JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 24 (2022) 228–235
Received: 7 June 2022
Accepted: 2 July 2022

Performance comparison between selected chess engines

Porównanie wydajności wybranych silników szachowych

Maciej Sójka*

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

Abstract
Selected chess engines were compared to each other in terms of performance, using Lucas Chess. The list of engines was divided into three categories, depending on strength in ELO points. The point of this study is to find the strongest and the lightest engines in each category. Then, each category was tested using three different starting positions. White, black and overall wins were highlighted. At the same time, data of CPU and RAM usage of each engine was collected. A script was developed to print CPU and RAM usage of a specific process. Maximum and average percent of used CPU and RAM were highlighted. Chess engines with most amount of wins were, from weakest to strongest: Bikjump, Rybka and Stockfish. Least amount of system resources was consumed by: Cinnamon, Demolito and Critter.

Keywords: chess; chess engines; performance comparison

Streszczenie
Wydajność wybranych silników szachowych została porównana z użyciem programu Lucas Chess. Listę silników podzielono na kategorie w zależności od punktów ELO. Celem badań było znalezienie najmocniejszych i najlżejszych silników w kategoriach. Każdą z kategorii zbadano pod kątem trzech ustawień szachownicy. Wyróżniono wygrane ogólne oraz wygrane jako różne kolory bierek. Równolegle przeprowadzono badanie zużycia zasobów komputera. Przygotowano skrypt zapisujący wartości procentowe wykorzystania pamięci RAM i procesora przez konkretne procesy. Wyróżniono średnie i maksymalne procentowe zużycie CPU i pamięci RAM. Silniki z największą liczbą wygranych, od najsłabszych do najsilniejszych, to: Bikjump, Rybka i Stockfish. Najmniejsze zapotrzebowanie na zasoby mają: Cinnamon, Demolito i Critter.

Słowa kluczowe: szachy; silniki szachowe; porównanie wydajności

*Corresponding author
Email address: maciej.sojka@pollub.edu.pl (M. Sójka)

1. Wstęp

Szachy współczesne powstały w piętnastym wieku w południowej Europie. Jest to gra dla dwóch graczy, która wykorzystuje kwadratową siatkę o szerokości ośmiu pól. Na początku rozgrywki każdy gracz posiada szesnaście typów bierek: króla, hetmana, dwie wieże, dwóch gońców, dwa konie i osiem pionków. Bierki oraz pola mają różne kolory (czarny i biały). Celem gry jest uniemożliwienie ruchu przeciwnika poprzez "szach mat", czyli zbicie króla.

Jednym z celów inżynierów i naukowców było stworzenie maszyny symulującej grę ludzkiego gracza. W tym celu stworzono tak zwane "silniki szachowe", czyli algorytmy analizujące pozycję bierek i zwracające ruch lub listę ruchów. Silniki szachowe zwykle sterowane są z poziomu linii poleceń i do interakcji w formie innej niż komendy tekstowe wymagany jest zewnętrzy interfejs graficzny (tak zwany "front-end").

Umiejętność graczy i silników szachowych wyrażana jest poprzez system punktowy ELO, który został stworzony przez Arpada Elo [1]. Punkty ELO są wyrażane poprzez niezerową liczbę. Zwycięzca dostaje punkty w przypadku wygranej, a przegrany traci tyle samo. W przypadku remisu wyżej oceniany gracz traci punkty na rzecz przeciwnika. Punkty ELO liczone są wewnątrz pewnej puli graczy. Ogólnoświatowy współczynnik ELO jest liczony przez Międzynarodową Federację Szachową (fr. Fédération Internationale des Échecs, FIDE [2]).

Celem badań jest porównanie silników szachowych w wyznaczonych kategoriach. Silniki będą grupowane na podstawie ich współczynnika ELO, a rozgrywki zostaną rozpoczęte na podstawie różnych pozycji startowych. Na każdą kategorię wyróżnione zostaną silniki:

- z największą ilością wygranych gier,
- wymagające najmniejszą ilość zasobów komputera.

2. Przegląd literatury

Jedną z pierwszych prac związaych z algorytmami znajdującymi optymalne ruchy jest aplikacja do rozwiązywania problemów "mat w x ruchów" opisana przez Vladan V. Vučković [3]. Logika programu wykorzystuje algorytm alfa-beta. Kod źródłowy został napisany w Asemblerze, co gwarantuje wysoką wydajność nawet na procesorze Celeron z taktowaniem 1GHz.

Wykorzystanie algorytmu alfa-beta w silniku szachowym zostało głębiej opisane w pracy W. B. Putra i L. Heryawan [4]. Użycie alfa-beta do redukcji niepotrzebnych krawędzi w drzewie możliwych ruchów zwiększyło wydajność badanego silnika.

