JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 37 (2025) 436–442
Received: 25 June 2025
Accepted: 19 August 2025

The impact of AI use on the performance of chess engines

Wpływ zastosowania sztucznej inteligencji na skuteczność silników szachowych

Jakub Król*, Jakub Smołka

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract

This paper presents a comprehensive comparative analysis of chess engines with particular focus on artificial intelligence technologies used in their implementation. Six engines were examined, representing various algorithmic approaches – from classical heuristic methods to advanced neural networks and reinforcement learning. Experiments were conducted for three different starting positions and with three time controls. The results clearly indicate the superiority of engines utilizing advanced machine learning techniques, which achieved the highest effectiveness in all tested conditions. The conducted research provides valuable information about the impact of applied AI technologies on the playing strength of chess engines in diverse conditions.

Keywords: chess engines; comparative analysis; neural networks; reinforcement learning

Streszczenie

Niniejsza praca przedstawia kompleksową analizę porównawczą silników szachowych ze szczególnym uwzględnieniem zastosowanych w nich technologii sztucznej inteligencji. Badaniu poddano sześć silników reprezentujących różne podejścia algorytmiczne – od klasycznych metod heurystycznych po zaawansowane sieci neuronowe i uczenie ze wzmocnieniem. Eksperymenty przeprowadzono dla trzech różnych pozycji startowych oraz przy trzech kontrolach czasu. Wyniki jednoznacznie wskazują na przewagę silników wykorzystujących zaawansowane techniki uczenia maszynowego, które osiągały najwyższą skuteczność we wszystkich testowanych warunkach. Przeprowadzone badania dostarczają informacji na temat wpływu zastosowanych technologii AI na ogólną siłę gry silników szachowych w różnorodnych warunkach.

Słowa kluczowe: silniki szachowe; analiza porównawcza; sieci neuronowe; uczenie ze wzmocnieniem

*Corresponding author
Email address: **s95456@pollub.edu.pl** (J. Król)

1. Wstęp

Szachy są uznawane za jedną z najstarszych strategicznych gier, których pozornie proste zasady prowadzą do niesamowitych możliwości taktycznych. Rozgrywka odbywa się na 64-polowej szachownicy, na której każdy z dwóch graczy posiada 16 bierek, w skład których wchodzą pionki i figury przemieszczające się w określony sposób. Celem gry jest zdobycie przewagi, aby finalnie pozbawić ruchu króla przeciwnika. Próby zrozumienia planu przeciwnika lub kreowanie własnego ataku czynią tę grę prawdziwym wyzwaniem intelektualnym o niezwykłym stopniu złożoności.

Z biegiem czasu opracowywano algorytmy, które miały zapewniać jak największą skuteczność i dokładność gry. Wraz z rozwojem technologii powstawały silniki szachowe, czyli programy symulujące ludzką grę. Początkowo nie były one w stanie rywalizować z najlepszymi szachistami. Jednak dzięki wykorzystaniu zaawansowanych algorytmów oraz technik uczenia maszynowego, niektóre silniki osiągnęły poziom gry, który znacznie przewyższa możliwości nawet najwybitniejszych arcymistrzów (arcymistrz to najwyższy tytuł szachowy).

Integralnym elementem oceny siły zarówno ludzi, jak i silników szachowych jest system rankingowy Elo [1]. Ten matematyczny model pozwala na określenie poziomu gracza na podstawie wyników rozegranych partii. Wprowadzenie tego systemu znacząco ułatwiło porównywanie zawodników oraz silników, stanowiąc podstawę do obiektywnej oceny ich możliwości.

Celem badań jest przeanalizowanie wybranych silników szachowych pod względem wykorzystanych technik sztucznej inteligencji oraz wyłonienie najlepszych z nich na podstawie liczby wygranych partii. Każdy z silników mierzy się z pozostałymi w różnych pozycjach startowych, aby ocenić wszystkie aspekty działania poszczególnych rozwiązań. Silniki rozgrywają partie między sobą z różnym tempem gry (np. 1 minuta + 1 sekunda za wykonany ruch lub 3 minuty + 2 sekundy za ruch). Takie podejście pozwoli sprawdzić, czy dana technika uczenia działa lepiej w zależności od dostępnego czasu na wybór najlepszego posunięcia.
W ramach przeprowadzanych badań postawiono następujące hipotezy badawcze:

- **H1:** Technika uczenia silnika szachowego wpływa na jego siłę gry i zrozumienie pozycji,

- **H2:** Tempo gry wpływa na dokładność gry silnika szachowego,
- **H3:** Początkowe ułożenie figur na szachownicy wpływa na dokładność gry silnika szachowego.

2. Przegląd literatury

Postęp w dziedzinie sztucznej inteligencji oraz rozwój zaawansowanych algorytmów obliczeniowych zrewolucjonizował sposób projektowania i optymalizacji silników szachowych. W przeglądzie literatury skupiono się na kluczowych badaniach analizujących różnorodne podejścia do tworzenia i doskonalenia tych silników, ze szczególnym uwzględnieniem wykorzystania AI. Omówione zostaną zarówno klasyczne algorytmy heurystyczne, jak i nowoczesne metody oparte na głębokim uczeniu i uczeniu ze wzmocnieniem.

