Универзитет „Св. Кирил и Методиј" Скопје
Факултет за информатички науки
и компјутерско инженерство

Анализа на два агента: Minimax против Expectiminimax во шах и нивна имплементација

семинарска работа

Ментор:
проф. д-р Соња Гиевска

Студенти:
Јана Анѓелкоска
Давид Дукоски

Скопје, 2025

Содржина

Вовед

Постојат различни типови на алгоритми што може да истражуваат детерминистички игри и нивната задача е да создадат интелигентен систем чија цел е да истражи што е најдобрата акција во секој момент од играта. Како најпознат од нив, класичниот Minimax алгоритам се потпира на добро дефинирана функција на проценка, што е од најголемо значење при преземање на некоја акција, поконкретно, за овој проект, играње на еден потег во шах. На друга страна, Expectimax алгоритмот има поразлично однесување од традиционалниот Minimax, и затоа вообичаено дава интересни резултати во игри на среќа. Целта на овој проект е да се анализира однесувањето на двата алгоритми кога ќе играат меѓусебно.

Фунцкија на проценка и евристички функции

Во овој дел ќе се разгледува еден од најважните аспекти на градењето агенти кои ќе играат потези кои се стратешки и тактички поволни. Ќе се соочиме со проблемот на дефинирање на соодветни тежини на „потпреблеми“ наречени **евристики**, кои всушност ќе даваат оцена на одреден аспект од шаховската позиција и ќе допринесуваат во дефинирање на севкупна оцена на една позиција.

Функцијата на проценка E

За еден алгоритам од спротивставено пребарување да игра разумни потези, важно е да се дефинира **функција на проценка/евалуација** (анг. Evaluation function). Формално дефинирано, фунцкијата на проценка $E: B \to \mathbb{Q}$, каде B е множество на сите позиици за шаховската табла, а $\mathbb{Q}$ е множеството рационални броеви. Оваа функција се користи за оцена на една позиција при постигнување на терминален јазел во дрвото на пребарување на било кој алгоритам за спротивставено пребарување.

Крајниот резултат $E(b), b \in B$ е рационален број кој ја рефлектира состојбата на шаховската табла според дефинирани **евристики** кои ги користи функцијата. Важно е да се напомене дека поради временски рестрикции за пресметка, што подлабоко оди агентот во дрвото (односно што подалеку гледа во индината) толку помало количество евристики се потребни во функцијата E, обратно на тоа, што поплитко истиот оди во дрвото толку повеќе евристики ни се потребни во истата функцијата. Причината за ова е да се постигне целта за агентот да истовремено игра најдобро што може и да се задржи брзината на пресметка на најдобрата вредност за E во дрвото.

Користејќи го горенаведеното, потребен е потегот на таблата кој доведува до пресметаната цел. Односно, доколку агентот игра со бели, и ако најдобрата позиција пронајдена со пребарување на длабочина d и $d \bmod 2 = 1$, тогаш:

$$\max_E \min_E E(b_{11}) = \ldots = \min_E \max_E E(b_{nd})$$

каде b_{ij} ја претставува i-тата позиција на длабочина j, и на длабочина d има n јазли со проширување на првиот разгледуван јазел. Доколку се води евиденција за најдобрите потези соодветни на секој исход од функциите, тогаш е пронајдена секвенцата потези (или линија) која според агентот треба да се игра.

Множеството H

Со намера да се дефинира (или, изгради на некој начин) алгоритмот кој ќе ја пресметува E, потребни се неколку евристички функции на кои на крај ќе им доделиме одредена важност (или **тежина**) пропорционално на колку допринесуваат на стратешката благосостојба на позицијата во сегашната фаза на игра. Во шахот се издвојуваат 3 фази во текот на играта: **отварање**, **средишница** и **завршница**. Нека H е множеството на евристички функции од облик $h : B \to \mathbb{R}$ кои ги користи E. Овде произволно може да се дефинира кои елементи точно ќе припаѓаат во него.

Главни фактори кои треба да ги имаме во предвид се: **квалитет** (или материјал), **стратешка предност** (пр. слободни пешаци, дуплирани пешаци, просторна контрола…), **генерални евристики** и **пенали**. Секоја од конкретните евристики кои се дефенирани подолу во А (Материјал), В (Стратешка предност), С (Генерални евристики) и D (Пенали) припаѓаат на H.

