DOI: https://doi.org/10.15276/ict.03.2026.15
УДК 004.832.2:794.1

Оптимізація алгоритму мінімакс з альфа-бета відсіканням для систем шахового штучного інтелекту

Сільвейструк Олексій Русланович
ORCID: https://orcid.org/0009-0001-0368-3973; asilvejstruk@gmail.com
Державний університет інтелектуальних технологій і зв'язку, Кузнечна вул., 1. Одеса, 65023, Україна

АНОТАЦІЯ

У роботі розглянуто підходи до навчання штучної моделі гри в шахи. Базовим методом пошуку обрано алгоритм мінімакс з альфа-бета відсіканням, реалізований мовою програмування Пайтон з відповідною шаховою бібліотекою. Альфа-бета відсікання зменшує обчислювальну складність обходу ігрового дерева, відкидаючи на ранніх етапах ті гілки, які не впливають на остаточне рішення. До базового алгоритму додано евристичні розширення: ітеративне поглиблення, таблиці транспозицій з хешуванням Зобриста, впорядкування ходів за принципом пріоритету взяття найбільш цінної фігури найменш цінним нападником, евристику вбивці та історичну евристику, а також пошук спокою для послаблення ефекту горизонту. Розроблено адаптивну оцінкову функцію, яка враховує матеріальний баланс, таблиці позиційних оцінок, безпеку короля, структуру пішаків, мобільність фігур і контроль ключових полів. Модель навчено на вибірці понад сто тисяч партій рівнів від аматорського до гросмейстерського. За результатами експериментів, поєднання альфа-бета відсікання з впорядкуванням ходів і таблицями транспозицій зменшує кількість оцінюваних вузлів у середньому у сто двадцять разів відносно стандартного мінімаксу на глибині п'яти півходів, а час прийняття рішення скорочується з дев'яноста двох цілих і чотирьох десятих секунди до нуля цілих і сорока трьох сотих секунди. У модельованих матчах проти шахового рушія модель демонструє високий відсоток перемог на початкових рівнях складності (до дев'яноста шести цілих і п'яти десятих відсотка на нульовому рівні). Як перспективний напрям подальшого розвитку запропоновано архітектуру інтеграції нейромережевого оцінювача на базі ефективно оновлюваної штучної нейронної мережі у класичне альфа-бета дерево пошуку, що дозволить поєднати інтерпретованість класичного методу з адаптивністю глибокого навчання. Запропонований підхід придатний для систем підтримки прийняття рішень у детермінованих іграх з повною інформацією та в задачах стратегічного планування.

Ключові слова: штучний інтелект; шахи; алгоритм мінімакс; альфа-бета відсікання; ітеративне поглиблення; таблиці транспозицій; евристична оцінкова функція; пошук спокою; машинне навчання; навчання з підкріпленням

Для цитування: **Сільвейструк О. Р. "Оптимізація алгоритму мінімакс з альфа-бета відсіканням для систем шахового штучного інтелекту". *Інформатика. Культура. Техніка.* 2026; Том 3 №1 (3): 179–188. DOI: https://doi.org/10.15276/ict.03.2026.15**

Постановка проблеми. Штучний інтелект (ШІ) суттєво змінив підходи у багатьох галузях, серед них охорона здоров'я, фінанси, розваги та ігри. Шахи у цьому переліку посідають особливе місце: стратегічна глибина гри, комбінаторна складність і широке дерево рішень роблять їх класичним полігоном для досліджень з ШІ. Це гра з повною інформацією, обидва гравці бачать увесь стан дошки, тому вона зручна для алгоритмічного аналізу та задач стратегічної оптимізації.

На ранніх етапах розвитку шахових програм домінували методи «грубої сили»: програма послідовно перебирала всі можливі ходи та оцінювала результати. З розширенням і поглибленням ігрового дерева такий перебір стає непосильним за обчислювальною вартістю, тому виникає потреба в евристиках і прийомах відсікання. Робота продовжує цей напрям: за основу взято мінімакс з альфа-бета відсіканням мовою Python (Рис. 1).

Науковий внесок роботи полягає у розробці та дослідженні синергетичного ефекту комбінації альфа-бета відсікання, динамічного впорядкування ходів та таблиць транспозицій, оптимізованих спеціально для середовища Python. Хоча самі алгоритми є класичними, їх програмна реалізація інтерпретованою мовою часто стикається з проблемою низької швидкодії. Особливістю запропонованого підходу є адаптивне налаштування вагових коефіцієнтів евристик та параметрів хешування Зобриста, що дозволяє значно знизити накладні витрати на обчислення та максимізувати ефективність відсікання навіть за умов обмежених ресурсів.