Szersze wykorzystanie silnika szachowego zostało zrealizowane przez Hongyu Zang, Zhiwei Yu i Xiaojun Wan w formie automatycznego komentatora gry [5]. Do oceny ruchów został wykorzystany silnik stworzony na

bazie sieci neuronowej i uczenia maszynowego. Na podstawie oceny szachownicy przez silnik generowane są informacje tekstowe wyświetlane na ekranie.

Praca Marco Block, Maro Bader i innych [6] opisuje poszerzenie silnika szachowego wykorzystującego uczenie maszynowe o kategoryzację stanu szachownicy na podstawie aktualnego stanu gry (z ang. „otwarcie", „gra środkowa", „gra końcowa"). Implementacja kategoryzacji, wraz z powiększeniem bazy ruchów, na których uczy się silnik, pozwoliła na znaczne zwiększenie jego umiejętności.

W pracy Norhan Hesham, Osama Abu-Elnasr, i Samir Elmougy [7] został przedstawiony alternatywny silnik szachowy, wykorzystujący bibliotekę uprzednio zdeklarowanych optymalnych ruchów. Na początku baza ruchów była wypełniana danymi z rozgrywek wcześniejszych mistrzów szachowych, którą silnik stopniowo dopełniał własnymi grami. Użycie takiego rozwiązania zwiększyło możliwości silnika wobec innych rozwiązań.

Hybrydowa reprezentacja szachownicy [8] przedstawiona przez Sigit Kariagil Bimonugroho i Nur Ulfa Maulidevi miała na celu przyspieszenie procesu analizy rozgrywki i wybrania optymalnego ruchu. Wyniki rozgrywek szachowych pokazały, że opisywana reprezentacja jest szybsza i mocniejsza od niektórych alternatyw.

Opis i porównanie dwóch silników szachowych zostało dokonane w pracy Shiva Maharaj, Nicka Polsona i Alexa Turka [9]. Głównym celem badań była ocena rozwiązania Łamigłówki Plasketta przez silniki Stockfish i LCZero. Obydwa silniki i łamigłówka zostały szeroko opisane od strony teoretycznej. Stockfish ostatecznie uzyskał lepsze wyniki ze względu na szybkość algorytmu wyszukującego optymalny ruch.

3. Metodyka badań

Komputer, na którym przeprowadzano badania, to Lenovo IdeaPad L340 Gaming. Pełna specyfikacja sprzętu opisana jest przez Tabelę 1. Laptop pracował w następującej konfiguracji:

- w opcjach zasilania pobór mocy został ustawiony na tryb „najwyższej wydajności",
- w opcjach UEFI praca wiatraków została ustawiona na tryb „najwyższej wydajności",
- podłączona została dodatkowa podstawka chłodząca.

Tabela 1: Specyfikacja podzespołów komputera badawczego

Procesor	Intel Core i7-9750H
Karta graficzna 1	NVIDIA GeForce GTX 1650
Karta graficzna 2	Intel UHD Graphics 630
Pamięć RAM	16 GB
System operacyjny	Windows 11

Do wykonania eksperymentu użyty został program Lucas Chess w wersji R 2.01a. Lucas Chess jest to zestaw narzędzi do gry, nauki i treningu umiejętności szachowych napisany w języku Python [10]. Zawiera on między innymi funkcję turnieju silników szachowych z gotową listą komputerowych graczy. W ramach pracy wyodrębniona została część silników, podzielona później na następujące kategorie:

