УДК 004.89

К.С. Глинська, студент магістратури,
Н.С. Костюкова, канд. техн. наук, доц.
ДВНЗ «Донецький національний технічний університет», м. Покровськ, Україна
katerinaglinskaya96@gmail.com,
natalia.kostiukova@donntu.edu.ua

Реалізація штучного інтелекту для гри в шахи

В роботі пропонується реалізація штучного інтелекту для гри в шахи на основі алгоритму альфа-бета відсікання з модифікаціями, аналізується ефективність запропонованих модифікацій у порівнянні з популярними шаховими програмами.
Ключові: штучний інтелект, гра в шахи, стратегічна гра, альфа-бета алгоритм, оцінка позиції, MVV/LVA, BITBOARD.

DOI: 10.31474/1996-1588-2017-2-25-32-39

Вступ

При створенні комп'ютерних ігор однією з проблем є реалізація для персонажів, керованих комп'ютером, поведінки, що справляє враження інтелектуальної, тобто за своєю складністю наближається до поведінки гравця- людини. Існує велика кількість підходів до реалізації штучного інтелекту в іграх, ці підходи суттєво відрізняються залежно від жанру гри. Так, існують свої особливості при реалізації штучного інтелекту в розвиваючих комп'ютерних іграх [1], шутерах, стратегіях, стимуляторах та ін. [2]. Загальними рисами у всіх випадках є неповна інформованість про дії гравця- людини та необхідність прийняття рішень у реальному часі. Зазвичай під ігровим штучним інтелектом розуміється сукупність програмних засобів, що використовуються в комп'ютерних іграх для моделювання розумної людської поведінки в діях персонажів, що управляються комп'ютером.

Шахи і шашки – ігри, історія яких налічує багато століть – були в числі перших ігор, що отримали комп'ютерну реалізацію. Так, в 1951 році в манчестерському університеті для комп'ютера Ferranti Mark I Кристофером Стречі була створена програма для гри в шашки, а Дитріом Принцем - для шахів[3] У подальшому комп'ютерні програми – симулятори шашок та шахів – були «полігоном» для випробування різних методів штучного інтелекту. Зусилля дослідників не були марними: у кінці 1950-х – на початку 1960-х років Артуром Семюелем був створений стимулятор шашок, рівень інтелектуальності якого був достатнім для гри з чемпіоном світу [4], а у 1997 році програма для комп'ютера Deep Blue виграла матч з шахів у чемпіона світу Гарі Каспарова [5.].

Незважаючи на довгу історію досліджень штучного інтелекту для гри в шахи, не існує єдиного алгоритму та погляду на програмування такого інтелекту, що пов'язано з існуванням різних підходів й комбінацій евристик, що обирає програміст на свій розсуд, та з неспроможністю сучасних комп'ютерів задовольнити всім вимогам розробника. За думкою математика О. С. Кронрода, гра в шахи є «дрозофілою штучного інтелекту» - зручним об'єктом для дослідження цієї галузі науки [6].

Метою дослідження є підвищення інтелектуальності шахових програм без збільшення часу на «обдумування» ходів.

Задачею дослідження є створення алгоритму штучного інтелекту для гри в шахи, що є кращим за алгоритми, реалізовані в програмах – аналогах з точки зору часових характеристик та можливості виграшу**.**

Дослідження виконано в рамках проекту Erasmus+ «GameHub: University-Enterprises Cooperation in Game Industry in Ukraine» 561728-EPP-1-2015-1- ES-EPPKA2-CBHE-JP.

Основні поняття та методи реалізації штучного інтелекту шахів

У 1951 році Клод Шеннон [7] описав дві стратегії пошуку кращого ходу в шахах, які грунтуються на евристичній функції оцінки кінцевих точок: тип А – передбачає перебір всіх можливих ходів на фіксовану глибину, з викликом в кінці оціночної функції (неможливо здійснити повний перебір, враховуючи обмежену продуктивність процесорів); тип В – виконує тільки вибіркові розширення певних вузлів дерева позицій (ходів), використовуючи накопичені шахові знання, щоб підрізати нецікаві гілки. У чистому вигляді ці методи сьогодні не використовуються, але, в залежності від уподобань автора конкретної програми, робиться ухил в бік одного або іншого варіанту. Приблизно до 1973 року всі шахові програми були типу В, які головним чином грунтувалися на статичній оцінці. З появою більш потужних процесорів програмісти стали створювати програми типу А.

Будь-яка шахова програма повинна реалі-

зовувати наступні кроки:

а) пошук та моделювання можливих ходів гравців на деяку глибину;

б) оцінка отриманих кінцевих позицій для їх порівняння та вибору кращого варіанту.

Від того, які алгоритми та методики використає програміст для генерації ходів та оцінки ситуації на полі, залежить рівень штучного інтелекту його гри.