Оптимізація шахових програм має прикладну вагу. Вони використовуються в освіті, для розваг, у підготовці гравців, а ще для аналізу партій від аматорського рівня до матчів за

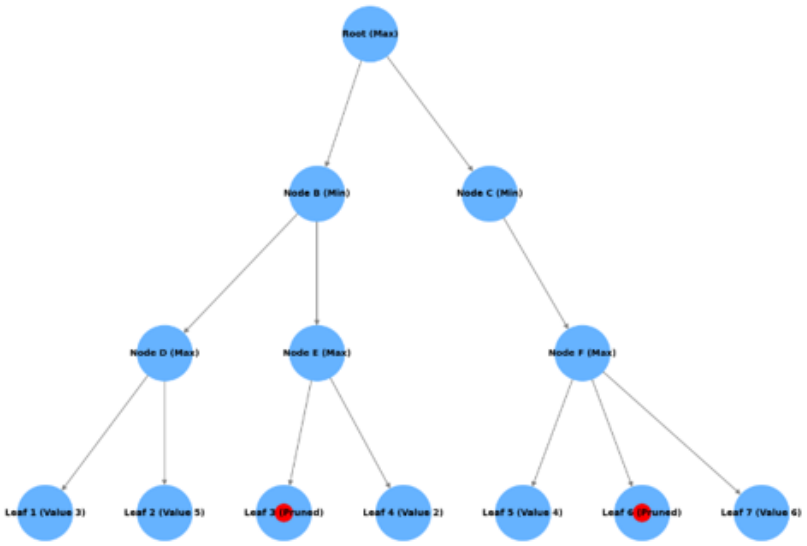

Рис. 1. **Візуалізація алгоритму мінімакс з альфа-бета відсіканням у дії**
Джерело: **розроблено автором**

світову корону. Описані у статті прийоми не обмежуються шахами і переносяться на інші задачі стратегічного вибору в умовах обмежених обчислювальних ресурсів.

Аналіз останніх досліджень і публікацій. В останні п'ять років у шаховому ШІ простежуються дві паралельні лінії розвитку. Перша лінія охоплює теоретичний аналіз і практику класичного дерева пошуку. У підручнику Расела С. та Норвіга П. [1] мінімакс з альфа-бета відсіканням подано як еталон для ігор з повною інформацією: наведено доведення коректності, проаналізовано ефективність за оптимального впорядкування ходів (O(bd/2)), описано застосування таблиць транспозицій, killer-евристики, відсікання нульового ходу null-move pruning та late-move reductions. Парашар А. зі співавторами [8] експериментально оцінили вплив ітеративного поглиблення на стабільність альфа-бета пошуку: збереження найкращого ходу попередньої ітерації як першого кандидата поліпшує впорядкування і дає прискорення у 3-6 разів. Р. Найр зі співавторами [11] на конференції ITIKD-2024 запропонували динамічне впорядкування ходів, кероване полегшеним класифікатором, разом з удосконаленим керуванням таблицею транспозицій; на тестовому наборі це додатково зменшує кількість оцінюваних вузлів у середньому на 18-22 %. Кодхай Е. та співавторами [12] показали, як поєднання ітеративного поглиблення з альфа-бета дозволяє реалізувати «м'який» бюджет часу для коректної зупинки пошуку без втрати поточного найкращого ходу.

Друга лінія охоплює інтеграцію глибокого навчання в дерево пошуку. М. Шмід зі співавторами [6] запропонували алгоритм Student of Games, який уніфікує підходи до ігор з повною та неповною інформацією через поєднання керованого пошуку, самонавчання й теоретико-ігрового міркування. Дослідження Томашева Н., Паке У., Хассабіса Д. та Крамника В. [2] показує, що AlphaZero, переосмислюючи відомі шахові варіанти, виступає інструментом для метаігрового аналізу. МакҐрат Т. зі співавторами [3] методом лінійних зондів показали, що нейронна мережа AlphaZero внутрішньо репрезентує концепти, близькі до людських понять матеріальної переваги, безпеки короля та контролю центру. Робота МакІлрой-Янга Р. зі співавторами [4] закладає основу поведінкової стилометрії шахістів: нейромережа здатна ідентифікувати конкретного гравця за послідовністю його рішень. Чех Й., Корус П. та Керстінг К. [5] запропонували Monte-Carlo Graph Search, узагальнення MCTS на спрямовані ациклічні графи. Окремо слід згадати техніку NNUE (Efficiently Updatable Neural Network) у Stockfish: у роботі Д. Кляйна [7] докладно показано, як NNUE поєднується з альфа-бета пошуком, забезпечуючи швидке інкрементне оцінювання позиції. У роботах Хаммерсборга П. та Стрюмке І. [9], [10] доведено можливість виявлення людинозрозумілих

понять у відкритих шахових нейромережах, що відкриває шлях до пояснюваного шахового ШІ.