- Kategoria 1 (do 2500 ELO):
  - Bikjump 2.01 – napisany w języku C++ przez Aarta J. C. Bika; Bikjump oferuje możliwość korzystania z bazy danych gier końcowych,
  - Cdrill 1800 build 4 – zaprojektowany przez Ferdinanda Moscę, ma za zadanie symulowanie gracza poniżej poziomu 2000 ELO,
  - Cinnamon 1.2c – napisany w języku C++ ze wsparciem wielu architektur i systemów; Cinnamon jest również dostępny do wykorzystania w stronach internetowych jako otwartoźródłowa biblioteka JavaScript,
  - Clarabit 1.00 – napisany w 2008 roku przez Salvadora Pallaresa Bejarano; Clarabit wykorzystuje "magiczne tablice bitów" [11], dzięki którym możliwe ruchy gońca i wieży mogą zostać obliczone równocześnie,
  - Maia-1900 – zmodyfikowana wersja LCZero przeznaczona do jak najlepszej symulacji ludzkiego gracza, w tym ludzkich błędów. Maia jest silnikiem wykorzystującym sieci neuronowe i oferuje poziomy trudności od 1100 do 1900 ELO,
  - Tarrasch ToyEngine Beta V0.906 – prosty silnik napisany w języku C++ przez Billa Forstera na potrzeby interfejsu użytkownika o tej samej nazwie (Tarrasch).
- Kategoria 2 (2501 – 2999 ELO):
  - Arminius 2017-01-01 – zaprojektowany przez Volkera Annussa, wykorzystuje sieć neuronową, oraz od wersji 2017, bardziej skomplikowaną wersję "magicznych tablic bitowych", zwaną "tablicami czarnej magii" [11],
  - Cheng 4.40 – napisany w języku C++ przez Martina Sedlaka ze wsparciem wielu systemów operacyjnych; Cheng wykorzystuje technikę zwaną "leniwym SMP" [12], która umożliwia wzajemne "podglądanie" postępu pracy różnych wątków podczas znajdywania ruchów,
  - Daydreamer 1.75 JA – napisany w języku C przez Aarona Beckera; nazwa "Daydreamer" pochodzi od zachowania pierwszych wersji silnika, które w losowych momentach popełniały ogromne błędy, przypominając graczy "zamyślonych"; początkowo silnik wykorzystywał uproszczoną funkcję ewaluacyjną Tomasza Michniewskiego, przypisując numeryczną wagę pionkom i polom szachownicy,
  - Demolito 32bit – opracowany w języku C przez Lucasa Braescha; Demolito wykorzystuje "okienko aspiracji" [13], które wydziela przestrzeń możliwych ruchów w pewnej odległości od zgadywanej wartości tak, aby możliwie przyspieszyć wyszukiwanie; jeśli optymalny ruch znajduje się poza "oknem", program wydziela "okno" z większym zakresem,
  - Gambitfruit Beta 4bx – napisany przez Ryana Beniteza w języku C++ i inspirowany silnikami Toga oraz Fruit; Gambitfruit ma za zadanie być wyjątkowo agresywnym przeciwnikiem,

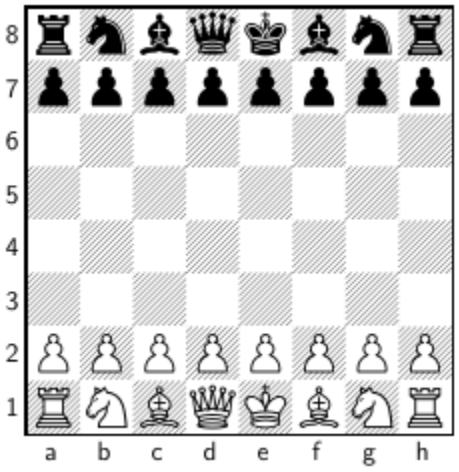
- Rybka 2.3.2a - rozwijany przez mistrza międzynarodowego Vasika Rajlicha, w latach 2006-2010 zdobywca czołowych miejsc międzynarodowych mistrzostw szachowych; śledztwo w 2011 roku zdyskwalifikowało Rybkę z późniejszych wydarzeń ze względu na oskarżenia o identyczność z silnikiem Fruit,
- Toga deepTogaNPS 1.9.6 – (początkowo sekretnie) oparty na silnik Fruit i zaprojektowany przez Fabien Letouzey; Toga po oskarżeniach o plagiat został wydany w wersji drugiej pod licancją GNU 2.0; najnowsze wydania Togi wykorzystują sieci neuronowe,
- Zappa 1.1 – zaprojektowany przez Anthony'ego Cozzie; po dyskwalifikacji w 2011 Zappa przejął część wygranych Rybki; na jego podstawie powstał komercyjny silnik szachowy Zap!Chess.