На следниот начин е дефиниран шаблон за како секоја евристика треба да се однесува, т.е во програмски термини, сите евристики ја наследуваат оваа тотално апстрактна класа:

```
from abc import ABC, abstractmethod

class Heuristic(ABC):

    """
        Functional interface to implement for a heuristic.
    """

    @abstractmethod
    def estimate(self, board: Board, color: Color) -> float:
        """
            A function that estimates the position, given 1 heuristic to
estimate with.

            Arguments:
            :param board: An arbitrary board state.
            :param color: The color for which the evaluation is executed.
            :return: An integer estimate of the position given the
heuristic.
        """
        pass
```

А. Материјал

Пресметката (евристиката) за **материјал** M е едноставна, бидејќи се доделува соодветна вредност на секоја од фигурите. Најчесто користени, и стандардни вредности се 9 - дама, 5 - топ, 3 - ловец и коњ и 1 - пешак. Својствено е дека кралот е фигура која не се вклучува во евристиката за материјал – целта е давање мат, а не земање на фигурата крал. Доколку k е бројот на коњи, b бројот на ловци, r бројот

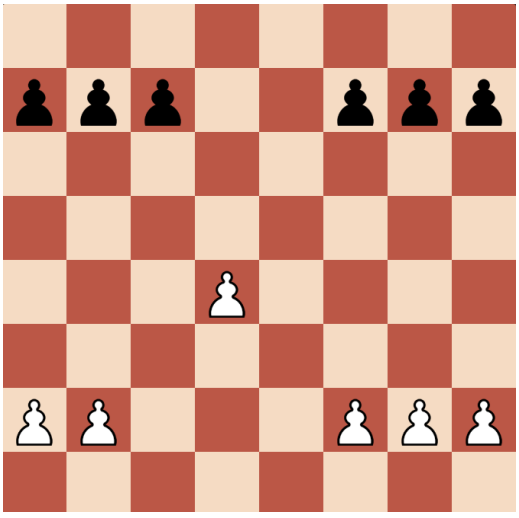
на топови, q бројот на кралици и p бројот на пешаци тогаш начинот за пресметување на материјалната состоба е:

$$\mathrm{M} = \mathrm{p} + 3(\mathrm{k} + \mathrm{b}) + 5\mathrm{r} + 9\mathrm{q}$$

Оваа евристика е наједноставна и очигледна, поради самата природа на играта, меѓутоа, не е пресудна. Наједноставен пример за ситуации кога материјалот не е најважен се тактичките позиции во кои едната страна „жртвува“ фигура со конкретна цел, која може да биде стратешка (често се однесува на слабеење на кралот/полиња околу кралот, смена на фигура што е од поголемо значење за противникот, отворени линии, слаби полиња...). Токму поради овие причини, мора да најдеме компромис меѓу општите стратешки правила на играта и оваа евристика. Притоа, секоја евристика која припаѓа на фамилијата „стратешки евристики“ може да има различна вредност во различни делови од играта, односно, да имаат различен придонес во вкупната проценка на партија врз основа на дали позицијата е отварање, средишница или завршница.

B. Стратешка предност

Интересен општ пример за ова, кој е поврзан со пешачки структури доаѓа од изолираните пешаци, односно пешачката структура која се нарекува ***Изолани.***



Изолани

Оваа постава на пешаците дава добри шанси за напад на кралот во средишница, бидејќи контролира клучни централни полиња, додека спротивната страна има повеќе шанси во завршници, каде што изолирани пешаци претставуваат слабост. Ова значи дека постојат спротиставени планови; Страната со изолиран пешак сака постепено да гради напад, додека спротивната страна сака да го неутрализира нападот и да влезе во завршница.

Пешачките структури се скелетот на партијата, бидејќи диктираат како ќе се постават фигурите и како ќе напредува играта, а впрочем се една и од најбитните и најобработените теми во областа на играта. Од тие причини, ќе ги дефинираме следните евристики што потекнуваат токму од пешачките структури:

- <u>Изолирани пешаци</u>

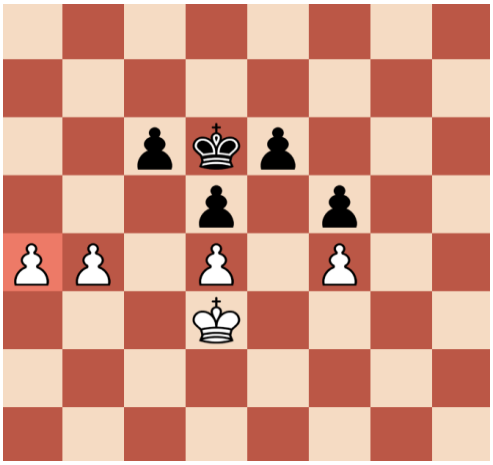
  Оваа евристика претходно дефиниравме како може да делува на партијата и плановите на играчите. Општо, изолираните пешаци кои **не се централни,** односно кои не доведуваат до пешачката структура Изолани, се сметаат како мала слабост за страната која ги има. Ова произлегува од едноставната причина дека не може да се одбранат од други пешаци, и се смета за непосакувано. Во општ случај, за подобра пешачка структура се смета онаа која ги нема таканаречените „острови“, односно неповрзани пешаци. Всушност, еден изолиран пешак е сам по себе „остров“.

- <u>Дуплирани пешаци</u>

  Пешаците од иста боја што се наоѓаат на иста колона се дуплирани, и во општ случај, не се лоши, но имаат своја негативна страна, бидејќи во одредени ситуации се полесни за напаѓање (особено во завршница), но и поради **помалата подвижност** на пешаците. Меѓутоа, тоа што треба да добие повисок „пенал“ се дуплирни и изолирани пешаци, што генерално се сметаат за слабост во секој период од играта.

- <u>Слободни пешаци</u>

Кога пешаците што на нивната колона и на соседните колони (лево и десно од неговата) немаат пешак од противникот се нарекуваат слободни пешаци. Вообичаено се силни во секој период од играта поради нивната подвижност, а често се причина за дефанзива на противникот, меѓутоа нивната сила е значително истакната во завршницата, бидејќи се користат како мотив за принципот „слабост на две страни“.



Завршница со слободен пешак

- Тип на позиција

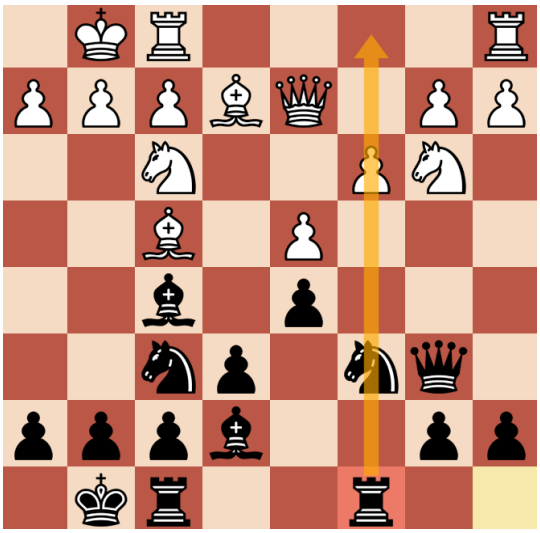
  Многу општо, типот на позиција може да го поделиме на отворен или затворен врз основа на пешачката структура. Имено, доколу позицијата има помалку пешаци (особено централни), тогаш фигурите може да контролираат повеќе отворени линии и дијагонали. Во затворени позиции, мобилноста на фигурите е помала - ова е резултат на блокирани/неподвижни пешаци кои вообичаено се во поголем број, и ја ограничуваат контролата на отворени линии и дијагонали.

При составување на евристиките треба да се земе предвид дека секоја фигура има своја евристика, и со ова може да дефинираме уште една „фамилија“ на евристики што доделуваат јачина на секоја фигура посебно. Причината за ова е едноставна; секоја фигура има различна

улога, различна тежина и во различни ситуации има различна предност или слабост. На пример, во затворени состојби, коњите имаат поголема моќ за подвижност - со тоа и поголема контрола на клучни полиња. Во отворени состојби, ловците се генерално подобри, а може и да го ограничат движењето на останатите фигури поради контрола на дијагоналите. Притоа, веќе сме дефинирале фунцкија поврзана со пешачката структура која може да ни помогне за тој конкретен случај. Следните евристики се дефинирани за секоја фигура посебно.