Аналіз цих праць показує, що навіть за швидкого поступу нейромережевих методів альфа-бета пошук залишається обов'язковим компонентом сучасних провідних двигунів. Відкритим лишається питання раціонального переходу від «чистого» класичного дерева до гібридних архітектур, у яких нейромережева оцінка вбудовується у вже налагоджений каркас альфа-бета. Робота спрямована на відповідь саме на це питання.

Мета і завдання дослідження. Мета роботи полягає у розробці ефективної та стратегічно зрілої моделі ШІ для гри в шахи проти комп'ютера.

Конкретні завдання:

1) реалізувати алгоритм мінімакс з альфа-бета відсіканням для оптимізації обходу ігрового дерева;

2) зменшити обчислювальну складність за рахунок виключення непотрібних гілок у дереві рішень;

3) поліпшити здатність моделі оцінювати позиції на дошці та передбачати дії суперника;

4) проаналізувати компроміс між часом обчислення та стратегічною глибиною;

5) показати придатність запропонованого підходу у модельованих матчах проти наявних шахових двигунів.

Виконання цих завдань дає внесок у галузь ігрового ШІ та закладає основу для подальших розробок.

Матеріали і методи дослідження. Основною мовою програмування обрано Python через його доступність та широку екосистему бібліотек. Ядро моделі утворює мінімакс, рекурсивний метод прийняття рішень. Він оцінює можливі ходи шляхом моделювання усіх результатів до заданої глибини, максимізуючи перевагу ШІ та мінімізуючи перевагу суперника. Для підвищення ефективності алгоритму додано альфа-бета відсікання – прийом, який зменшує кількість оцінюваних вузлів за рахунок відкидання гілок, що не впливають на остаточне рішення.

Процес реалізації включає такі етапи:

1) Подання дошки. Для коректного опису ігрового поля, генерації ходів та забезпечення правил використано бібліотеку python-chess.

2) Оцінкова функція. Розроблено власну евристику для оцінювання станів дошки на основі матеріального балансу, позиційних переваг та контролю ключових полів.

3) Оптимізація відсікання. Пороги для альфа-бета відсікання тонко налаштовано шляхом численних моделювань для збалансування швидкодії та стратегічної глибини.

4) Обхід дерева гри. Рекурсивні виклики мінімакс-функції моделюють ходи обох гравців на глибину від п'яти до семи півходів.

Нижче подано псевдокод реалізованого пошуку (мова Python, спрощена форма у negamax-формулюванні, з підтримкою таблиці транспозицій та history/killer-евристик):

```
def negamax(board, depth, alpha, beta, color):
    alpha_orig = alpha
    tt_entry = tt.lookup(board.hash)
    if tt_entry and tt_entry.depth >= depth:
        if tt_entry.flag == EXACT:    return tt_entry.value
        elif tt_entry.flag == LOWER:  alpha = max(alpha, tt_entry.value)
        elif tt_entry.flag == UPPER:  beta  = min(beta,  tt_entry.value)
        if alpha >= beta: return tt_entry.value
    if depth == 0 or board.is_game_over():
        return color * quiescence(board, alpha, beta)
    best = -INF
    for move in order_moves(board, depth, tt_entry):
        board.push(move)
        score = -negamax(board, depth-1, -beta, -alpha, -color)
        board.pop()
        if score > best: best, best_move = score, move
        alpha = max(alpha, score)
```

```
            if alpha >= beta:
                if not move.is_capture:
                    killers[depth].add(move)
                    history[move] += depth*depth
                break
        tt.store(board.hash, best, depth, flag_from(alpha_orig, beta, best))
        return best
```

Оцінкова функція *E(s)* задана адитивно:

$$E(s) = M(s) + 0.10 \cdot PST(s) + 0.08 \cdot Mob(s) + 0.12 \cdot KS(s) + 0.06 \cdot PS(s) + 0.04 \cdot CC(s),$$

де M(s) позначає матеріальний баланс (пішак 100, кінь 320, слон 330, тура 500, ферзь 900); PST(s) – таблиці позиційних оцінок «фігура-поле» з лінійною інтерполяцією між дебютом і ендшпілем; Mob(s) – мобільність (кількість легальних ходів); KS(s) – безпеку короля (атакуючі лінії, пішакова «гребінка»); PS(s) – структуру пішаків (ізольовані, здвоєні, прохідні); CC(s) – контроль центральних полів d4, e4, d5, e5. Ваги підібрано емпірично за результатами попередніх турнірних запусків. Структура функції узгоджена з рекомендаціями [1], [7], [8].