- Kategoria 3 (3000 ELO i więcej):
  - Andscacs 0.9432n – stworzony przez Daniela José Queraltó na przełomie 2013 i 2014 roku; na mistrzostwach WCRCC 2016 Andscacs otrzymał drugie miejsce nie przegrywając żadnej rozgrywki, co uniemożliwiło mistrzowi (Komodo) osiągnięcie perfekcyjnego wyniku [14],
  - Critter 1.6a – napisany w 2008 roku przez Richarda Vidę, początkowo w języku Pascal; w 2009 został przepisany na język C/C++; Critter jest w stanie wyszukiwać optymalny ruch używając do ośmiu wątków jednocześnie,
  - Gull 3 –stworzony przez ThinkingALot i inspirowany innymi silnikami szachowymi; od wersji 1.2 Gull jest opisany przez jeden plik źródłowy C++ i wspiera rekurencyjne wyszukiwanie ruchów; od wersji trzeciej dostępne są porty na systemy Linux i MacOS,
  - Hannibal 1.4b – stworzony przez Sama Hamiltona i Edsela Apostola ze wsparciem Audy Arandela; głównym celem Hannibala jest bycie najsilniejszym, jak to tylko możliwe; do wyszukiwania ruchów wykorzystuje on algorytm alfa-beta, bazy danych końcowych ruchów oraz tablice bitowe,
  - Houdini 1.5a – zaprojektowany w 2010 roku przez Roberta Houdarta i inspirowany innymi silnikami, w tym Stockfish; od wersji 2.0 wzwyż Houdini jest dostępny wyłącznie w wersjach płatnych, a starsze wersje są darmowe dla celów innych niż komercyjne,
  - Komodo 12.1.1 – stworzony w 2010 roku przez Dona Daileya, od 2013 rozwijany przez Marka Leflera; książka otwarć Komodo powstała przy współpracy z ekspertem, Erdoganem Günesem; od 2011 roku najnowsze wersje Komodo są oprogramowaniem płatnym; wcześniejsze wersje dostępne są na platformy Windows, Linux, MacOS oraz Android; na jego podstawie powstał silnik Dragon, nauczony maszynowo na rozgrywkach Komodo,
  - LCZero 0.27 – adaptacja projektu silnika do gry w japońską grę "Go" pod zasady gry w szachy napisana w języku C++; LCZero jest otwartoźródłowy i ma na celu połączenie metody wyszukiwania Monte Carlo z technikami głębokich sieci neuronowych; do najbardziej optymalnego wykorzystania tych sieci polecane jest uruchamianie LCZero na karcie graficznej z rdzeniami CUDA [15],
  - Stockfish 14.1 – napisany w języku C++ i wydany pod otwartoźródłową licencją GPL 3.0, od 2018 roku faktycznie najsilniejszy powszechnie dostępny silnik szachowy; od 2019 roku, dzięki wsparciu ze strony społeczności skupionej wokół szacho-podobnej gry "Shogi", Stockfish uzyskał możliwość wykorzystania sieci neuronowych; Stockfish jest dostępny na platformach Windows, Linux, MacOS, iOS, Android oraz w prawie wszystkich serwisach oferujących grę szachową i szacho-podobną on-line.

3.1. Badanie liczby wygranych rozgrywek

Silniki szachowe były badane pod kątem liczby wygranych rozgrywek poprzez funkcję turniejów szachowych w programie Lucas Chess. Turnieje były przeprowadzane w postaci "każdy-z-każdym", przy czym dowolna para silników rozgrywała między sobą dokładnie trzy razy. Rozgrywki były objęte limitem czasowym, dając dokładnie jedną minutę na gracza. Silnikom przydzielano miejsce w rankingu na podstawie wyniku punktowego, gdzie wygrany dostaje 1 punkt, przegrany 0 punktów, a remisujący po połowie punktu. Przeprowadzono łącznie dziewięć turniejów, gdzie dla wcześniej wymienionych trzech kategorii przetestowano następujące startowe ułożenia pionków:

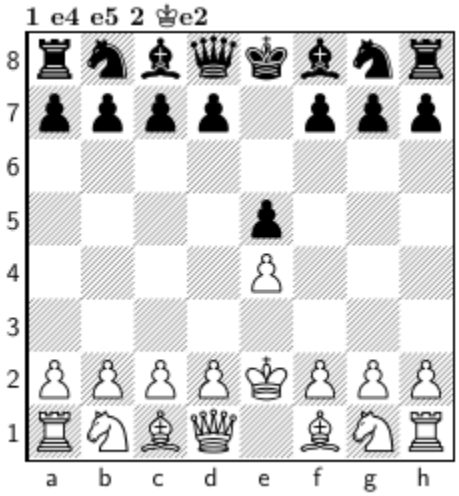
- klasyczna szachownica – standardowe ułożenie pionków według zasad gry w szachy, przedstawione na Rysunku 1,



Rysunek 1: Standardowe ułożenie szachownicy.

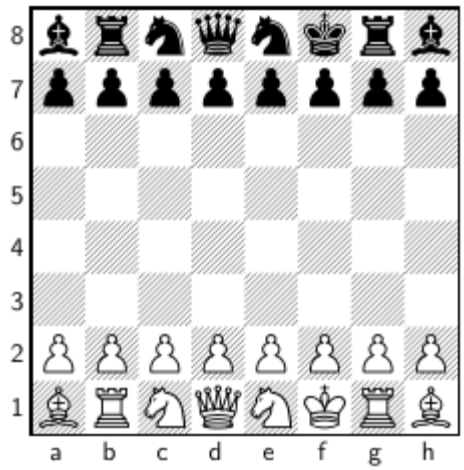
- otwarcie "Bongcloud" – przedstawione na Rysunku 2 i złożone z ruchów 1.e4 e5 oraz 2.Ke2, jest najgorszym rozpoczęciem rozgrywki z perspektywy gracza białego [16]; przesunięcie króla do przodu uniemożliwia późniejszą roszadę i blokuje hetmana oraz gońca na polu f1; uważa się, że „Bongcloud" został

wynaleziony przez jednego z użytkowników popularnego serwisu szachowego Chess.com o pseudonimie "Lenny_Bongcloud",



Rysunek 2: Otwarcie „Bongcloud".