Більшість сучасних шахових програм можна віднести до однієї з трьох категорій, які відрізняються функціями оцінки, розпізнаванням складних позиційних стратегій гри, швидкістю прийняття рішень [8]:

а) програми fast searchers;

б) програми knowledge-based;

в) гібриди між двома першими категоріями.

Проблема створення шахової програми полягає в тому, щоб знайти оптимальний компроміс між швидкістю та якістю штучного інтелекту. Хоча більшість сучасних комп'ютерів мають досить високу продуктивність, покладатися лише на це було б нерозумно, враховуючи кількість розрахунків, які треба зробити навіть оптимізованій програмі. Якщо кількість можливих позицій в шахах після першого кроку білих дорівнює 20, то в ході гри це число стрімко зростає і після трьох ходів білих і трьох ходів чорних кількість можливих позицій сягає майже 121 млн. [9] Повне дерево перебору в шахах, хоча і є кінцевим, формувати та аналізувати неможливо, але це і не потрібно. Шахіст рахує варіанти в середньому на 5-6 ходів, тому досить навчити програму будувати й оцінювати ходи на обмежену глибину, знаходити цікаві й відкидати некорисні, відсікаючи зайві гілки дерева.

Для побудови дерева рішень та знаходженні вигіднішого ходу для комп'ютерного гравця, використовуючи отримані оцінки, застосовуються алгоритм мінімакс, NegaMax та їх модифікації й доповнення [10, 11, 12].

Після генерації можливих ланцюжків ходів треба оцінити результат – поточну ситуацію на полі. Існує матеріальна та позиційна оцінка. Перша враховує лише кількість наявного матеріалу – фігур своїх та противника – з урахуванням цінностей кожного типу фігур. Кожен програміст може по-різному встановити їх вартість, але приблизне співвідношення відоме й наведене на рисунку 1.

Рисунок 1 – Цінність фігур в шахах

Як правило, оцінюючи цінність кожної фігури, враховують її розташування на полі. Наприклад, пішакам важливіше прагнути досягти краю дошки, коням – контролювати середину поля, королю на початку гри – знаходитись в куті, а в ендшпілі (завершальний етап гри) – переходити у центр [13]. Таким чином, для отримання вартості кожної фігурі треба враховувати додатковий коефіцієнт клітини, на якій він розташований, для чого створюють спеціальні вагові матриці для кожного типу фігур.

Окрім цього, при формуванні позиційної оцінки можуть надаватися бонуси або штрафи за наступні умови ігрової ситуації:

а) кількість полів під боєм своєї сторони та сторони суперника;

б) зайняття ключових позицій: у центрі дошки для коня або ферзя, відкритих прямих для тур, головних діагоналей для слона;

в) наявність прохідних пішаків;

г) наявність заблокованих (здвоєних) пішаків;

д) була здійснена рокіровка;

е) рівень безпеки короля, відстань між королем та ворожим ферзем зменшилася;

ж) захист своїх фігур, х-гау позиція – розташування ферзя та слона на одній діагоналі під взаємним захистом посилює фігури та сприяє атаці;

з) кількість можливих для гравця ходів та інше.

Окрему увагу при розробці шахових програм слід приділити ефекту горизонту – обмеження кількості ходів, що розглядаються. При цьому деякі важливі моменти гри можуть буди не помічені штучним інтелектом, через що він зробить помилкову оцінку позиції. Наприклад, противник віддає цінні фігури, плануючи мат через декілька ходів, але програма не бачить цього через ефект горизонту – вона оцінить ситуацію на полі як сприятливу для себе. Тому існує декілька правил, коли треба поглибити генерацію ходів, незважаючи на обмеження на рівень дерева:

а) відбувається розмін фігур;

б) наявний прохідний пішак, що перетвориться на іншу сильну фігуру;

в) здійснено шах королю;

г) отримана оцінка ходу, що значно відрізняється від інших або очікуваної;

д) стрімко зростає напруженість біля короля.

Ці ситуації мають сильний вплив на ігровий процес, тому повинні розглядатись до деякої стабілізації ситуації. Цей процес збільшення глибини дерева отримав назву «пошук спокою» (Quiescence Search). У інших випадках генерація варіантів завершується або по досягненні заданої глибини пошуку, або по закінченню виділеного часу.

На сьогоднішній день користувачам доступна велика кількість шахових програм. На рис.2 наведено рейтинги лідерів шахових движків з оцінкою Ело [14].

Більшість сучасних шахових програм на-

дають схожі можливості. Як правило, користувач може обирати сторону («білі» або «чорні»), бачити історію ходів, відміняти хід, обирати складність гри. Деякі шахи дозволяють обмежувати час на хід або гру взагалі, зберігати гру, укладати рейтинг, отримувати підказки під час гри, бачити можливі ходи та інше.