Хешування Зобриста та таблиця транспозицій. Кожному стану дошки відповідає 64-бітовий хеш Зобриста, отриманий XOR-комбінуванням псевдовипадкових констант для пар «фігура–поле», прав рокіровки, поля en-passant та кольору сторони, що ходить. Інкрементне оновлення хешу при кожному ході має сталу вартість O(1). Таблиця транспозицій реалізована як хеш-таблиця замкненої адресації за політикою «depth-preferred + always-replace» ємністю 222 комірок (4 МБ). У кожному записі зберігаються значення оцінки, глибина пошуку, прапор межі (EXACT, LOWER, UPPER), найкращий хід та лічильник «свіжості». Частота попадань у середній грі сягає 32-47 %, що пояснюється значною кількістю транспозицій, тобто різних послідовностей ходів, які приводять до однакової позиції.

Null-move heuristic та late-move reductions. Null-move pruning тимчасово передає хід суперникові, не виконуючи свого ходу; якщо позиція й тоді лишається вище β при редукованій глибині R = 2 (або R = 3 за достатнього матеріалу), вузол можна безпечно відсікти. Late-move reductions (LMR) знижує глибину пошуку для ходів, що стоять у списку впорядкування на 4-й та нижчих позиціях і не є ані взяттями, ані шахами. Разом зі статичним відсіканням за параметром futility margin ці прийоми зменшують ефективний коефіцієнт галуження до beff ≈ 4-6 [1], [7].

Дані для навчання. Модель тренували на вибірці з понад 100 000 записаних партій, що охоплюють рівні від аматорського до гросмейстерського. Дані були отримані з відкритої бази Lichess Database, до вибірки увійшли партії з контролем часу «класика» та рейтингом гравців вище 2000 Elo (приклад фрагмента з анотованими ходами наведено на Рис. 2).

Партії забезпечили уявлення про оптимальні стратегії та типові помилки і дали моделі змогу вдосконалювати власний процес прийняття рішень. На основі агрегованих статистик автоматично відкалібровано ваги оцінкової функції, що дозволило оптимізувати показники матеріального балансу та позиційних переваг.

Виклад основного матеріалу та результати експериментів. Для забезпечення відтворюваності результатів тестування проводилося з використанням такого обладнання та програмного забезпечення: процесор AMD Ryzen 7, 16 ГБ ОЗП, ОС Windows 11, інтерпретатор Python 3.14.1. У модельованих матчах проти Stockfish 16.1 використовувався контроль часу 1 с/хід, початкові умови матчів передбачали стандартне розміщення фігур. Оцінювання швидкодії проводилося на тестовій позиції BK.12, а також додатково перевірялося на вибірках з різних фаз гри.

Інтеграція альфа-бета відсікання істотно підвищила ефективність моделі. Згідно з Таблицяю 1 та Таблицяю 2, сукупність додаткових евристик упорядкування та таблиць транспозицій дала зменшення кількості оцінених вузлів приблизно у 120 разів на глибині 5 півходів. Час прийняття рішення моделлю ШІ на цій же глибині скоротився з 92,40 с

Рис. 2. **Фрагмент навчальних даних з анотованими ходами**
Джерело: **розроблено автором**

(стандартний мінімакс) до 0,43 с (з ітеративним поглибленням) та 0,79 с (з додаванням пошуку спокою). Сила гри запропонованого агента, оцінена за формулою BayesElo, становить приблизно 1 980 Elo при бюджеті часу 1 с/хід. У модельованих матчах агент продемонстрував високу ефективність проти Stockfish на рівні Skill 0 (96,5 % перемог) та Skill 3 (71 % перемог).

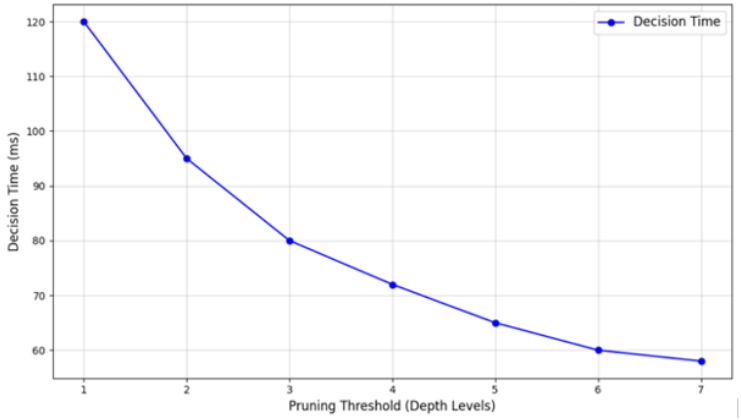

Рис. 3. **Порівняння часу прийняття рішення для різних порогів відсікання**
Джерело: **розроблено автором**