- Chess960 – wariant szachowy stworzony przez Bobby'ego Fischera, w którym pozycja figur na pierwszej i ósmej linii szachownicy jest generowana losowo; aby rozgrywka była bardziej zbalansowana, obydwaj gracze mają takie samo ustawienie figur, gońce stoją na polach o przeciwnym kolorze, a król znajduje się pomiędzy wieże; w ten sposób możliwych jest 960 różnych ustawień pionków; Rysunek 3 przedstawia wykorzystane do badań jedno z możliwych ustawień.



Rysunek 3: Jedno z ustawień Chess960 użyte w badaniu.

3.2. Badanie zapotrzebowania na CPU i RAM

Do sprawdzenia wykorzystania procesora i pamięci systemowej wykorzystany został własny skrypt napisany w języku Python. Zadaniem skryptu miało być pobranie wartości procentowej wykorzystania CPU i RAM z procesu silnika i przesłanie danych do pliku .csv. Procesy wyszukiwane były po nazwie, którą należało odczytać z „menadżera zadań" systemu Windows. Lista nazw procesów wyszukiwanych przez skrypt opisana jest przez Listing 1. Proces o nazwie „maia" nie jest na liście, ponieważ silnik Maia jest oparty LCZero i system „widzi" go pod tą samą nazwą.

Listing 1: Lista procesów badanych silników szachowych

```
nazwy = [
  'CDrill_1800_Build_4.exe', 'cinnamon_1.2c-generic.exe',
  'lc0.exe', 'TarraschToyEngineV0.906.exe',
  'bikjump.exe', 'clarabit_100_x32_win.exe',
  'Arminius2017-01-01-32Bit.exe', 'cheng4.exe',
  'daydreamer-175-32-ja.exe', 'demolito_32bit_old.exe',
  'gfruit.exe', 'Rybka v2.3.2a.w32.exe',
  'DeepToga1.9.6nps.exe', 'zappa.exe',
  'andscacs_32_no_popcnt.exe', 'Critter_1.6a_32bit.exe',
  'Gull 3 w32 XP.exe', 'Hannibal1.4bx32.exe',
  'Houdini_15a_w32.exe', 'komodo-12.1.1-64-bmi2.exe',
  'Stockfish-14_1_x64-bmi2.exe'
]
```

Ciało skryptu widoczne na Listingu 2 składa się z pętli nieskończonej sprawdzającej dla każdego elementu listy, czy istnieje proces w systemie o takiej samej nazwie. Następnie, w przypadku zgodności nazw, otwierany jest plik .csv o nazwie takiej samej jak proces (jeśli takowego pliku nie ma, to jest on tworzony automatycznie). Do pliku dopisywane są informacje o procentowym użyciu procesora i pamięci systemowej oraz znak końca linii. Po zamknięciu pliku skrypt czeka sekundę, aby następnie przejść do kolejnej iteracji pętli. Dostęp do procesów systemowych umożliwia biblioteka psutil [17], a do oczekiwania użyto biblioteki time. Skrypt działał podczas rozgrywki silników w turniejach szachowych.

Listing 2: Ciało pętli zapisującej dane do plików

```
while True:
  for n in nazwy:
    for p in psutil.process_iter():
      if p.name() == n:
        f = open(p.name()+".csv", "a")
        f.write(str(p.cpu_percent())+','+str(p.memory_percent()))
        f.write("\n")
        f.close()
  time.sleep(1)
```

4. Wyniki

Informacje o silnikach i rozegranych partiach w turniejach szachowych są dostępne w programie Lucas Chess. Po pomyślnym przebiegu turnieju generowana jest tabela ze szczegółowymi informacjami na temat wygranych, przegranych i remisów. Dodatkowo program oferuje eksport danych o turnieju do pliku o własnościowym formacie .mvm. Do badań wykorzystano liczby wygranych łączne, wygranych jako gracz biały, wygranych jako gracz czarny oraz punkty ELO.

Dane o zużyciu zasobów komputera zostały przeniesione z plików .csv do arkusza kalkulacyjnego, gdzie wyliczono maksymalne zużycie, średnie zużycie oraz odchylenie standardowe dla procesora i pamięci RAM.

4.1. Kategoria 1

Turniej z szachownicą klasyczną zakończył się zwycięstwem silnika Carabit. Każdy silnik szachowy rozegrał 30 gier, z czego Clarabit i Bikjump wygrywali średnio co drugą rozgrywkę. Ze względu na pierwszeństwo ruchu, liczba wygranych jako gracz biały jest nieznacznie większa od liczby wygranych jako czarny we

wszystkich przypadkach, oprócz silnika Tarrasch, który przegrał prawie każdą grę. Liczba wygranych jest wprost proporcjonalna do współczynnika ELO, co widoczne jest w Tabeli 2.