Maciej Sójka w swoim artykule przeprowadza szczegółowe porównanie wydajności wybranych silników szachowych, koncentrując się na różnicach pod względem siły gry oraz zużycia zasobów sprzętowych [2]. Autor podkreśla znaczenie wyboru odpowiedniego silnika w zależności od konkretnych zastosowań, takich jak analiza gry czy gra symulacyjna. Porównuje silniki szachowe podczas klasycznych partii szachowych, ale także w wariancie Chess960 [3] oraz z pozycji wygranej dla czarnych bierek. Dzięki temu testuje silniki w różnych, skomplikowanych sytuacjach.

Jednym z kluczowych przełomów w dziedzinie silników szachowych jest algorytm AlphaZero przedstawiony przez D. Silvera i współautorów [4]. Wykorzystując technikę uczenia ze wzmocnieniem, algorytm opanowuje grę w szachy bez wcześniejszej wiedzy, demonstrując potencjał uniwersalnych algorytmów AI w rozwiązywaniu złożonych problemów decyzyjnych. Rozwinięciem tej pracy są badania T. Zahavy'ego et al. [5], które wprowadzają element kreatywności do AlphaZero, rozszerzając możliwości AI poza tradycyjne strategie. Według autorów, przyszłe badania mogą prowadzić do nowych odkryć w dziedzinie strategii szachowych, co zostało już potwierdzone, gdy AlphaZero zdominowała inne silniki.

Szczególne znaczenie mają badania nad metodami heurystycznymi i ich wydajnością. W. B. Putra i L. Heryawan wykazali, jak algorytm alfa-beta ogranicza przestrzeń przeszukiwań w analizie strategicznej [6]. Jest to najpowszechniej stosowany algorytm w logicznych grach planszowych. Polega na „odcinaniu" zbędnych gałęzi w drzewie przeszukiwań.

Rozwój silników szachowych opartych na uczeniu maszynowym omówili M. Block et al. [7], wskazując na efektywność uczenia ze wzmocnieniem w dynamicznej poprawie strategii poprzez adaptację do zmieniających się warunków i strategii przeciwnika. Jednak według autorów proces treningu wymaga optymalizacji, aby skrócić czas potrzebny na osiągnięcie efektywności. Yu Nasu skoncentrował się na sieciach neuronowych w ocenie pozycji w komputerowym shogi (japońska odmiana szachów), co pozwala na efektywniejsze dostosowywanie modeli do nowych danych bez konieczności ponownego trenowania [8]. Autor dostrzega potencjał wykorzystania uczenia maszynowego w strategicznych grach planszowych. Podobny kierunek reprezentują badania Chi S.Y.G., które eksplorują zastosowanie głębokich sieci rezydualnych w wariancie Crazyhouse Chess. Ten wariant jest bardziej skomplikowany od klasycznego i wymaga specyficznych metod analizy, w których ResNet wykazuje przewagę [9].

Q. A. Sadmine, A. Husna i M. Müller przeanalizowali efektywność silników Stockfish i Leela Chess Zero w porównaniu z tabelami końcówek [10]. Ich badania podkreślają różnice w podejściu obliczeniowym oraz zastosowania obu silników. Stockfish wykazał się wyższą dokładnością w klasycznych końcówkach, podczas gdy Leela Chess Zero okazała się bardziej elastyczna w adaptacji do różnych strategii końcówek.

Rozwój silników szachowych opartych na sztucznej inteligencji stanowi niezwykle dynamicznie rozwijającą się dziedzinę nauki, w której różnorodne podejścia wzajemnie się uzupełniają. Badania ukazują nie tylko potencjał AI w optymalizacji gry, ale także jej zastosowanie w szerszym kontekście analizy strategicznej.

3. Metoda badań

Badania zostały przeprowadzone na komputerze wyposażonym w kartę graficzną NVIDIA GeForce GTX 1080, procesor Intel Core i7-8700, 16 GB pamięci RAM oraz system operacyjny Windows 10. Taka konfiguracja sprzętowa zapewniała odpowiednią wydajność do uruchamiania i analizowania partii rozgrywanych przez silniki szachowe, bez ryzyka ograniczeń wydajnościowych wpływających na wyniki eksperymentu.

Do przeprowadzenia badań nad silnikami szachowymi wykorzystano program LucasChess [11], który jest zaawansowanym zestawem narzędzi dedykowanych grze, nauce i treningowi szachowemu. Program oferuje funkcje umożliwiające tworzenie turniejów pomiędzy różnymi silnikami szachowymi, co pozwala na ich porównanie w kontrolowanych warunkach. Badania obejmują testowanie silników w niestandardowych pozycjach początkowych (ze straconym tempem przez białe oraz pozycja z losowo ustawionymi figurami na ostatniej linii), co pozwala sprawdzić zdolności adaptacyjne i kreatywność silników w sytuacjach wykraczających poza klasyczne partie szachowe. Taki podział eksperymentów umożliwia dokładną ocenę mocnych i słabych stron różnych silników szachowych, uwzględniając zarówno ich teoretyczne możliwości, jak i praktyczną efektywność w nietypowych scenariuszach.