Rank	Name	Rating			Score
		Elo	+	-	
1	Komodo 8 64-bit 4CPU	3322 (+18)	+20	-20	71.5%
2	Stockfish 5 64-bit 4CPU	3300 (+17)	+20	-19	69.0%
3	Houdini 4 64-bit 4CPU	3297 (+23)	+22	-22	67.2%
4	Gull 2.8b 64-bit 4CPU	3221 (+20)	+30	-30	53.7%
5	Equinox 3.20 64-bit 4CPU	3201 (+14)	+25	-25	51.8%
6	Critter 1.6a 64-bit 4CPU	3188 (+13)	+18	-18	54.2%
7	Rybka 4.1 64-bit 4CPU	3175 (+15)	+18	-18	53.4%

Рисунок 2 – Рейтинг шахових програм за 2017 рік

Серед основних недоліків існуючих програм найбільш поширені наступні:

а) тривалий вибір ходу програмою – до 10-30 секунд для високої складності гри;

б) відсутність можливості зміни інтерфейсу програми;

в) відсутність можливості відмінити хід та отримати підказу під час гри;

г) відсутність можливості продовжити незавершену гру;

д) неможливість змінити сторону або складність гри в процесі.

Реалізація штучного інтелекту у шаховій програмі

В програмі, за допомогою якої проводились експерименти, реалізовані наступні евристики та алгоритми:

а) алгоритм NegaMax та його оптимізація – альфа-бета відсікання для пошуку кращого ходу;

б) алгоритм пошуку з нульовим вікном та NegaScout;

в) евристика нульового ходу;

г) PV-таблиці, Zobrist-ключи та хеш-таблиці для запам'ятовування кращих стратегій гри та оцінок позицій;

д) алгоритм ітераційного занурення та алгоритм пошуку спокою при побудові дерева рішень;

е) принцип MVV/LVA для сортування захоплень фігур;

ж) евристики вбивця та історії для сортування «тихих» ходів;

з) алгоритми матеріальної та позиційної оцінки;

и) подання дошки через бітборди, генератори ходів, атак, магічні бітборди.

Для подання дошки була використана техніка BitBoard, що передбачає кодування позицій фігур на дошці бітами 64-розрядного числа без знаку [15]. Наприклад, розташування білих пішаків, описане в бітовій нотації, наведено на рисунку 3.

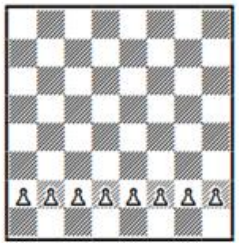


Рисунок 3– Позиції білих пішаків в бітовій нотації

Використовуючи побітові операції зсуву, порозрядні AND, OR, XOR та інші, можна отримувати маски взять та генерації переміщень. Використання бітбордів дозволяє зменшити об'єм пам'яті та збільшити швидкість.

Перед початком рахунку мають бути сформовані наступні початкові структури даних:

а) список фігур по клітинах поля;

б) 64-бітові числа маски для всіх чорних й білих, у яких біти, що відповідають зайнятим позиціям, встановлені в 1;

в) масив обчислених заздалегідь бітових масок для можливих переміщень (атак) деяких фігур (тури, слона, коня, короля) для кожної клітини дошки;

г) масиви «магічних» чисел для генерації атак далекобійних фігур за принципом магічних бітбордів [16];

д) чотири масиви для кожної сторони, що містять дані, скільки на кожній горизонталі, вертикалі, висхідній і низхідній діагоналях знаходиться фігур.

Матеріальну оцінку розраховують як загальну цінність своїх фігур мінус цінність фігур суперника. Але у цієї формулі слід додатково врахувати позиційну оцінку фігури, тобто її цінність залежно від розташування на полі. Позиційна оцінка фігури виконується на основі матриці коефіцієнтів (кожен її елемент відповідає певній клітині), значення яких помножають на матеріальну оцінку фігури.

Додатково до оцінки слід додавати бонуси за кількість атакованих своїми фігурами позицій (в тому числі, коли вона «атакує» власну фігуру, адже це рівнозначно її захисту) та, відповідно, штрафи за такі ж самі загрози з боку суперника. Для оцінки напруженості коло короля можна враховувати відстань між королем та ферзем [17].

Розглянемо алгоритм штучного інтелекту. Спочатку відшукується кращий хід за алгоритмом ітераційного занурення й альфа-бета відсікання. Якщо досягнута максимальна глибина або час, відведений на хід, вийшов, штучний інтелект робить кращий знайдений хід. Після цього оновлюється логічне представлення дошки й інтерфейс, що включає дошку, послідовність зіграних кроків,

виділений останній крок комп'ютера, ідентифікатор поточного гравця, підсвічення короля під шахом, якщо є. Далі аналізується поточний стан гри – чи був поставлений мат. Якщо так – гра завершена, в іншому випадку черга передається іншій стороні – білим або чорним. Залежно від типу гри (хто грає за цей колір, людина або ШІ) буде очікуватись хід людини, аналізуватись й так далі до завершення гри. Алгоритм циклічно змінює ходи гравців й описує всі розроблені типи гри – ШІ проти людини, ШІ проти ШІ, людина з людиною. В останньому випадку буде використаний урізаний варіант, що пропускає хід комп'ютера (див.рис.4).