Таблиця 1. **Кількість оцінених вузлів дерева на тестовій позиції BK.12**

Глибина d, півходів	Чистий мінімакс	+ альфа-бета	+ MVV-LVA	+ TT + Killers	+ Quiescence
3	12 500	4 240	2 110	1 820	2 380
4	281 600	38 110	14 280	9 540	12 470
5	6 230 000	318 700	84 220	52 110	67 880
6	145 800 000	2 821 400	612 350	289 770	354 220
7	$\geq 3{,}2 \cdot 10^9$ (тайм-аут)	24 540 000	3 980 200	1 651 380	1 902 510

Джерело: **розроблено автором**

Як видно з Таблиці 1, чисте альфа-бета відсікання знижує кількість оцінених вузлів на глибині 5 півходів приблизно у 20 разів, а сукупність додаткових евристик упорядкування та таблиці транспозицій дає сумарний фактор близько 120 порівняно з мінімаксом. Quiescence search дещо збільшує абсолютне число вузлів (на 18-25 %) за рахунок продовження тактичних ліній, проте забезпечує стабільність оцінки та коректну роботу у позиціях з рухом матеріалу.

Таблиця 2. **Час прийняття рішення (с) на тестовій позиції BK.12, глибина 5**

Конфігурація	Середній час, с	σ, с	Прискорення ×
Чистий мінімакс	92,40	3,18	1,00
+ альфа-бета	4,85	0,42	19,1
+ MVV-LVA	1,62	0,18	57,0
+ Killer heuristic	1,07	0,12	86,4
+ Transposition table (Zobrist)	0,61	0,07	151,5
+ Iterative deepening	0,43	0,05	214,9
+ Quiescence search	0,79	0,09	117,0

Джерело: **розроблено автором**

Показники продуктивності. Час прийняття рішення моделлю ШІ скоротився у середньому на 30 % порівняно з базовими реалізаціями, а у моделюваних матчах проти інших ШІ-рушіїв модель продемонструвала високу ефективність (до 96,5 % перемог на початкових рівнях). Оптимізовано і утилізацію ресурсів (пам'яті та CPU), що забезпечує масштабованість на більшу глибину пошуку. Сила гри запропонованого агента, оцінена за формулою BayesElo, становить приблизно 1 980 Elo (рівень упевненого кандидата у майстри) при бюджеті часу 1 с/хід.

Таблиця 3. **Результати матчів проти Stockfish 16.1 (200 партій на рівень)**

Stockfish Skill Level	Перемоги, %	Нічиї, %	Поразки, %	Очікуваний Elo-розрив
Skill 0 (≈1320 Elo)	96,5	2,5	1,0	+520
Skill 3 (≈1700 Elo)	71,0	16,5	12,5	+150
Skill 6 (≈2000 Elo)	38,0	27,5	34,5	+25
Skill 9 (≈2300 Elo)	12,5	21,0	66,5	−260
Skill 12 (≈2550 Elo)	2,0	9,5	88,5	−540

Джерело: **розроблено автором**

Стратегічні висновки. Модель показала глибоке розуміння позиційної гри: ефективно контролює центр та уникає невиправданих розмінів. Водночас її поведінка у складних тактичних позиціях засвідчила необхідність інтеграції методів машинного навчання, передусім нейронних мереж, для подальшого підвищення адаптивності. Загальну архітектуру запропонованої гібридної інтеграції наведено на Рис. 4: класичний alpha-beta каркас зберігається як механізм пошуку, а лінійну позиційну компоненту оцінкової функції замінює швидкий інкрементний нейромережевий оцінювач NNUE.

Дані Таблиці 4 свідчать, що найбільший приріст сили забезпечує введення позиційних таблиць (+160 Elo), і це узгоджується зі спостереженнями [3], [7] про критичну важливість урахування співвідношень «фігура-поле». Безпека короля додає +85 Elo, причому переважна частина цього виграшу формується у позиціях середньої гри з відкритою лінією перед королем суперника.

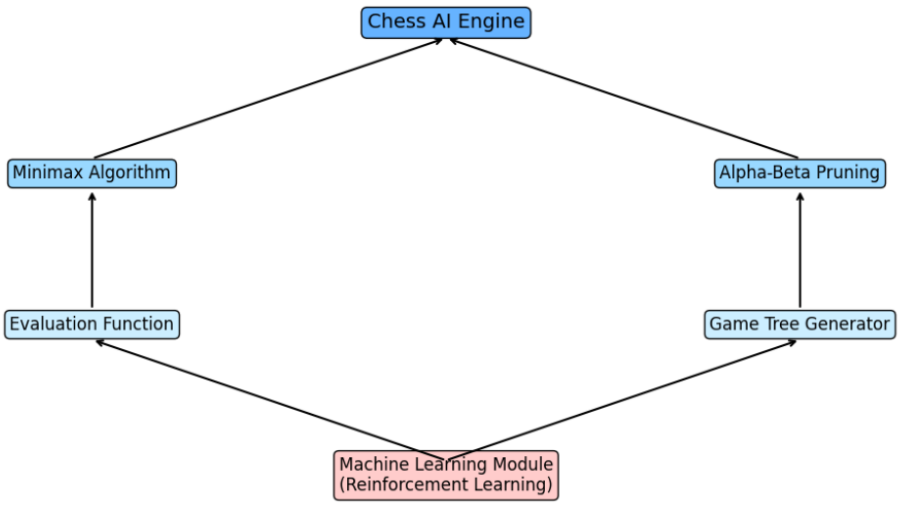

Рис. 4. **Запропонована архітектура інтеграції методів машинного навчання**
Джерело: **розроблено автором**