- Евристика за топ

  Топовите се сметаат за силни фигури кога во средишници и завршници се наоѓаат на отворени линии - колоните на таблата на кои нема пешаци. Јаки топови исто така се наоѓаат на полуотворени линии - оние каде што има само еден пешак, бидејќи најчесто овозможуваат поголема контрола, подвижност и можност за маневри, а истото може да го употребите како евристика за дама. Генерално, ова својство е пресудно за важноста за оваа фигура.



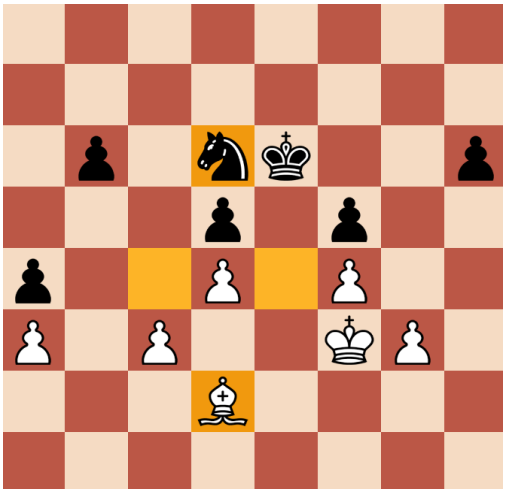
Топ на полуотворена линија

- Евристика за ловец

  Како што е опишано претходно, ловците се силни во отворени позиции поради можноста да контролираат дијагонали, и како последица да контролираат повеќе полиња и да ограничат други фигури. Меѓутоа, постојат различни позиции во кои ловците може да се блокирани од други фигури, што најчесто се пешаци. Карактеристично својство за ловците е дека делуваат најдобро во пар, особено ако позицијата е отворена. Често се преферираат во партии, особено кога играчот напаѓа, или во **завршници**, особено во позиции каде што противничката пешачка структура е подлежна на напад.

- Евристика за коњи

  Коњите имаат уникатен начин на движење што им дава можност за интересни стратешки планови, кои се особено значајни во затворени позиции, каде што најдобро се истакнува нивната можност за активност на таблата. Вообичаено се силни на аутпости - полиња каде што се бранети од пешаците на едниот играч, а не може да се нападнати од пешаците на противникот - но, ова е случај за повеќето фигури и секоја од нив е посилна (или еднакво силна) од другите фигури во различен тип позиции.



Силен коњ против слаб ловец

C. Генерални евристики

Групата на евристики чија цел е да ја испитаат целокупната хармонија на таблата се **генералните евристики**, во кои спаѓаат мобилност (подвижност) на фигурите и контрола на центарот.

- Подвижноста на фигурите на еден натпреварувач, наједноставно може да ја добиеме ако го земеме вкупниот број легални потези за таа страна, така што ова би ги поттикнало алгоритмите да ги стават фигурите на позиции што им овозможуваат поголема подвижност. Понатаму, оваа функција може да се прошири така што може да испитаме дали постојат заробени фигури или секвенца на потези што заробуваат фигура.

- Контрола на центарот е една од побитните евристики на почетокот на партијата и вообичаено, пешачката структура е добар индикатор за колку една страна владее со централните полиња (e4, e5, d4, d5). Притоа, страната што има пешаци на овие полиња би имала мала предност во тој поглед, но исто така се земаат предвид и фигурите што ги бранат (односно напаѓаат) тие полиња.

D. Пенали

Во раните фази на игра, вообичаено, потезите со крал (што не се рокада) или со тешки фигури (дама и топ) се сметаат за непосакувани. Причина за ова е дека лесните фигури (коњ, ловец и пешаци) може лесно да ги напаѓаат, а ни на помали длабочини на истражување и да ги заробат. Поради тоа, корисно е да се имплементираат пенали - функции што даваат казна за вакви потези на почеток на партијата.

Впрочем, овие функции треба да се вклучени во евалуацијата на позицијата, бидејќи просторната контрола на фигурите е исто така важна, односно, колку повеќе

полиња напаѓа некоја фигура, толку се зголемува нејзината вредност. Доколку не би давале пенали на рани потези со тешките фигури, **очекувано однесување на алгоритмите би биле рани потези со дама** (поттикнато од генералните евристики што наградуваат „владеење“ со повеќе простор).

Пенал што се однесува на безбедност на кралот испитува колку полиња околу кралот се нападнати, така што алгоритмите би биле во одредена количина „казнети“ ако полињата околу кралот (или ако неговата позиција) се небезбедни.