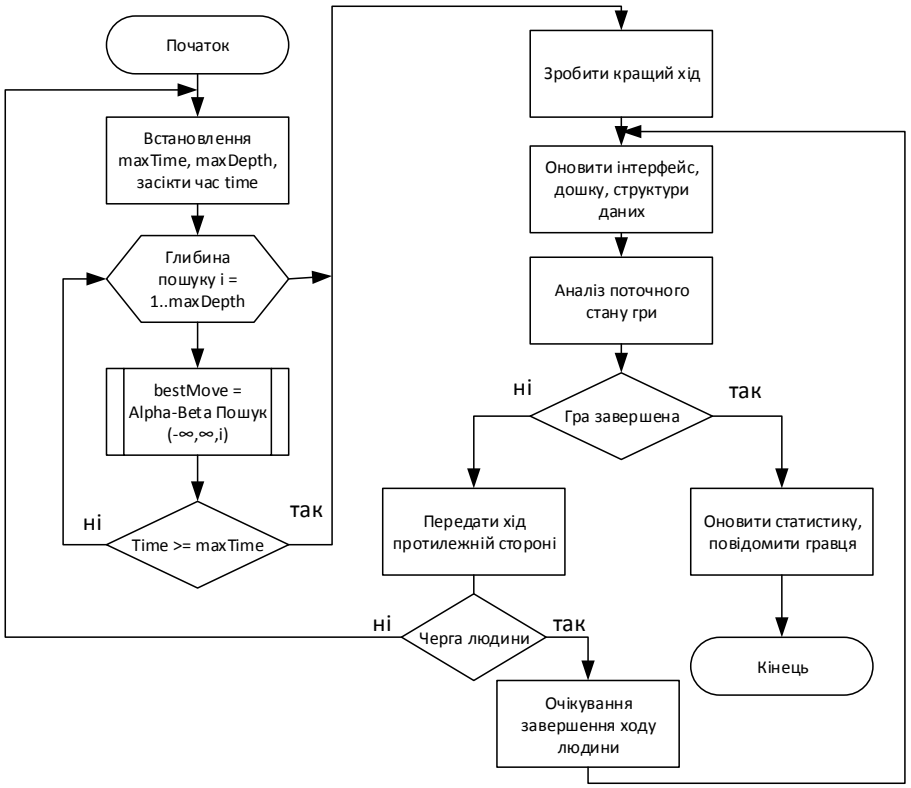


Рисунок 4 – Загальний алгоритм ігрового процесу для ШІ

Засоби оптимізації алгоритму

З метою оптимізації порядку ходів авторами запропоновано реалізувати:

а) використання евристики нульового ходу в надії відразу отримати відсікання;

б) результат із хешу PvTable – кращий хід для поточної позиції, якщо є;

в) розглядання відсортованих за принципом MVV/LVA взятій;

г) евристика вбивці – розглядання кращих ходів для поточного рівня пошуку (для сусіднього вузла);

д) евристика історії – розглядання ходів, відсортованих по статистиці їх використання й впливу на оцінку раніше;

е) після цього розглядаються всі інші ходи.

Алгоритм альфа-бета пошуку кращого ходу рекурсивно обробляє дерево можливих ходів (стратегій гри) на задану глибину depth, з кожним кроком зменшуючи її. Коли глибина дорівнює 0, викликається функція пошуку спокою з більшою глибиною. Алгоритм функції дуже схожий на алгоритм альфа-бета, але розглядає лише взяття, щоб згладити ефект горизонту. Після цього програма намагається отримати відсікання за бетою, використавши евристику нульового ходу. Якщо евристика спрацьовує, можна повернути бету, якщо ні – генеруємо та розглядаємо ходи, відсортувавши їх належним чином, описаним в попередньому пункті. Зробивши певний хід (на рівні бітбордів, а не інтерфейсу), виконуємо перебір з нульовим вікном. Якщо еврістика не спрацювала, виконаємо повний пошук, але вже в скороченому діапазоні (по принципу NegaScout). Після чого можна оновити альфу, якщо вона покращилась, зберігти відомості про кращий хід й перейти до інших ходів поточного рівня дерева. Після огляду кожного з них повертаємо отриману оцінку – альфу. Знайдена оцінка відповідатиме кращому ходу, сам хід буде збережений в таблиці PVTable з відповідним хешем позиції [18]. Блок-схема алгоритму показана на рисунку 5.