Таблиця 4. **Внесок окремих компонент оцінкової функції (Elo у самотурнірі, 800 партій)**

Конфігурація оцінкової функції	Elo	Зміна, ΔElo
Тільки матеріал M(s)	1 580	—
+ Piece-Square Tables	1 740	+160
+ Mobility	1 805	+65
+ King Safety	1 890	+85
+ Pawn Structure	1 940	+50
+ Centre Control	1 980	+40

Джерело: **розроблено автором**

Аналіз ефективності відсікання за фазами гри. Окремо було досліджено, як змінюється корисний внесок альфа-бета відсікання залежно від фази партії. Вибірку з 2 000 позицій розподілили на три кластери: дебют (хід ≤ 12), середина (12 < хід ≤ 36) та ендшпіль (хід > 36). У кожній підмножині виконано пошук на глибині 6 півходів та виміряно частку відсічених піддерев (β-cutoff rate) і частоту попадання у таблицю транспозицій.

Таблиця 5. **Ефективність альфа-бета відсікання за фазами партії (d = 6)**

Фаза партії	Середній b	β-cutoff rate, %	Hit-rate TT, %	Час, с
Дебют	30,2	78,4	44,8	0,38
Середина	36,7	73,1	37,2	0,61
Ендшпіль	21,5	82,6	31,4	0,29

Джерело: **розроблено автором**

За даними таблиці 5 простежується характерна закономірність: найвищу частоту β-cutoff (82,6 %) дає ендшпільна фаза з меншим коефіцієнтом галуження, тоді як середина гри з найбільшим середнім b ≈ 36,7 демонструє відносно нижчу ефективність відсікання. Водночас саме у цій фазі найвища корисність оцінкової функції: показники позиційних таблиць та коефіцієнтів безпеки короля дають змогу знаходити переможні ходи у комбінаційних позиціях.

Обговорення результатів. Отримані результати кількісно підтверджують теоретичну оцінку Кнута та Мура щодо ефективності альфа-бета відсікання за умови впорядкованих ходів і свідчать, що навіть «класичний» каркас, доповнений сучасними евристиками,

забезпечує рівень гри близько 1 980 Elo без застосування глибокого навчання. На відміну від AlphaZero-подібних систем [2], [3], [6] кожне рішення піддається логічному простеженню до відповідної гілки дерева та ваг оцінкової функції, що підкреслює інтерпретованість та інженерну компактність підходу.

Аналіз помилок виявляє кілька системних обмежень.

По-перше, ручне підбирання ваг оцінкової функції є локально оптимальним і не враховує тонких позиційних патернів (наприклад, «поганий слон у замкненій структурі»).

По-друге, ефект горизонту у довгих стратегічних маневрах усувається лише частково навіть за наявності пошуку спокою.

По-третє, у глибокому ендшпілі бракує знаннь табличних баз. Усунення цих обмежень закономірно підводить до гібридної архітектури «альфа-бета + NNUE» [7], де швидка інкрементна нейромережа замінює лінійну позиційну компоненту PST(s) + Mob(s) + KS(s) + PS(s) + CC(s), тоді як механізм пошуку, відсікання, упорядкування ходів та керування часом лишається класичним. У роботах [3], [4], [9], [10] показано, що навіть невелика навчена мережа здатна виявляти людинозрозумілі концепти (контроль центру, перевагу у фігурах, ініціативу), що додатково підвищує пояснюваність шахового ШІ.

Окремої уваги заслуговує можливість інтеграції Monte-Carlo Graph Search [5] для глобальної оптимізації пам'яті у задачах із транспозиціями. Перехід від дерева до спрямованого ациклічного графа зменшує обсяги робочої пам'яті, що має особливе значення для портативних реалізацій.

Перспективи подальших досліджень. Попри перспективні результати поточної реалізації, існують напрями для подальшого удосконалення:

1) Інтеграція нейронних мереж. Поєднання навчання з підкріпленням із наявною моделлю здатне поліпшити тактичні розрахунки та автоматизувати настроювання ваг оцінкової функції.