Turnieje zostały przeprowadzone w trzech kontrolach czasu: 1 minuta + 1 sekunda, 3 minuty + 2 sekundy oraz 5 minut + 2 sekundy (czas podstawowy przysługujący zawodnikowi na rozegranie partii + czas bonusowy, dodawany po wykonaniu każdego posunięcia). W ten sposób powstało 9 konfiguracji (trzy tempa rozgrywki dla trzech typów pozycji), gdzie dla każdej utworzono oddzielny turniej, w którym silniki szachowe rozegrały po 10 partii (po dwie partie przeciwko każdemu ze wszystkich badanych silników, tak aby wyrównać liczbę partii rozgrywanych kolorem białym, jak i czarnym). Wybrane silniki szachowe są dostępne w programie

LucasChess, bez konieczności ich importowania do środowiska.

W celu obiektywnego porównania skuteczności poszczególnych silników zastosowano klasyczny system ocen punktowych. Za wynik każdej partii przyznawano punkty według zasad:

- 1 punkt za zwycięstwo,
- 0,5 punktu za remis,
- 0 punktów za porażkę.

Na tej podstawie obliczano skuteczność silnika w danej konfiguracji turnieju jako procent maksymalnej możliwej liczby punktów, zgodnie z poniższym wzorem:

$$Skuteczność(\%) = \left(\frac{W + 0{,}5 * R}{W + R + P}\right) * 100 \qquad (1)$$

gdzie:

- W - liczba wygranych partii,
- R - liczba remisów,
- P - liczba przegranych partii.

3.1. Wybrane silniki szachowe

W badaniach wykorzystano sześć różnorodnych silników szachowych, które reprezentują zarówno nowoczesne podejścia oparte na sztucznej inteligencji, jak i klasyczne algorytmy heurystyczne. Każdy z silników posiada unikalne cechy, które wpływają na sposób oceny pozycji, podejmowania decyzji oraz ogólną siłę gry.

Silniki wykorzystujące sztuczną inteligencję:

- **Stockfish 17** [12] – jeden z najsilniejszych dostępnych silników, oparty na zaawansowanych metodach przeszukiwania drzewa gry. Od 2020 roku korzysta z efektywnie aktualizowanej sieci neuronowej (NNUE), co umocniło jego pozycję w ścisłej czołówce pomiędzy silnikami szachowymi. Wersja 17 została wydana w 2023r.,
- **Lc0 v0.31.2 (Leela Chess Zero)** [13] – silnik oparty na sieciach neuronowych oraz uczeniu ze wzmocnieniem, wykorzystujący uczenie maszynowe i metodę Monte Carlo Tree Search (MCTS). Jest to ogólnodostępna wersja AlphaZero, która rezygnuje z klasycznych heurystyk i uczy się grając samodzielnie, przeciwko samemu sobie,
- **Komodo Dragon 1** [14] – silnik łączący klasyczne algorytmy z oceną pozycyjną, znany ze strategicznej i solidnej gry. Podobnie jak Stockfish, wykorzystuje efektywnie aktualizujące sieci neuronowe (NNUE). We wcześniejszej wersji bez „1” w nazwie, Komodo Dragon bazował jedynie na standardowych algorytmach.

Silniki klasyczne (bez wykorzystania AI):

- **Houdini 1.5a** [15] – silnik o wysokiej sile gry, szczególnie skuteczny w dynamicznych i taktycznych pozycjach. Bazuje na klasycznych heurystykach oraz obszernej bazie debiutów z archiwalnych partii arcymistrzów. W badaniu zastosowano wersję 1.5a, jedną z wcześniejszych, lecz wciąż sprawną,
- **Andscacs 0.9432n** – hiszpański silnik charakteryzujący się wydajnością i efektywną heurystyką oceny pozycji. Wyróżnia się skutecznością zarówno w grze pasywnej, jak i agresywnej, stosując zoptymalizowany algorytm alfa-beta,
- **Gull 3 32bit** – silnik znany z dynamicznej gry i dobrej optymalizacji. Często konkuruje z czołowymi silnikami, cechuje się niskim zużyciem zasobów systemowych. Podobnie jak Houdini oraz Andscacs, bazuje na standardowych algorytmach, bez wykorzystania AI.

3.2. Pozycje startowe w rozgrywkach

Klasyczna pozycja startowa (Rysunek 1) – standardowa konfiguracja figur stosowana we wszystkich tradycyjnych turniejach. W tym ustawieniu białe mają niewielką przewagę, wynikającą z faktu rozpoczęcia partii.

Rysunek 1: Standardowa pozycja szachowa.

Pozycja ze straconym tempem przez białe (Rysunek 2) – ustawienie, w którym białe rozpoczynają grę od nieoptymalnego ruchu, co daje przewagę czarnym. Umożliwia to ocenę wpływu inicjatywy w pozycji oraz utraty wyboru pierwszego ruchu przez białe.