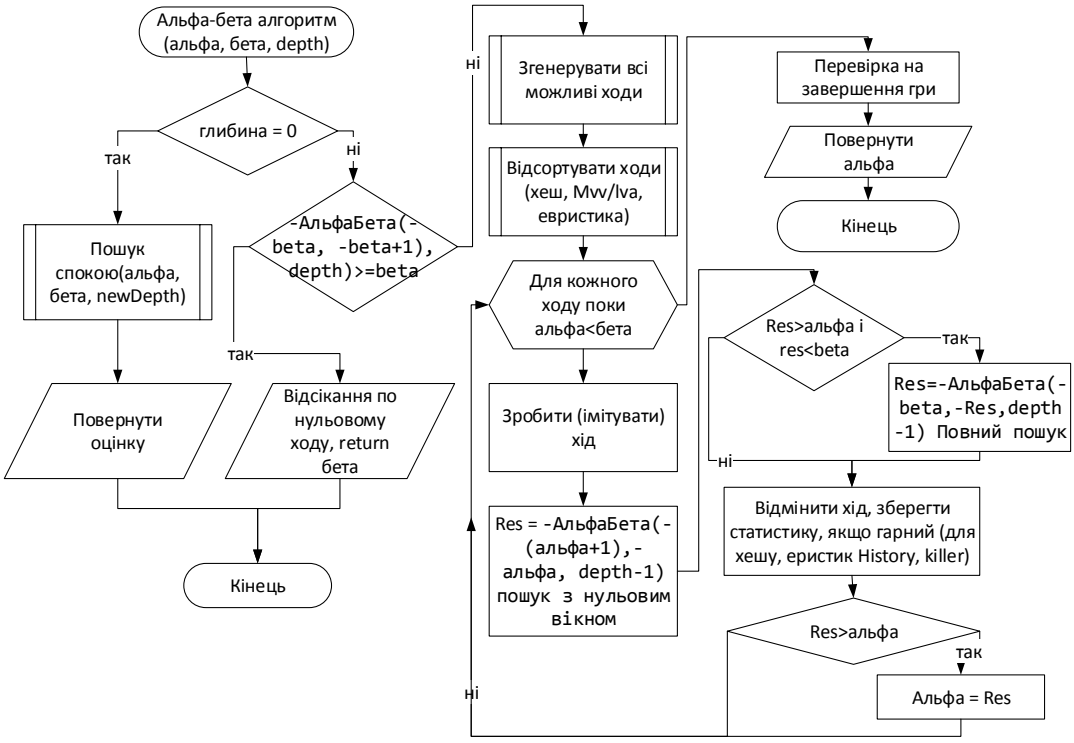


Рисунок 5 – Алгоритм альфа-бета відсікання

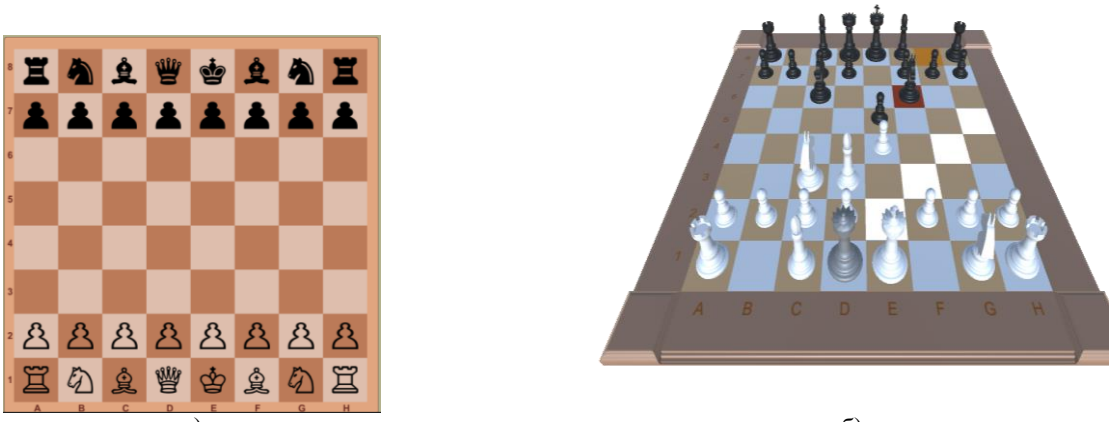
Результати експериментів

Розроблений програмний продукт розрахований на використання в операційній системі Windows XP SP2 і вище. Додаткові системні вимоги:

а) графічна карта: DX9 (модель шейдера 3.0) або DX11 з підтримкою можливостей рівня 9.3;

б) центральний процесор: підтримка набору інструкцій SSE2.

В програмі передбачено використання двовимірного аби тривимірного ігрового шахового поля (рис. 6).



а) б)

Рисунок 6 – Шахова дошка у двомірному (а) та тривимірному (б) вигляді

Реалізовано чотири варіанти гри:

а) за білих;

б) за чорних;

в) ШІ проти ШІ – штучний інтелект грає і за білих, і за чорних, користувач може лише дивитись на гру;

г) проти людини – ходи за обидві сторони робить людина.