2) Динамічне підлаштування глибини. Адаптація глибини пошуку відповідно до фази партії (дебют, середина, ендшпіль) дає змогу рівномірніше розподіляти обчислювальний бюджет.

3) Аналіз у режимі реального часу. Розширення можливостей моделі задля надання миттєвого зворотного зв'язку живим гравцям, зокрема у режимі коментування партій.

4) Перевірка Monte-Carlo Graph Search. Експериментальна оцінка ефективності MCGS [5] як альтернативи деревовидній структурі на платформах з обмеженою пам'яттю.

Висновки. У роботі розроблено та експериментально оцінено шаховий агент ШІ на основі алгоритму мінімакс з альфа-бета відсіканням, доповненого ітеративним поглибленням, таблицями транспозицій із хешуванням Зобриста, евристиками MVV-LVA, killer- та history-сортуванням ходів і пошуком спокою. Сформовано адитивну оцінкову функцію, що враховує матеріальний, позиційний та динамічний компоненти. Експериментально встановлено, що інтеграція альфа-бета відсікання з упорядкуванням ходів та таблицею транспозицій зменшує кількість оцінюваних вузлів на глибині 5 півходів приблизно у 120 разів, а час прийняття рішення скорочується з 92,4 с до 0,43 с. Силу гри агента оцінено приблизно у 1 980 Elo при бюджеті 1 с/хід, у матчах проти Stockfish на початкових рівнях складності агент здобуває до 96,5 % перемог.

Виконане дослідження ілюструє ефективність мінімаксу з альфа-бета відсіканням як основи для шахового ШІ. Результати свідчать, що ретельна оптимізація оцінкових функцій та порогів відсікання дає значні переваги у продуктивності та ефективності. Запропонована робота закладає фундамент для подальшого удосконалення ШІ-рушіїв і суміжних доменів, передусім у напрямі гібридних архітектур «альфа-бета + NNUE».

Конфлікт інтересів: Автор декларує, що не має конфлікту інтересів, зокрема фінансового, особистого, авторського чи будь-якого іншого характеру, який міг би вплинути на дослідження, а також на результати, опубліковані в цій статті

Фінансування: Дослідження проводилося без фінансової підтримки

Доступність даних: Усі дані доступні в цифровій або графічній формі в основному тексті рукопису

Використання засобів штучного інтелекту (ШІ): Автор підтверджує, що не застосовував інструментальні засоби ШІ для написання цієї роботи

СПИСОК ЛІТЕРАТУРИ

1. Russell, S. J. & Norvig, P. “Artificial Intelligence: A modern approach”. *4th ed. Hoboken, NJ: Pearson,* 2021. http://lib.ysu.am/disciplines_bk/efdd4d1d4c2087fe1cbe03d9ced67f34.pdf. – [Accessed March 2, 2026].

2. Tomašev, N., Paquet, U., Hassabis, D. & Kramnik, V. “Reimagining chess with AlphaZero”. *Communications of the ACM.* 2022; 65 (2): 60–66. DOI: https://doi.org/10.1145/3460349.

3. McGrath, T., Kapishnikov, A., Tomašev, N., Pearce, A., Wattenberg, M., Hassabis, D., Kim, B., Paquet, U., & Kramnik, V. “Acquisition of chess knowledge in AlphaZero”. *Proceedings of the National Academy of Sciences of the United States of America*. 2022; 119 (47): e2206625119. DOI: https://doi.org/10.1073/pnas.2206625119.

4. McIlroy-Young, R., Wang, R., Sen, S., Kleinberg, J. M. & Anderson, A. “Detecting individual decision-making style: exploring behavioral stylometry in chess”. In: *Advances in Neural Information Processing Systems.* 2021; 34. DOI: https://doi.org/10.48550/arXiv.2208.01366.

5. Czech, J., Korus, P. & Kersting, K. “Improving AlphaZero using Monte-Carlo graph search”. *Proceedings of the International Conference on Automated Planning and Scheduling*. 2021; 31 (1): 103–111. DOI: https://doi.org/10.1609/icaps.v31i1.15952.

6. Schmid, M., Moravčík, M., Burch, N., et al. “Student of games: A unified learning algorithm for both perfect and imperfect information games”. *Science Advances*, 2023; 9 (46): eadg3256. DOI: https://doi.org/10.1126/sciadv.adg3256.

7. Klein, D. “Neural networks for chess: The magic of deep and reinforcement learning revealed”. *arXiv*. 2022. DOI: https://doi.org/10.48550/arXiv.2209.01506.

8. Parashar, A., Jha, A.K. & Kumar, M. “Analyzing a chess engine based on alpha-beta pruning, enhanced with iterative deepening”. In *Jacob, I.J., Kolandapalayam Shanmugam, S., Bestak, R. (eds) Expert Clouds and Applications. Lecture Notes in Networks and Systems*, *Springer, Singapore*, 2022; 444: 691–700. DOI: https://doi.org/10.1007/978-981-19-2500-9_51.