Како дел од проблемот се јавува потребата да се вреднуваат некои од евристиките повеќе од други во одредени делови од играта (на пример, активност на крал во завршница). Вообичаено, нема одредено „правило“ според кое може да речеме дека позицијата е во средишница. Бројот на потези не е добар индикатор за ова, бидејќи постојат отварања што може брзо да преминат во завршници, како и средишници што траат повеќе потези. Наместо ова, може да се искористи моменталниот збир на фигурите на таблата, што ќе биде главен показател за моменталната состојба. Притоа, во предвид се ќе се земаат и бројот на пешаци на двете страни. Со помош на ова, може да добиеме функција што ќе ни враќа во кој период од играта се наоѓаат двата алгоритми.

Целосна дефиниција за E

Нека сите горенаведени евристики припаѓаат во H, односно $h \in H$, ако h е било која од истите и нека $b \in B$. Откако сме ги дефинирале тежинските вектори $\overrightarrow{w_o}$, $\overrightarrow{w_m}$ и $\overrightarrow{w_e}$ од $\mathbb{R}^n$, каде n е број на евристики кои ги користиме, можеме прецизно да ја дефинираме фунцкијата E како линеарна комбинација (или внатрешен/скаларен производ) на вредностите од секоја евристика, каде елементот на позиција i, $0 < i \leq n$ во соодветниот вектор на тежини соодветствува на вредноста на евристиката со позиција i во векторот $\vec{v} = [\, h_1(b) \;\; h_2(b) \;\; \ldots \;\; h_n(b) \,]$, исто од $\mathbb{R}^n$:

$$E(b) = \overrightarrow{w_k} \cdot \vec{v} = \vec{w}_k \overrightarrow{v^T} = \sum_{i=1}^{n} w_{ki} v_i$$

Каде k е еден од o, m или e во **зависност од која фаза на играта е тековна** (o - отварање, m - средишница или e - завршница). Конкретните вредности во овие вектори нема да ги дискутираме, ама ќе кажеме дека тежините се подесени преку набљудување на однесувањето на алгоритмите во повеќе игри. Генерално, евристики што се вреднуваат најмногу во отварање се однесуваат на контрола на центарот, активност на фигури и пенали поврзани со рано играње на дама или крал. Во средишница, истите работи се наградени, меѓутоа нема пенали за движење на тешките фигури, притоа, подвижноста на фигурите и безбедноста на кралот се поважни од претходно. Во завршница, посакувани се отворени ловечки дијагонали и топовски линии, а голема улога играат и сите пешачки евристики.

Адаптирање на Expectiminimax

Прво, да разјаснеме дека Expectiminimax **не е алгоритам направен за игри како шах (пример за потврда на ова е целта на проектот)**, поради тоа што шах е детерминистичка игра. Но, можеме да го адаптираме за да игра случајно за разлика од Minimax, со цел некогаш да го изненади својот противник. Ќе дефинираме веројатностна распределба, која е низа a_n така што генерираме n случајни вредности (за n легални потези, каде позициите кои произлегуваат од нив со нивни вредности на функцијата на проценка), односно ако една вредност е случајна променлива V, таа е таква што $V \sim U(\mathbf{0}, \mathbf{1})$. Потоа за да се добие случајна веројатностна распределба секоја вредност ќе се подели со сумата на случаните вредности (нормализирање):

$$a_i = \frac{a_i}{\sum_{j=1}^{n} a_j}, 0 < \ i \ \leq n$$

За да се добие потегот со најголемо математичко очекување, го бираме преку:

$$arg \max_{a \in a_n} aE(b_{id}) \ ,$$

Под претпоставка дека вредноста a важи за потегот кој доведува до позиција b_{id} , доколку пребаруваме на длабочина d. Користеме $arg\ max$ за лесно да можеме да пристапуваме до тој потег во генерираните потези програмски. Исто така, овој агент претпоставува дека противникот игра оптимално, односно дека е обичен minimax.

Оптимизација

Во овој проект, од аспект на оптимизација ќе се разгледува времето што им е потребно на алгоритмите да истражат потези до одредена длабочина што ние произволно ќе ја поставиме. Прво ќе направиме мала анализа врз тоа како влијае светот на 8x8 таблата врз пресметковниот интензитет, гледајќи ги можните позиции на таблата после одреден број на потези од еден играч (анг. ply).