Були проведені наступні види тестування:

а) тестування вкладу окремих евристик в роботу штучного інтелекту;

б) тестування з різними параметрами (глибина дерева, обмеження часу пошуку, бонуси та штрафи за розташування фігур для отримання позиційної оцінки тощо) з метою пошуку оптимальних значень;

в) тестування штучного інтелекту при грі з людиною та іншими програмами;

г) тестування поведінки програми у заданих ситуаціях (класичні шахові задачі, тестові задачі на фігурну активність та ефект горизонту).

В процесі тестування корегувалися матриці позиційних оцінок, максимальний час та глибина пошуку, використані евристики й їх порядок. Наприклад, евристика Aspiration Search поруч з іншими використаними евристиками не мала вагомого внеску, тому була опущена в кінцевому алгоритмі програми. Результати тестування впливу деяких евристик на швидкість перебору варіантів виявились такими: евристика Null Move дала прискорення на 56%, NegaScout – на 38%, PVS – на 74%, Killer heuristic – на 14%, History heuristic – на 17%.

При тестуванні виконувалась оцінка глибини пошуку, якої досягає алгоритм за певний час, затрачений для обробки заданої глибини дерева час (при скасуванні обмежень на глибину та час відповідно), кількість оглянутих вузлів тощо. В результаті розробки було поставлено обмеження часу на хід від 1 до 3 секунд залежно від рівня складності. Максимальна глибина дерева також обмежується залежно від рівня – від 2 до 20 (10 рівнів складності). Як правило, загальний пошук альфа-бета не виходить за глибину 8 ходів при максимальній кількості фігур на полі, але «небезпечні» ходи (зокрема, взяття фігур) розглядаються набагато (максимум – у 3 рази) глибше до деякої стабілізації ситуації.

На фінальному етапі розробки були проведені 25 ігор з 5 середніми гравцями на максимальних рівнях штучного інтелекту (8-10), на яких комп'ютер одержав перемогу. На низьких рівнях складності (1-5) людина, як правило, перемагає.

Для тестування використовувалась база тестових наборів для шахових движків, зараз відома як wacnew.epd, яка існує і доповнюється ще з 70-х років XX століття. Використана база тестів складається з 300 позицій у форматі FEN та вказаних кращих ходів для них, розроблена програма успішно впоралася з тестами.

Крім того, виконувалось порівняння створеного штучного інтелекту – програми myChessAI- з іншими програмами з максимальним рівнем штучного інтелекту. Створена авторами програма перемогла AsisChess, SparkChess, при грі з одним з найсильніших движків Rybka на максимальному рівні складності розроблена програма зазнала поразки на 34 кроці.

Аналіз середнього часу ходу (рис.7) показав, що за часовими характеристиками розроблена гра не поступається аналогам, при цьому якість гри, як показано на рис.9, приблизно така ж, як і в програмах- аналогах.

Як видно, більшість ходів програма робить досить розумно й не допускає грубих помилок у грі. У подальшому слід звернути увагу на кроки, які Rybka кваліфікувала як небезпечні й помилкові, щоб переглянути окремі стратегії гри штучного інтелекту.

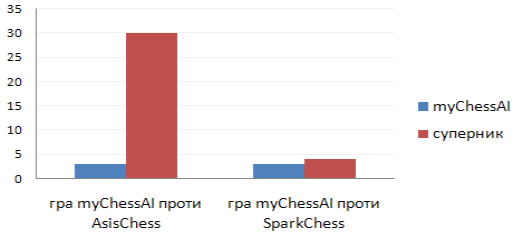


Рисунок 7 – Порівняння середнього часу ходу (в секундах)

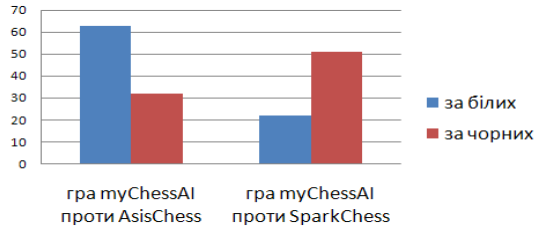


Рисунок 8 – Порівняння середньої кількості ходів гри

Рисунок 9 – Доля кроків (в %) для різних програм

Тестування ігрового додатку відбувалось у наступних середовищах:

а) ОС Windows 10, процесор Intel Pentium CPU N3700 1.60 GHz, відеокарта Intel HD Graphics;

б) ОС Windows 7, процесор Intel Core i3-7100 3.2 GHz, відеокарта Nvidia GeForce 780M;

в) ОС Windows 7, процесор Intel Pentium G4560 3.5 GHz, відеокарта ATI Radeon HD 3650;

г) ОС Windows 10, Intel Pentium N3710 2.6 GHz, відеокарта Intel HD Graphics 405.

Висновки

Аналіз отриманих результатів показав, що програму слід вдосконалювати, підвищуючи рівень її штучного інтелекту до рівня кращих шахових програм. Зокрема, використання готових баз даних (бібліотек) дебютів та ендшпілей, що застосовуються в багатьох сучасних комп'ютерних шахах, значно вплине на результат змагання з іншими програмами [19]. Інший варіант – не використовувати існуючі бази, а створювати власні, змусивши движок навчатись на проведених іграх. Надалі планується розробка підсистеми навчання штучного інтелекту з використанням штучної нейронної мережі

.