Tabela 2: Wyniki turnieju szachowego kategorii 1 na klasycznej szachownicy

Silnik	Wygrane łącznie	Wygrane jako biały	Wygrane jako czarny	ELO
Clarabit	21	11	10	2058
Bikjump	19	10	9	2026
Cinnamon	17	9	8	1930
Maia	13	7	6	1900
Cdrill	11	5	6	1800
Tarrasch	1	0	1	1481

Przejście z pozycji startowej na otwarcie „Boncgcloud" przerzuca przewagę na gracza czarnego, co jest najbardziej widoczne dla „słabszych" silników. Wyjątkiem stanowi Cinnamon, wygrywający częściej jako bracz biały. Silnik Cdrill wykazał największą różnicę wyników wobec wcześniejszego turnieju, ogólnie wygrywając o 3 gry więcej, co przeniosło go w tabeli na miejsce czwarte. Maia, z taką samą liczbą całkowitych wygranych co wcześniej, spadł na miejsce piąte, co widać w Tabeli 3.

Tabela 3: Wyniki turnieju szachowego kategorii 1 rozpoczynając od otwarcia „Bongcloud"

Silnik	Wygrane łącznie	Wygrane jako biały	Wygrane jako czarny	ELO
Clarabit	22	11	11	2058
Bikjump	20	9	11	2026
Cinnamon	14	8	6	1930
Cdrill	14	7	7	1800
Maia	13	4	9	1900
Tarrasch	2	0	2	1481

W turnieju Chess960 zwycięzcą został silnik Bikjump, wygrywając dwie z trzech rozgrywek. Tak jak dla szachownicy klasycznej, obydwaj gracze mieli szanse zbliżone do równych. Wyjątkiem stanowią Maia i Tarrasch. Silnik Maia, tak jak w przypadku otwarcia „Bongcloud", wygrywał częściej jako gracz czarny. Tarrasch, dzięki nietypowemu ułożeniu pionków, udało się wygrać siedem rozgrywek, z czego pięć jako gracz biały. Wygrane silników dla Chess960 widoczne są w Tabeli 4.

Tabela 4: Wyniki turnieju szachowego kategorii 1 dla Chess960

Silnik	Wygrane łącznie	Wygrane jako biały	Wygrane jako czarny	ELO
Bikjump	20	9	11	2026
Clarabit	15	7	8	2058
Cinnamon	16	8	8	1930
Maia	12	3	9	1900
Cdrill	11	6	5	1800
Tarrasch	7	5	2	1481

Silniki z kategorii pierwszej wykazują się znikomym zapotrzebowaniem na RAM, co widoczne jest w Tabeli 5. Tarrasch i Cdrill pobierają mniej niż jeden promil z 16 GB dostępnej pamięci. Wszystkie badane silniki są napisane w postaci aplikacji C/C++ bez interfejsu użytkownika, tak więc pamięć wykorzystywana jest głównie do tablic i obliczeń numerycznych. Inaczej wygląda zużycie procesora. Tylko Cinnamon i Clarabit nie wykorzystywały chwilowo w pełni jednego z wątków.

Tabela 5: Wyniki pomiarów wydajności dla kategorii 1

Silnik	CPU śr. [%]	CPU maks. [%]	RAM śr. [‰]	RAM maks. [‰]
Bikjump	49,6	100,0	1,244	1,252
Cdrill	51,4	100,0	0,558	0,562
Cinnamon	42,6	81,2	2,298	2,303
Clarabit	43,7	83,0	2,774	2,776
Maia	46,1	92,3	1,841	4,361
Tarrasch	45,9	100,0	0,247	0,248

4.2. Kategoria 2

W przeciwieństwie do innych kategorii, silniki z przedziału 2501-2999 ELO są na podobnym poziome, co przekłada się na zróżnicowane wyniki rozgrywek. Każdy silnik w tej kategorii rozegrał łącznie 42 rozgrywki. Dla szachownicy klasycznej zwycięzcą został Cheng z niewielką przewagą wobec Rybki, co przedstawionej jest w Tabeli 6. Silnik Demolito, pomimo dość wysokiej liczby wygranych, uzyskał przedostatnie miejsce ze względu na to, że przegrał 20 z pozostałych 26 gier.