Rysunek 2: Pozycja ze straconym tempem.

Szachy Fischera (Rysunek 3) – wariant szachów, w którym pozycje figur na linii podstawowej są losowane, zgodnie z zasadami Chess960. Eliminuje to znaczenie przygotowania debiutowego, zwiększając rolę kreatywności i elastyczności. Wariant ten został opracowany przez byłego mistrza świata Bobby’ego Fischera.

Rysunek 3: Pozycja chess960.

4. Wyniki

Po przeprowadzonych turniejach w różnych konfiguracjach zebrano dane, na których podstawie przedstawiono wyniki. Rezultaty zaprezentowane w tabelach przedstawiają procentowy rezultat (skuteczność), liczbę zwycięstw, remisów oraz porażek. Każda z tabel opisuje wyniki z jednej z dziewięciu konfiguracji, w której był przeprowadzany turniej.

W pozycji klasycznej przy tempie 1min+1s najwyższy wynik uzyskały silniki Stockfish i Komodo Dragon (używające NNUE), które nie przegrały żadnej partii. Lc0, wykorzystujący uczenie ze wzmocnieniem zakończył test z wynikiem skuteczności na poziomie 50%, natomiast klasyczne silniki (Gull, Houdini, Andscacs) wypadły najsłabiej. Gull uzyskał więcej zwycięstw niż Lc0, ale brak remisów obniżył jego ogólny wynik. Wyniki są widoczne w Tabeli 1.

Tabela 1: Pozycja klasyczna, 1min+1s [S – skuteczność, Z – liczba zwycięstw, R – liczba remisów, P – liczba porażek]

Silnik	S (%)	Z	R	P
Stockfish	80	6	4	0
Komodo Dragon	80	6	4	0
Lc0	50	3	4	3
Gull	40	4	0	6
Andscacs	25	1	3	6
Houdini	25	2	1	7

Dla pozycji klasycznej z czasem 3min+2s Stockfish i Komodo Dragon ponownie osiągnęły najwyższe wyniki, podkreślając przewagę sieci neuronowych. Lc0 przegrał aż 5 partii, zrównując wynik z Andscacs, który w swojej implementacji wykorzystuje jedynie standardowe algorytmy. Najsłabiej wypadły Houdini i Gull, przegrywając 6 z 10 partii, co przedstawia Tabela 2.

Tabela 2: Pozycja klasyczna, 3min+2s [S – skuteczność, Z – liczba zwycięstw, R – liczba remisów, P – liczba porażek]

Silnik	S (%)	Z	R	P
Stockfish	85	7	3	0
Komodo Dragon	75	6	3	1
Lc0	40	3	2	5
Andscacs	40	3	2	5
Gull	35	3	1	6
Houdini	25	1	3	6

W przypadku konfiguracji 5min+2s, Stockfish uzyskał najwyższy wynik, przewyższając Komodo. Lc0 znacząco zwiększył poziom skuteczności, co sugeruje na znaczący wpływ dłuższego czasu na namysł dla techniki uczenia ze wzmocnieniem. Natomiast Gull i Houdini ponownie zajęły ostatnie miejsca, co jest widoczne w Tabeli 3.

Tabela 3: Pozycja klasyczna, 5min+2s [S – skuteczność, Z – liczba zwycięstw, R – liczba remisów, P – liczba porażek]

Silnik	S (%)	Z	R	P
Stockfish	85	7	3	0
Komodo Dragon	75	5	5	0
Lc0	70	6	2	2
Andscacs	30	1	4	5
Gull	25	3	3	6
Houdini	15	0	3	7

Tabela 4 ukazuje dominację silników Stockfish i Komodo Dragon w pozycji ze straconym tempem dla białych. Lc0 utrzymał stabilny, umiarkowany poziom. W odróżnieniu od silników wykorzystujących AI, standardowe silniki były w stanie wygrać jedynie jedną z dziesięciu partii. Mógł na to wpłynąć znacząco ograniczony czas na wybór posunięcia w bardzo nietypowej pozycji.

Tabela 4: Pozycja ze straconym tempem białych, 1min+1s [S – skuteczność, Z – liczba zwycięstw, R – liczba remisów, P – liczba porażek]

Silnik	S (%)	Z	R	P
Stockfish	90	8	2	0
Komodo Dragon	80	7	2	1
Lc0	65	5	3	2
Houdini	30	1	4	5
Gull	20	1	2	7
Andscacs	15	1	1	8

Według wyników z Tabeli 5, w tempie 3min+2s, gorsza pozycja nie przeszkodziła silnikowi Stockfish w dominacji. Lc0 wykazał się dużą liczbą remisów, co dało mu trzecie miejsce. Tym razem Andscacs uzyskał najgorszy wynik skuteczności, na poziomie 20%. Standardowe silniki odnotowały nieznaczną poprawę skuteczności względem szybszego tempa gry.