Список літератури

1. Подуфалов М. Використання штучного інтелекту в розвиваючих комп'ютерних іграх для дітей шкільного віку [Електронний ресурс] / М. Подуфалов // Матеріали XLVI науково-технічної конференції підрозділів ВНТУ, Вінниця, 22-24 Березня 2017 Р. - Електрон. текст. дані. - 2017. - Режим доступу : https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2017/paper/view/2134.
2. Шампандар А. Дж. Искусственный интеллект в компьютерных играх. Вильямс, 2007, 761 стр.
3. A brief history of computing by Jack Copeland. The Turing archive for the history of computing. http://www.alanturing.net/index.htm
4. Каррыев Б. С. ИТ-революция: хроники 1904–2008 гг. История развития информационно-коммуникационных технологий за сто лет: телевидение, компьютеры, телефон, интернет [электронная книга]. SIBIS, 2009
5. Опис системи deep blue: https://sjeng.org/ftp/deepblue.pdf
6. Алимов А. А. Искусственный интеллект в компьютерных играх. Многоуровневое планирование и реактивное поведение агентов / А. А. Алимов, О. А. Шабалина // Известия Волгоградского государственного технического университета. – 2011. – №10 (103). – С. 1–5.
7. Shannon C.E. Programming a computer for playing chess. Philosophical Magazine, Ser.7, Vol. 41, No. 314 - March 1950. http://vision.unipv.it/IA1/aa2009-2010/ProgrammingaComputerforPlayingChess.pdf
8. Ботвинник М. М. Шах двадцатому веку / М. М. Ботвинник. – М.: Зебра Е, 2010. – 494 с.
9. Лысенко А. В. Оценка позиции. Компьютерные шахматы / А.В.Лысенко, Е. Я.Гик. – М.: ФиС, 1990. – 176 с.
10. Джонс М. Т. Программирование искусственного интеллекта в приложениях / М. Тим Джонс; Пер. с англ. – М.: ДМК Пресс, 2015. – 313 с.
11. Levy D., Claude S. Computer Chess Compendium / D. Levy, S. Claude. – Ishi Press, 2009. – 452 с.
12. Levy D. All about Chess and Computers: Containing the Complete Works, Chess and Computers / D. Levy, M. Newborn. – Computer Science Press, 1982._– 146 с.
13. Волчок А. С. Тактика и стратегия шахмат / А. С. Волчок. – К.: Здоровье, 1985. – 176 с.
14. Computer Chess Rating List 40/4, December, 6, 2017, http://www.computerchess.org.uk/ccrl/404FRC/
15. Heinz E. Scalable Search in Computer Chess: Algorithmic Enhancements and Experiments at High Search Depths / E. Heinz. – Vieweg, Verlag, 1999. – 270 с.
16. An AI Chess Engine [Електронний ресурс]. – Режим доступу: https://web.stanford.edu/class/cs221/sample-projects/arijitb-aparnak-chess.pdf
17. Does Deep-Blue use AI? [Електронний ресурс]. – Режим доступу: https://www.aaai.org/Papers/Workshops/1997/WS-97-04/WS97-04-001.pdf
18. Exploring speed and intelligence [Електронний ресурс]. – Режим доступу: https://www2.cs.arizona.edu/news/honors/StevensP.pdf
19. Ананд В. Энциклопедия шахматных дебютов / В. Ананд. – Н.: Кипр, 1993. – 178 с.

References

1. Podufalov, M. (2017) "Using of artificial intelligence in learning computer games for school- age children" ["Vykorystannya shtuchnoho intelektu v rozvyvayuchykh komp'yuternykh ihrakh dlya ditey shkil′noho viku"], available at: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2017/paper/view/2134.
2. Shampandar, A.J. (2007), "Artificial intelligence in computer games" ["Iskusstvennyy intellekt v komp'yuternykh igrakh]? Williams, Moscow, 761 p.
3. A brief history of computing by Jack Copeland. The Turing archive for the history of computing. available at: http://www.alanturing.net/index.htm
4. Karryev, B.S. (2009) "The IT Revolution: Chronicles of 1904-2008. The history of the development of information and communication technologies for a hundred years: television, computers, telephone, Internet" ["IT-revolyutsiya: khroniki 1904–2008 gg. Istoriya razvitiya informatsionno-kommunikatsionnykh tekhnologiy za sto let: televideniye, komp'yutery, telefon, internet (elektronnaya kniga)], SIBIS.
5. Specification of Deep Blue system, available at: https://sjeng.org/ftp/deepblue.pdf
6. Alimov, A.A., Shabalina, O.A. (2011), Алимов А. А. "Artificial intelligence in computer games. Multilevel planning and reactive behavior of agents: ["Iskusstvennyy intellekt v komp'yuternykh igrakh. Mnogourovnevoye planirovaniye i reaktivnoye povedeniye agentov"], Bulletin of the Volgograd State Technical University, No 10 (103), pp.1-5.
7. Shannon, C.E. (1950) "Programming a computer for playing chess", Philosophical Magazine, Ser.7, Vol. 41, No. 314 - March 1950, available at: http://vision.unipv.it/IA1/aa2009-2010/ProgrammingaComputerforPlayingChess.pdf.
8. Botvinnik, M.M. (2010) "Shakh the twentieth century" ["Shakh dvadtsatomu veku"], Mjscow, Zebra, 494 p.
9. Lysenko, A.V., Gik, E. Ya. (1990) "Evaluation of the position. Computer chess" ["Otsenka pozitsii.