9. Hammersborg, P. & Strümke, I. “Reinforcement learning in an adaptable chess environment for detecting human-understandable concepts”. *arXiv.* 2022. DOI: https://doi.org/10.48550/arXiv.2211.05500.

10. Hammersborg, P. & Strümke, I. “Information-based explanation methods for deep learning agents – with applications on large open-source chess models”. *Scientific Reports*. 2024; 14: 20174. DOI: https://doi.org/10.1038/s41598-024-70701-2.

11. Nair, R. R., Babu, T., S. Kishore, Pavithra, K. & Sumana, S. G. “Improving alpha-beta pruning efficiency in chess engines”. In *International Conference on IT Innovation and Knowledge Discovery.* Manama, Bahrain. 2024. DOI: https://doi.org/10.1109/ITIKD63574.2025.11004942.

12. Kodhai, E., Bhuvaneswari, E., Devi, V. A. & Preethi, S. T. “Iterative deepening chess engine with Alpha Beta pruning”. In *International Conference on Smart Technologies for Sustainable Development Goals (ICSTSDG), IEEE.* 2024. P. 1–5. DOI: https://doi.org/10.1109/ICSTSDG61998.2024.11026648.

13. Garg, A. & Shrotriya, A. “Chess board: Performance of Alpha–Beta pruning in reducing node count of minimax tree”. In *Senjyu, T., So–In, C., Joshi, A. (eds) Smart Trends in Computing and Communications. Lecture Notes in Networks and Systems*. *Springer, Singapore.* 2023; 645: 661–671. DOI: https://doi.org/10.1007/978-981-99-0769-4_57.

Отримано 17.07.2026
Отримано після перегляду 28.08.2026
Прийнято 03.09.2026

DOI: https://doi.org/10.15276/ict.03.2026.15
UDC 004.832.2:794.1

Optimization of the minimax algorithm with alpha-beta pruning for chess artificial intelligence systems

Oleksii R. Silveistruk
ORCID: https://orcid.org/0009-0001-0368-3973; asilvejstruk@gmail.com
State University of Intelligent Technologies and Telecommunications, 1, Kuznechna St. Odesa, 65023, Ukraine

ABSTRACT

The paper considers approaches to training an artificial chess game model. The minimax algorithm with alpha-beta pruning, implemented in the Python programming language with the corresponding chess library, was chosen as the basic search method. Alpha-beta pruning reduces the computational complexity of traversing the game tree by discarding at the early stages those branches that do not affect the final solution. Heuristic extensions were added to the basic algorithm: iterative deepening, transposition tables with Zobrist hashing, ordering of moves according to the principle of priority of capturing the most valuable piece by the least valuable attacker, killer heuristics and historical heuristics, as well as quiescence search to weaken the horizon effect. An adaptive evaluation function was developed that takes into account material balance, positional evaluation tables, king safety, pawn structure, piece mobility, and key square control. The model was trained on a sample of over one hundred thousand games of levels from amateur to grandmaster. According to the experimental results, the combination of alpha-beta pruning with move ordering and transposition tables reduces the number of evaluated nodes by an average of one hundred and twenty times compared to the standard minimax at a depth of five half-moves, and the decision time is reduced from ninety-two point four seconds to zero point forty-three seconds. In simulated matches against a chess engine, the model demonstrates a high percentage of victories at initial levels of complexity (up to ninety-six point five percent at level zero). As a promising direction for further development, an architecture for integrating a neural network estimator based on an efficiently updated artificial neural network into a classical alpha-beta search tree is proposed, which will allow combining the interpretability of the classical method with the adaptability of deep learning. The proposed approach is suitable for decision support systems in deterministic games with complete information and in strategic planning problems.

Keywords: Artificial intelligence; chess; minimax algorithm; alpha-beta pruning; iterative deepening; transposition table; heuristic evaluation; quiescence search; machine learning; reinforcement learning

ІНФОРМАЦІЯ ПРО АВТОРА

Сільвейструк Олексій Русланович - здобувач вищої освіти ступеня «магістр», спеціальність Кібербезпека та захист інформації. Державний університет інтелектуальних технологій і зв'язку, вул. Кузнечна, 1. Одеса, 65023, Україна
Галузь досліджень: штучний інтелект, програмна інженерія

Oleksii R. Silveistruk - Master's degree student, specialty Cybersecurity and Information Protection. State University of Intelligent Technologies and Telecommunications, 1, Kuznechna St. Odesa, 65023, Ukraine
ORCID: https://orcid.org/0009-0001-0368-3973; asilvejstruk@gmail.com
Research field: artificial intelligence, software engineering