Tabela 5: Pozycja ze straconym tempem białych, 3min+2s [S – skuteczność, Z – liczba zwycięstw, R – liczba remisów, P – liczba porażek]

Silnik	S (%)	Z	R	P
Stockfish	90	8	2	0
Komodo Dragon	75	6	3	1
Lc0	55	3	5	2
Gull	35	2	3	5
Houdini	25	2	1	7
Andscacs	20	1	2	7

Przy 5min+2s Stockfish i Komodo utrzymały wysoką skuteczność. Lc0 wygrał tylko 3 partie, pomimo dłuższego czasu na wybór najlepszego posunięcia. Wskazuje to na przewagę sieci neuronowych nad uczeniem ze wzmocnieniem w niestandardowych pozycjach. Houdini i Gull ponownie wypadły słabo, osiągając odpowiednio 1 i 0 zwycięstw. Wyniki te przedstawia Tabela 6.

Tabela 6: Pozycja ze straconym tempem białych, 5min+2s [S – skuteczność, Z – liczba zwycięstw, R – liczba remisów, P – liczba porażek]

Silnik	S (%)	Z	R	P
Stockfish	90	8	2	0
Komodo Dragon	80	7	2	1
Lc0	45	3	3	4
Andscacs	35	2	3	5
Houdini	25	1	3	6
Gull	25	0	5	5

W partiach typu Chess960, w tempie 1min+1s, Stockfish i Komodo ponownie zdominowały rywalizację, co wyraźnie widoczne jest w Tabeli 7. Zastosowanie NNUE w implementacjach tych silników okazało się niezawodne w każdym typie pozycji. Pozostałe silniki uzyskały znacznie niższe wyniki. Lc0 uzyskało najniższą skuteczność w porównaniu z innymi konfiguracjami turniejów, zrównując poziom do silników wykorzystujących standardowe algorytmy (Andscacs, Gull, Houdini).

Powodem tak niezadowalających rezultatów może być połączenie skomplikowanej pozycji z szybkim tempem gry.

Tabela 7: Pozycja Chess960, 1min+1s [S – skuteczność, Z – liczba zwycięstw, R – liczba remisów, P – liczba porażek]

Silnik	S (%)	Z	R	P
Stockfish	90	8	2	0
Komodo Dragon	85	7	3	0
Andscacs	35	1	5	4
Lc0	30	1	4	5
Houdini	30	2	2	6
Gull	30	2	2	6

Dla pozycji „Szachów Fischera” z tempem 3min+2s dominacja Stockfisha utrzymała się, ponownie nie przegrał on żadnej partii. Lc0 był trzeci, głównie dzięki remisom. Dodatkowy czas pozwolił mu poprawić wynik względem szybszego tempa gry w tej pozycji. W dolnej części Tabeli 8 znajdują się Houdini i Andscacs.

Tabela 8: Pozycja Chess960, 3min+2s [S – skuteczność, Z – liczba zwycięstw, R – liczba remisów, P – liczba porażek]

Silnik	S (%)	Z	R	P
Stockfish	90	8	2	0
Komodo Dragon	75	6	3	1
Lc0	55	3	5	2
Gull	35	2	3	5
Andscacs	25	1	3	6
Houdini	20	1	2	7

W ostatnim teście (Chess960, 5min+2s), Stockfish ponownie zakończył turniej bez porażki. Komodo Dragon i Lc0 utrzymały wysoką formę, oddzielając silniki wykorzystujące AI z klasycznymi implementacjami. Houdini zakończył rundę z ośmioma porażkami i wynikiem na poziomie 15%. Wyniki przedstawia Tabela 9.

Tabela 9: Pozycja Chess960, 5min+2s [S – skuteczność, Z – liczba zwycięstw, R – liczba remisów, P – liczba porażek]

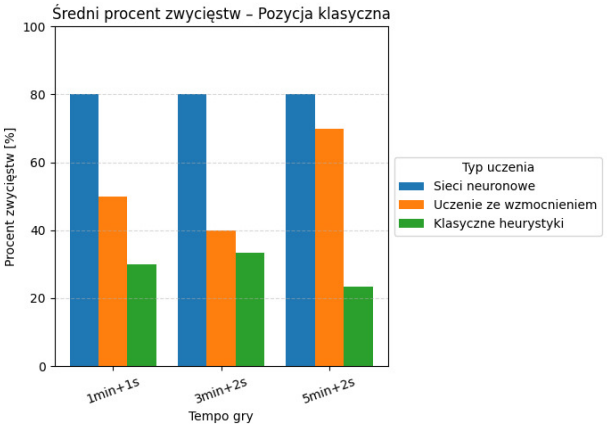
Silnik	S (%)	Z	R	P
Stockfish	90	8	2	0
Komodo Dragon	70	5	4	1
Lc0	55	4	3	3
Andscacs	35	3	1	6
Gull	35	2	3	5
Houdini	15	1	1	8