Komp'yuternyye shakhmaty"], Moscow, 176 p.
10. Jones, M.T. (2015) "Programming of artificial intelligence in applications" ["Programmirovaniye iskusstvennogo intellekta v prilozheniyakh"], Moscow, DMK Press, 313 p.
11. Levy, D., Claude S. (2009) "Computer Chess Compendium", Ishi Press, 452 p.
12. Levy, D., Newborn, M. (1982) "All about Chess and Computers: Containing the Complete Works, Chess and Computers", Computer Science Press, 146 c.
13. Volchok, A.S. (1985), "The tactics and strategy of chess" ["Taktika i strategiya shakhmat"], Kiev, Zdorovje, 176 p.
14. Computer Chess Rating List 40/4, December, 6, 2017, available at: http://www.computerchess.org.uk/ccrl/404FRC/
15. Heinz, E. (1999), "Scalable Search in Computer Chess: Algorithmic Enhancements and Experiments at High Search Depths", Vieweg, Verlag, 270 c.
16. An AI Chess Engine, available at: https://web.stanford.edu/class/cs221/sample-projects/arijitb-aparnak-chess.pdf
17. Does Deep-Blue use AI?, available at: https://www.aaai.org/Papers/Workshops/1997/WS-97-04/WS97-04-001.pdf
18. Exploring speed and intelligence, available at: https://www2.cs.arizona.edu/news/honors/StevensP.pdf
19. Anand, V. (1993), "Encyclopedia of chess debuts" ["Entsiklopediya shakhmatnykh debyutov"], Nikosia, Cyprus, 178 p.

Надійшла до редакції 15.10.2017

Е.С. ГЛИНСКАЯ, Н.С. КОСТЮКОВА
Донецкий национальный технический университет (г.Покровск, Украина)

РЕАЛИЗАЦИЯ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА ДЛЯ ИГРЫ В ШАХМАТЫ

В работе предлагается реализация искусственного интеллекта для игры в шахматы на основе алгоритма альфа-бета отсечения с модификациями, анализируется эффективность предложенных модификаций в сравнении с популярными шахматными программами.

Ключевые слова: искусственный интеллект, игра в шахматы, стратегическая игра, альфа-бета алгоритм, оценка позиции, MVV / LVA, BITBOARD.

K.S. HLYNSKA, N.S. KOSTYUKOVA
Donetsk National Technical University (Ukraine)

IMPLEMENTATION OF ARTIFICIAL INTELLIGENCE FOR CHESS GAME

The proposed chess engine implementation is oriented to improve artificial intelligence of strategic computer games by increasing the strength of its moves and reducing the time required. The developed algorithm is based on existing techniques of chess programs realization such as the construction of a decision tree, alpha-beta pruning, and their modifications. The combination of various modifications was considered to select the most optimal one. The problems of modern chess games were taken into account when designing an application. Various approaches to the development of chess programs, existing analogues and methods used to improve the artificial intelligence of chess were examined before implementation. The main idea of the algorithm proposed is based on using the classical knowledge of the best chess game strategies together with heuristics of Killer, History, the Null move, NegaScout, PVS and some others. The contribution of each heuristic and modification to the algorithm work was tested before leaving or removing it from the final program. To optimize the operation time of the algorithm, magic bitboards, advanced move and attacks generation scheme, MVV/LVA move sorting methods were used. Additional ways to improve the level of game artificial intelligence, further steps in the development of the chess program have been established in accordance with the latest achievements in this field. For testing the algorithm, the official tables of the best moves for the given positions were used, as well as games against a person and other chess programs, classical chess tasks for setting a mat in a given number of moves. For comparison with existing programs, a moving time, a frequency of the best move selection, the winning frequency were analyzed. The developed program showed good results of the games, although it was inferior to the best similar applications.

The proposed algorithm was implemented in a full-fledged chess program. The developed program can compete with the person and the majority of modern similar programs not yielding to them in the provided functions and interface. The implemented algorithm is aimed at optimizing the artificial intelligence of the computer player and can be easily modified to improve any strategic game.

Key words: artificial intelligence, alpha-beta pruning, chess game, strategic game, MVV/LVA, Bitboard.