] D.B. Fogel, T.J. Hays, S.L. Hahn, and J. Quon. A self-learning evolutionary chess program. *Proceedings of the IEEE*, 92(12):1947–1954, Dec 2004.

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