Na podstawie wszystkich przeprowadzonych testów można zauważyć wyraźną dominację silnika Stockfish, który nie przegrał żadnej partii niezależnie od ustawienia początkowego ani tempa gry. W każdej konfiguracji utrzymywał najwyższą skuteczność, co potwierdza jego pozycję lidera wśród współczesnych silników szachowych. Komodo Dragon również wykazywał bardzo wysoką stabilność i siłę gry, ustępując jedynie minimalnie Stockfishowi. Oba silniki wykorzystujące efektywnie aktualizujące sieci neuronowe wyróżniły się wysoką skutecznością na tle pozostałych. Lc0 prezentowało umiarkowane wyniki, jego technika uczenia ze wzmocnieniem była wyraźnie skuteczniejsza od klasycznych algorytmów zawężania gałęzi przeszukiwań najlepszych posunięć, lecz również słabsza w stosunku do NNUE. Najgorzej wypadły Houdini, Gull i Andscacs, które miały problemy ze skutecznością w bardziej złożonych lub dynamicznych warunkach. Oznacza to, że klasyczne metody nie są w stanie konkurować z nowoczesnymi technikami wykorzystującymi sztuczną inteligencję.

5. Dyskusja

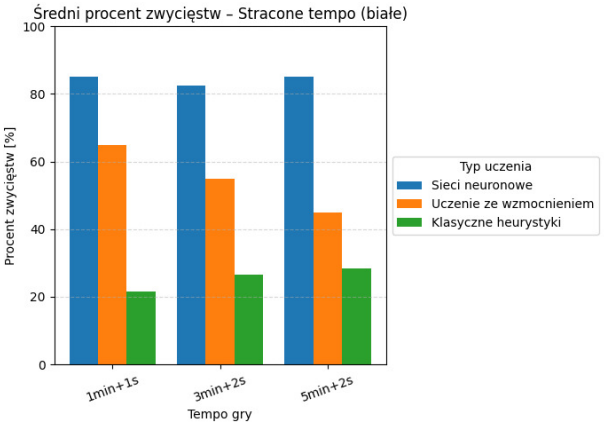
Przeprowadzone badania porównawcze dla sześciu wybranych silników szachowych (Stockfish, Komodo Dragon, Lc0, Gull, Andscacs i Houdini) w trzech różnych typach pozycji oraz przy trzech różnych kontrolach czasu dostarczyły kompleksowego obrazu ich wydajności. Analiza zgromadzonych danych pozwala na obserwację istotnych zależności dotyczących skuteczności poszczególnych silników w zależności od warunków gry oraz zastosowanych technik sztucznej inteligencji.

Wyniki silników w pozycjach klasycznych dla trzech temp gry przedstawiono na Rysunku 4. Widoczna jest wyraźna dominacja silników wykorzystujących AI. Stockfish i Komodo Dragon, których implementacje opierają się na efektywnie aktualizujących sieciach neuronowych utrzymują wysoki i stabilny poziom skuteczności niezależnie od czasu gry. Lc0, bazujące na uczeniu ze wzmocnieniem wykazuje zróżnicowaną skuteczność – od około 40% przy tempie 3min+2s do niemal 70% przy 5min+2s, co sugeruje, że dłuższy czas na analizę pozycji znacząco poprawia jego wyniki. Silniki bazujące na klasycznych heurystykach (Gull, Andscacs, Houdini) osiągają najniższe rezultaty, które dodatkowo maleją przy wydłużeniu czasu gry – z poziomu około 30% do nieco ponad 20%.



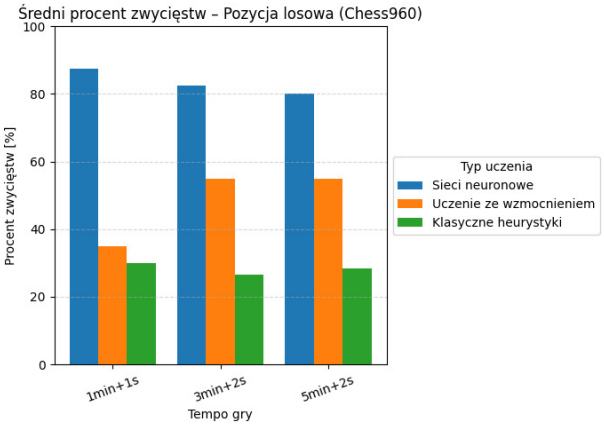
Rysunek 4: Skuteczność technik uczenia w pozycji klasycznej.

Analizując wyniki dla pozycji ze straconym tempem dla białych (Rysunek 5), ponownie widoczna jest dominacja silników Stockfish i Komodo Dragon, utrzymujących około 85% skuteczności. Interesującą obserwacją jest spadek wyników Lc0 wraz z wydłużeniem czasu gry – od około 65% przy 1min+1s do 45% przy 5min+2s. Klasyczne silniki wykazują odwrotną tendencję – im więcej czasu na analizę, tym lepsze ich wyniki, co potwierdza skuteczność algorytmów alfa-beta przy wydłużonym czasie gry. Jednak wyniki te nadal nie są nawet zbliżone do skuteczności silników wykorzystujących NNUE (Stockfish, Komodo Dragon) lub uczenie ze wzmocnieniem (Lc0).