Број на потег (еден потег од 1 играч)	Можни позиции
1	20
2	400
3	8.902
4	197.281
5	4.865.609
…	…
60	$\approx 10^{40}$

Бројот на можни позиции расте експоненцијално со бројот на изиграни потези. Поточно, бидејќи временската комплексност на minimax е $O(b^d)$ каде факторот на разгранување на дрвото е b, каде $b \in [\mathbf{30, 40}]$, од обопштено гледиште на играта, а d е бројот на потегот на таблата, односно длабочината на која сакаме агентот да пребарува. Доколку агентот има пронајдено потег во отварањето со досегашниот имплементиран minimax алгоритам на длабочина 4, добиваме 26,36 секунди за процесирање после првиот потег, и 41,96 после вториот. Доколку направиме неколку оптимизации, добиваме 7,18 секунди и 0,01 секунди соодветно (ова не е аномалија, објаснето е зошто во следните потточки).

α-β поткастрување

Овој класичен метод за оптимизација го спречува minimax алгоритмот од разгледување позиции кои немаат потреба да се прегледувани. Ова го извршува така што доколку веќе се пронајде оптималниот потег за бел (или црн) на длабочина d, да се спречи разгранувањето. Нека α е евалуацијата на досега максималната евалуација во разгледаните јазли, а β минималната. Доколку кај некој јазел е задоволено $\beta \leq \alpha$, ќе престанеме да гледаме понатамошни гранки на кај тој јазел, бидејќи ако сме пронашле ваква евалуација, никогаш нема да пронајдеме позиција (јазел) која ќе го замени досегашниот најдобар резултат кај јазелот во гранките кои следат од јазелот.

Порано поткастрување

Можеме дополнително да го забрзаме процесот на пребарување преку наложување на порано поткастрување кај дрвото на пребарување. За таа цел, може да се дефинира правило за подредување на потезите, со цел веројатно подобри потези да се разгледуваат први и да се поткастри дрвото кога ќе наиде алгоритмот на нив. Ова ќе го правиме според 3 евристики: **MVV-LVA** (Most Valuable Victim - Least Valuable Aggressor), историска евристика (**History Heuristic**) и „потези убијци“ (**Killer moves**).

A. MVV-LVA

MVV-LVA е алгоритам кој ќе ги подреди како први потезите каде се зема фигура т.ш им дава приоритет на оние каде најмалку вредната фигура зема што повеќе вредна фигура.

B. Историска евристика

History heuristic помага со подредување на потези кои веќе биле оптимални на одредена длабочина (изиграни), т.ш приоритет им се дава на тие кои биле почесто играни, и на поголема длабочина.

Пример, доколку потегот ♗xd4 веќе е пронајден како изигран претходно и сега повторно е изигран и пронајден на длабочина d, и вредноста на историската евристика за ♗xd4 е v, вредноста е ажурирана т.ш $v \leftarrow v + 2^d$. На овој начин не само што се следи фреквенција на вакви потези, исто така се дава експоненцијална награда доколку се изигра на поголема длабочина. Доколку не се изиграл потегот претходно, но е изигран сега, тогаш неговата вредност во историската табела е 2^d.

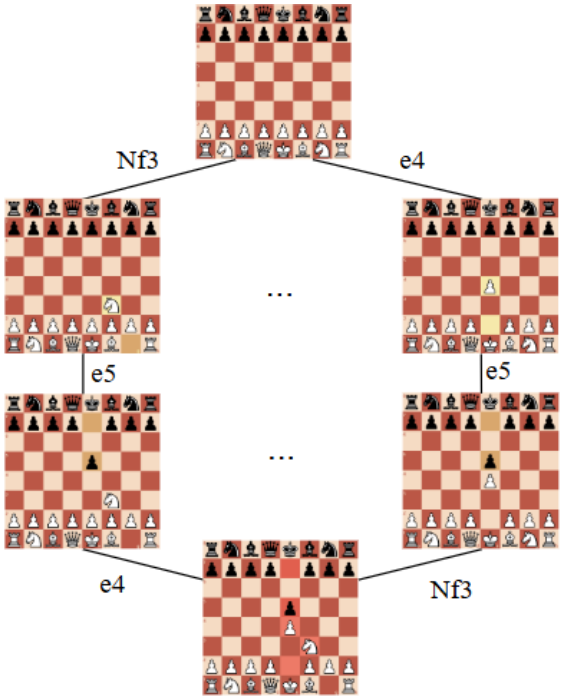
C. „Потези убијци“

Killer moves се потези кои предизвикале поткастрување на дрвото на играта. Овие потези природно би имале највисок приоритет во сортирањето на потезите за прегледување, бидејќи со максимална подобност би го поткастриле дрвото.