Tabela 6: Wyniki turnieju szachowego kategorii 2 na klasycznej szachownicy

Silnik	Wygrane łącznie	Wygrane jako biały	Wygrane jako czarny	ELO
Cheng	25	13	12	2750
Rybka	23	12	11	2936
Toga	15	9	6	2843
Dayderamer	12	7	5	2670
Gambitfruit	16	9	7	2750
Arminius	12	7	5	2662
Demolito	16	15	1	2627
Zappa	2	2	0	2581

Rozpoczynając od otwarcia „Bongcloud", większość silników w kategorii drugiej radziła sobie zdecydowanie lepiej jako gracz czarny. Wyjątek stanowi Demolito, który wygrywał wyłącznie jako gracz biały, co widoczne jest w Tabeli 7. Największą liczbę wygranych uzyskał Rybka. Silnik Cheng, pomimo zdobycia „tylko" 20 wygranych, uzyskał wyższy wynik od Togi ze względu na większą liczbę remisów. Analogicznie, Arminius znajduje się na wyższej pozycji od silnika Daydreamer z tego samego powodu.

W przypadku Chess960 silnik Rybka zdominował kategorię drugą wygrywając 3 na 4 rozgrywki. Toga, nauczony maszynowo głównie na rozgrywkach tradycyjnych, spadł z trzeciego miejsca na szóste, za sprawą wielu przegranych. Silnik Demolito, tak jak wcześniej,

faworyzuje rozgrywkę jako gracz biały, co widoczne jest w Tabeli 8.

Tabela 7: Wyniki turnieju szachowego kategorii 2 rozpoczynając od otwarcia „Bongcloud”

Silnik	Wygrane łącznie	Wygrane jako biały	Wygrane jako czarny	ELO
Rybka	23	8	15	2936
Cheng	20	4	16	2750
Toga	22	9	13	2843
Demolito	20	20	0	2627
Gambitfruit	14	4	10	2750
Arminius	9	4	5	2662
Daydreamer	11	5	6	2670
Zappa	8	3	5	2581

Tabela 8: Wyniki turnieju szachowego kategorii 2 dla Chess960

Silnik	Wygrane łącznie	Wygrane jako biały	Wygrane jako czarny	ELO
Rybka	32	17	15	2936
Cheng	21	11	10	2750
Demolito	20	20	0	2627
Gambitfruit	16	11	5	2750
Arminius	9	4	5	2662
Toga	15	7	8	2843
Daydreamer	11	5	6	2670
Zappa	5	3	2	2581

Pomiary wydajności pokazały, że silniki szachowe z kategorii 2 potrzebują więcej zasobów systemowych niż „słabsze” odpowiedniki. Wszystkie silniki, oprócz Demolito, momentalnie osiągały 100 procent zużycia CPU. Najmniejsze średnie zapotrzebowanie na wątek procesora wykazuje Rybka. Tak jak wcześniej, pamięć RAM komputera nie jest wykorzystywana w znacznym stopniu. Najbardziej pamięciożerny jest silnik Zappa, wykorzystując do pół procenta pamięci systemu, co widoczne jest w Tabeli 9.

Tabela 9: Wyniki pomiarów wydajności dla kategorii 2

Silnik	CPU śr. [%]	CPU maks. [%]	RAM śr. [‰]	RAM maks. [‰]
Arminius	45,0	100,0	1,549	1,554
Cheng	48,4	100,0	1,421	1,426
Daydreamer	47,7	100,0	2,819	2,839
Toga	43,9	100,0	1,334	1,407
Demolito	47,1	95,6	1,341	1,346
Gambitfruit	49,3	100,0	1,275	1,278
Rybka	40,4	100,0	1,422	1,424
Zappa	41,1	100,0	1,947	4,883

4.3. Kategoria 3

Podobnie jak dla kategorii pierwszej, silniki powyżej 3000 ELO uzyskały wyniki takie, jak ich „siła w punktach”. Każdy silnik rozegrał 42 rozgrywki, z czego ogólny zwycięzca, Stockfish, wygrał aż 39. Niska pozycja Gull, tak jak w podobnych przypadkach wcześniej, wynika z dużej liczby przegranych. Niewielka preferencja wobec gracza białego także tu występuje, co widoczne jest w Tabeli 10.

Tabela 10: Wyniki turnieju szachowego kategorii 3 na klasycznej szachownicy

Silnik	Wygrane łącznie	Wygrane jako biały	Wygrane jako czarny	ELO
Stockfish	39	20	19	3500
Komodo	27	14	13	3300
LCZero	14	8	6	3300
Andscacs	7	4	3	3264
Gull	10	6	4	3125
Critter	6	5	1	3091
Houdini	8	5	3	3093
Hannibal	4	3	1	3000

Zmiana przewagi między graczami poprzez otwarcie „Bongcloud” wywarło największy wpływ na silniki z kategorii trzeciej, co widać w Tabeli 11. Tylko Stockfish, Komodo i LCZero były w stanie wygrać więcej niż dwie rozgrywki na 21 możliwych jako gracz biały. Przewagę gracza czarnego efektywnie wykorzystał silnik Houdini, „przebijając” rywali z większą liczbą punktów ELO.