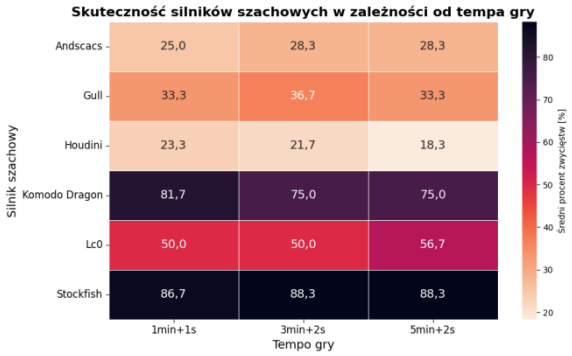
Rysunek 5: Skuteczność technik uczenia w pozycji ze straconym tempem białych.

Na Rysunku 6 przedstawiono skuteczność w losowych pozycjach Chess960. Stockfish i Komodo Dragon ponownie osiągają najwyższe wyniki, choć zauważalna jest lekka tendencja spadkowa przy dłuższych tempach. Z kolei Lc0, wykorzystujący uczenie ze wzmocnieniem, osiąga znaczący wzrost skuteczności – z 30% przy 1min+1s do ponad 55% przy 3min+2s i 5min+2s. Sugeruje to, że silnik ten wymaga więcej czasu na analizę nietypowych pozycji. Klasyczne silniki, jak Gull, Houdini i Andscacs, osiągają najsłabsze wyniki i nie wykazują poprawy wyników w przypadku wydłużonego czasu gry. Widoczna jest znacząca przewaga silników wykorzystujących techniki sztucznej inteligencji.



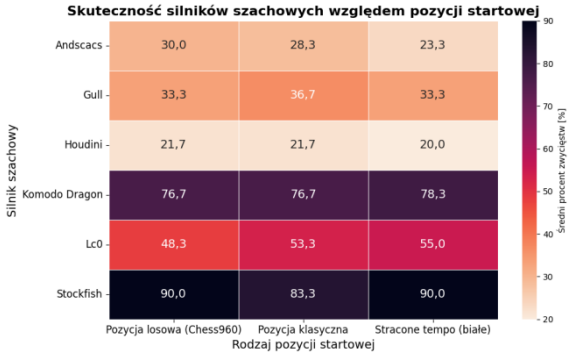
Rysunek 6: Skuteczność technik uczenia w pozycji chess960.

Pierwsza mapa cieplna (Rysunek 7) przedstawia skuteczność silników w zależności od tempa gry, niezależnie od pozycji startowej. Ukazuje ona bezsprzeczną przewagę implementacji wykorzystujących AI. Stockfish cechuje się niezwykłą stabilnością, osiągając 80–90% skuteczności we wszystkich tempach gry. Komodo Dragon osiąga nieznacznie niższe wartości. Lc0 charakteryzuje się dużą stabilnością skuteczności, lecz na znacznie niższym poziomie skuteczności niż silniki korzystające z sieci neuronowych (Stockfish, Komodo Dragon). Klasyczne silniki (Gull, Andscacs, Houdini) pozostają znacznie poniżej 40% skuteczności, niezależnie od dostępnego czasu. Najsłabiej wypada Houdini, który posiada skuteczność na poziomie 20% w każdym możliwym tempie rozgrywki. Oznacza to, że dodatkowy czas na znalezienie ruchu nie ma większego znaczenia dla klasycznych silników szachowych.



Rysunek 7: Mapa cieplna skuteczności silników w zależności od tempa gry.

Druga mapa cieplna (Rysunek 8) koncentruje się na wpływie pozycji startowej na skuteczność silników, niezależnie od tempa gry. Stockfish i Komodo Dragon pozostają liderami we wszystkich układach początkowych. Lc0 uzyskuje najlepsze wyniki w pozycjach klasycznych, a jego skuteczność wyraźnie spada w niestandardowych pozycjach, co może wskazywać na ograniczenia podejścia opartego na uczeniu ze wzmocnieniem w warunkach odbiegających od wzorcowych. Klasyczne silniki nie wykazują znaczących różnic między typami pozycji, lecz ich ogólna skuteczność jest niska.



Rysunek 8: Mapa cieplna skuteczności silników w zależności od pozycji początkowej.

Podsumowując, przeprowadzone badania wyraźnie wskazują na dominację silników wykorzystujących zaawansowane techniki sztucznej inteligencji – zarówno sieci neuronowe (Stockfish, Komodo Dragon), jak i uczenie ze wzmocnieniem (Lc0). Silniki te konsekwentnie osiągają najwyższe wyniki we wszystkich warunkach testowych. Natomiast standardowe implementacje osiągały najsłabsze wyniki. Zauważalne są również zróżnicowane reakcje silników na długość kontroli czasu - niektóre silniki zyskują przewagę przy dłuższej analizie (zwłaszcza Lc0), podczas gdy inne nie odnotowują znaczącej poprawy. Wskazuje to na złożone zależności pomiędzy architekturą silnika szachowego, typem zastosowanego algorytmu oraz warunkami gry, co jest istotnym obszarem do dalszych badań i optymalizacji. Kolor bierek nie miał dużego wpływu na skuteczność silników

szachowych, jednak można było odnotować większą liczbę remisów i przegranych partii po stronie koloru czarnego.