Приоритетот на евристиките во сортирањето е во редослед: Killer Moves, MVV-LVA па History Heuristic. За Killer Moves веќе разјаснавме зошто се први во подредувањето. MVV-LVA доаѓа пред History Heuristic поради тоа што природно е поверојатно според MVV-LVA потегот да е добар, наспроти History Heuristic.

Транспозиции

Транспозиција во шах е позиција која може да се добие преку различни пермутации на секвенца на потези кои и претходат.



Тука гледаме еден пример за како можеме да стигнеме до позицијата со **FEN** rnbqkbnr/pppp1ppp/8/4p3/4P3/5N2/PPPP1PPP/RNBQKB1R b KQkq -. Се гледа дека можат да постојат огромен број вакви пермутации за секоја можна позиција, и нема потреба да ги пресметуваме евалуациите за нив, доколку веќе ги знаеме. Со цел да го направиме ова, ќе го користиме концептот на **мемоизација** и ќе чуваме **табела на транспозиции**, која е всушност хеш табела. Клуч во ваквата табела ќе ни претставува првите 4 дела од FEN-стрингот на позицијата, односно поставката на фигури на таблата, кој е на потег, права на рокада и поле на en-passant.

Кога ќе се најде оптимален потег, тој ќе се зачува, заедно со длабочината на која е изигран и евалуацијата на таблата во тоа време. Ова се прави бидејќи не е посакувано да зачуваме позиции кои веќе постојат, ама се пронајдени на пониска длабочина, што значи дека е помалку прецизна евалуацијата за таа позиција.

Исто така, за да видиме дали сегашната позиција, доколку постои во табелата, можеме да ја искористиме нејзината соодветна постоечка евалуација, ќе провериме дали сегашната длабочина е поголема или еднаква од таа која е зачувана за сегашната позиција. Доколку не е тоа случајот, ние не можеме да го искористиме зачуваниот резултат, бидејќи не се важи за доволно длабок/презицен.

Поради оваа табела, некогаш агентот игра потег во истиот момент и не пребарува низ дрвото, бидејќи веќе има зачуван резултат како одговор на нашиот потег на ниска длабочина и ова го објаснува времето искористено за вториот потег во **4**, кое е 0.01 секунди.

Резултати

После 10 изиграни партии меѓу агентите, резултатите од игрите се:

Играч	Победа	Пораз	Реми
Minimax	8	0	2
Expectiminimax	0	8	2

Според резултатите, забележуваме дека адаптацијата на Expectiminimax скоро никогаш нема да го победи Minimax. Едноставно, детерминистичката природа на шах се поклопува со таа од Minimax, но Expectiminimax знае да игра интересни потези кои произлегуваат ретко и случајно, но доведуваат до некаква борбена шанса која резултира во реми. Дополнително, двата агенти успешно во играта ги интегрираат имплементираните алгоритми за секоја од евристиките, и им служат како успешна водилка во трите фази од играта. Прецизноста при игра на Minimax во игрите достигнува до 85% (ова значи дека 85% од потезите што ги игра се совпаѓаат со најдобрите или приближно најдобрите потези што можат да се изиграат во партијата), а 70% на адаптацијата на Expectiminimax.

Полиња за подобрување

Справување со временски ограничувања

Поради сегашните перформанси на агентите, разгледување јазли на $d > 4$ одзема доста од нашето време, покрај сите оптимизации. Главната причина за ова е самата околина во која е развиен, односно python 3.12. Доколку сакаме многу подобри перформанси на агентите (тие слични на Stockfish, Fritz…) треба да се развијат истите во јазици како C++ или rust, односно јазици кои оперираат на пониско ниво, поблиску до хардверот, со цел оптимално да се генерираат инструкции. Дури и неколку дополнителни инструкции можат да ги успорат перформансите на еден шаховски агент, поради експоненцијалната природа на пребарување низ огромниот простор на состојби на таблата. Пример за ваков агент е познатиот **Stockfish**, кој е имплементиран во C++.

Напредни техники за градење агенти

Користење понапредни техники од областа на учење со поттикнување (анг. reinforcement learning) како што се **DQN** (Deep-Q Network), **MCTS** (Monte-Carlo Tree Search) и областа на **длабокото учење** (анг. Deep learning), поспецифично длабоки невронски мрежи или комбинација од двете техники, можат значително да ги подобрат перформанците и временски и во однос на квалитет на играње на агентот. Пример за ваков модел е **AlphaZero**, кој е имплементиран како комбинација на MCTS и длабоки невронски мрежи.

Технички препораки за започнување во оваа област

Доколку сакате да развиете агент сличен на тие во проектот, што игра брзо и е во конкуренција со просечни шаховски играчи, од огромна важност е да користете јазик на пониско ниво до хардверот, како што е наведено во **5.1**. Од друга страна, доколку целта Ви е да воспоставите имплементација и потврда на некоја теоретска идеја

или концепт, python би бил идеален јазик и всушност ова беше нашата цел. Претходно наведените препораки важат доколку користете minimax алгоритам и негови оптимизации, но доколку користете понапредни техники, како на пример од областа на учење со поттикнување, резултатите треба да се доста подобри и со користење јазик како python. Обопштено, концептите со кои треба да сме запознаени пред развивање ваков проект се следни:

- Детерминистички и недетерминистички повеќе-агентни средини;
- Спротивставено пребарување;
- Произволен програмски јазик врз основа на целта на развивањето;
- Играта шах, нејзините правила и различни стратешки концепти во истата. (пешачки структури, поставување фигури, безбедност на крал…);
- Развивање потребни евристики (множеството H дефинирано во **2.2**), кои го насочуваат агентот на играње логични потези, во дадена позиција;
- Назначување тежини на секоја од еврискиите, како и вредност која ја враќаат, со цел да се изгради функцијата E дефинирана во **2.1**;
- Користење на веќе имплементирана табла (пр. python-chess) со правила во соодветниот програмски јазик (ова е важен чекор бидејќи е веројатно оптимално имплементирана играта на шах со библиотеките, но можна е и рачна имплементација доколку е соодветно оптимизирана);
- Функционалност на избраниот алгоритам за спротивставено пребарување без никаква оптимизација, каде се интегрира имплементацијата на функцијата евалуација;
  или користење дополнителни методологии од длабоко и/или учење со поттикнување;
- Временска оптимизација и фино-подесување (анг. fine-tuning) на избраниот алгоритам (доколку е можно).

Горенаведените концепти не мора да се проучат длабоко со цел да се имплементира, но доста е олеснително доколку се знаат на тоа ниво. Во рефенците подолу можете можете да најдете видео кое е доста корисно после сфаќање на овие концепти. Дополнително, во **репозиториумот на проектот** можете да најдете

експериментален директориум наречен utils каде можете да тестирате генерација на потези до одредена длабочина со цел да го истражете просторот на пребарување и неговиот обем.

Референци и искористени материјали

- Mauricio Flores Rios. (2015). Chess Structures: A Grandmaster Guide. Quality Chess.
- Stuart J. Russel & Russell Norvig. (2022). Artificial Intelligence: A Modern Approach. Pearson.
- David C Lay. (2011). Linear Algebra And Its Applications. Pearson.
- AlphaZero. (2017). In Wikipedia. URL https://en.wikipedia.org/wiki/AlphaZero
- Stockfish. (2008). In Wikipedia. URL https://en.wikipedia.org/wiki/Stockfish_(chess)
- MVV LVA. (2018). Chess Programming wiki. URL https://www.chessprogramming.org/MVV-LVA
- History Heuristic. (2018). Chess Programming wiki. URL https://www.chessprogramming.org/MVV-LVAy Heuristic
- Killer Move. (2018). Chess Programming wiki. URL https://www.chessprogramming.org/Killer_Move
- Memoization. (2020). In Wikipedia. URL https://en.wikipedia.org/wiki/Memoization
- Forsyth-Edwards Notation (FEN) (?). Chess.com. https://www.chess.com/terms/fen-chess
- Sebastian Lague. (2021). Coding Adventure: Chess. URL https://www.youtube.com/watch?v=U4ogK0MIzqk&embeds_referring_euri=https%3A%2F%2Fwww.bing.com%2F&embeds_referring_origin=https%3A%2F%2Fwww.bing.com&source_ve_path=MjM4NTE