6. Wnioski

Przeprowadzone badania pozwoliły na sformułowanie wniosków oraz potwierdziły wszystkie trzy postawione hipotezy. Wykazano, że technika uczenia silnika szachowego (H1), tempo gry (H2) oraz początkowe ustawienie figur (H3) mają istotny wpływ na jego skuteczność.

Silniki wykorzystujące zaawansowane techniki sztucznej inteligencji – sieci neuronowe (Stockfish, Komodo Dragon) oraz uczenie ze wzmocnieniem (Lc0) – konsekwentnie osiągały lepsze wyniki niż silniki bazujące na klasycznych algorytmach heurystycznych (Houdini, Gull, Andscacs). Szczególnie wyraźna przewaga AI widoczna była w niestandardowych warunkach gry, takich jak losowe pozycje Chess960 czy pozycje ze straconym tempem. Stockfish wykazał największą stabilność i dominację – nie przegrał żadnej partii niezależnie od pozycji ani tempa gry. Komodo Dragon utrzymywał również bardzo wysoką skuteczność. Lc0 prezentował większą zmienność wyników – zyskiwał na dłuższym czasie gry w pozycjach klasycznych, ale tracił przewagę w układach niestandardowych, co wskazuje na złożoną interakcję między algorytmem a strukturą pozycji.

Przeprowadzone badania, mimo kompleksowego podejścia, posiadają pewne ograniczenia, które warto uwzględnić i wskazać kierunki przyszłych badań, które powinny objąć:

- analizę wpływu konfiguracji sprzętowej (CPU vs GPU) na wydajność silników AI,
- rozszerzenie testów o większą liczbę partii dla zwiększenia istotności statystycznej,
- głębszą analizę specyficznych typów pozycji (np. końcówki, pozycje taktyczne),
- wykorzystanie nowszych wersji istniejących silników oraz innych obiecujących implementacji.

Sztuczna inteligencja w szachach nie tylko przewyższyła klasyczne metody, ale również ujawniła potencjał dalszego rozwoju poprzez adaptację do skomplikowanych zadań i pozycji. Najsilniejsze silniki nie tylko wygrywają, ale wykazują odporność na zmienne warunki gry. Przyszłość rywalizacji silników szachowych będzie zależeć od tego, jak skutecznie uda się połączyć moc obliczeniową z elastycznymi, uczącymi się modelami.

Literatura

[1] A. Elo, The Proposed USCF Rating System, Its Development, Theory, and Applications, Chess Life 22 (1967) 242–247.

[2] M. Sójka, Performance Comparison Between Selected Chess Engines, Journal of Computer Sciences Institute 24 (2022) 228–235.

[3] Strona lichess.org, zasady wariantu chess960, https://lichess.org/variant/chess960, [01.07.2025].

[4] D. Silver, T. Hubert, J. Schrittwieser, A General Reinforcement Learning Algorithm That Masters Chess, Shogi, and Go Through Self-Play, Science 362 (2018) 1140–1144, https://doi.org/10.1126/science.aar6404.

[5] T. Zahavy, V. Veeriah, S. Hou, Diversifying AI: Towards Creative Chess With AlphaZero, arXiv (2023), https://doi.org/10.48550/arXiv.2308.09175.

[6] W. B. Putra, L. Heryawan, Applying Alpha-Beta Algorithm in a Chess Engine, Jurnal Teknosains 6 (2016) 37–43.

[7] M. Block, M. Bader, E. Tapia, Using Reinforcement Learning in Chess Engines, Research in Computing Science 35 (2008) 31–40.

[8] Y. Nasu, Efficiently Updatable Neural-Network-Based Evaluation Functions for Computer Shogi, In The 28th World Computer Shogi Championship Appeal Document (2018) 185.

[9] S. Y. G. Chi, Exploring the Performance of Deep Residual Networks in Crazyhouse Chess, arXiv (2019), https://doi.org/10.48550/arXiv.1908.09296.

[10] Q. A. Sadmine, A. Husna, M. Müller, Stockfish or Leela Chess Zero? A Comparison Against Endgame Tablebases, In Advances in Computer Games, Springer Nature Switzerland (2023) 26–35.

[11] Strona internetowa programu Lucas Chess, https://lucaschess.pythonanywhere.com, [01.07.2025].

[12] Strona internetowa silnika szachowego Stockfish, https://stockfishchess.org, [01.07.2025].

[13] Strona internetowa silnika szachowego Lc0, https://lczero.org, [01.07.2025].

[14] Strona internetowa silnika szachowego Komodo Dragon, https://komodochess.com, [01.07.2025].

[15] Geneza i opis silnika szachowego Houdini, https://www.chessprogramming.org/Houdini, [01.07.2025].