Tabela 11: Wyniki turnieju szachowego kategorii 3 rozpoczynając od otwarcia „Bongcloud”

Silnik	Wygrane łącznie	Wygrane jako biały	Wygrane jako czarny	ELO
Stockfish	35	14	21	3500
Komodo	30	10	20	3300
LCZero	13	5	8	3300
Houdini	12	1	11	3093
Critter	11	2	9	3091
Andscacs	4	0	4	3264
Gull	12	2	10	3125
Hannibal	6	0	6	3000

Turniej Chess960 dla kategorii trzeciej przebiegał podobnie do turnieju z szachownicą klasyczną z tą różnicą, że wcześniej wysoko pozycjonowany silnik LCZero przestał wygrywać. Wynika to z budowy silnika – tak jak wcześniej opisane Maia i Toga, LCZero został wytrenowany maszynowo wyłącznie na szachownicy klasycznej. Podobnie jak algorytm do rozpoznawania obrazów nie dający sobie rady, gdy wejście jest obrócone lub odbite w lustrze, silnikowi LCZero udało się wygrać tylko 6 na 42 rozgrywki Chess960, co widoczne jest w Tabeli 12.

Silniki z kategorii trzeciej wykazują się względnie wysokim zapotrzebowaniem na pamięć RAM, co widoczne jest w Tabeli 13. Jedynie silniki Critter i Andscacs pobierały zasoby systemowe w liczbie porównywalnej do kategorii poprzednich. Stockfish, LCZero i Komodo zostały napisane pod architekturę 64-bitową oraz domyślnie korzystały z dwóch wątków, co przełożyło się na większe zapotrzebowanie na zasoby. Wartość powyżej stu procent przy zużyciu CPU oznacza, że program korzysta z więcej niż jednego wątku.

Tabela 12: Wyniki turnieju szachowego kategorii 3 dla Chess960

Silnik	Wygrane łącznie	Wygrane jako biały	Wygrane jako czarny	ELO
Stockfish	36	16	20	3500
Komodo	30	16	14	3300
Gull	11	5	6	3125
Andscacs	7	4	3	3264
Critter	9	5	4	3091
Houdini	8	4	4	3093
LCZero	6	2	4	3300
Hannibal	6	3	3	3000

Tabela 13: Wyniki pomiarów wydajności dla kategorii 3

Silnik	CPU śr. [%]	CPU maks. [%]	RAM śr. [‰]	RAM maks. [‰]
Andscacs	47,6	88,8	2,927	2,932
Critter	42,0	99,6	1,649	1,652
Gull	48,4	100,0	4,780	4,795
Hannibal	48,7	100,0	7,476	7,480
Houdini	44,3	100,0	2,475	2,478
Komodo	90,4	194,5	8,938	8,963
LCZero	90,8	190,5	2,589	2,783
Stockfish	83,5	200,0	9,624	9,718

5. Wnioski

- Dla kategorii 1:
  - „najmocniejszym" silnikiem okazał się być Bikjump, wygrywając łącznie 59 gier na 90 możliwych, Clarabit stanął na drugim miejscu z 58 wygranymi, Cinnamon trzeci z liczbą zwycięstw równą 49,
  - najmniejsze zużycie pamięci RAM wykazywał Cdrill, Cinnamon najmniej wykorzystywał procesor.
- Dla kategorii 2:
  - największą liczbę wygranych osiągnął silnik Rybka (78 ze wszystkich 126 gier), na drugim miejscu Cheng z 66 wygranymi, trzeci Demolito, wygrywając 56 rozgrywek,
  - najmniejsze zużycie RAM wykazywał Gambitfruit, Rybka z najniższym średnim zapotrzebowaniem na procesor, Demolito z najniższym zużyciem maksymalnym.
- Dla kategorii 3:
  - „najsilniejszym" był Stockfish, osiągając 110 zwycięstw na 126 rozgywek, na drugim miejscu stanął Komodo z 87 zwycięstwami, silniki LCZero i Gull osiągnęły ex aequo trzecie miejsce z 33 wygranymi rozgrywkami każdy,
  - silnik Critter zużywał w najmniejszym stopniu pamięć systemową i procesor.
- Dodatkowo:
  - silniki szachowe wykorzystują procentowo znacznie mniej pamięć RAM niż procesor,
  - losowanie pionków w Chess960 powoduje, że silniki szachowe zbudowane w oparciu o uczenie maszynowe i sieci neuronowe radzą sobie zdecydowanie gorzej niż w przypadku szachownicy klasycznej,
  - silnik Demolito ma wysoką różnicę umiejętności w zależności od strony szachownicy – ze wszystkich 56 wygranych tylko jedna z nich nie była zwycięstwem jako gracz biały,
  - dokonane porównanie nie jest w pełni dokładne – przyszłe badania mogłyby uwzględnić różne limity czasowe podczas rozgrywki.

Literatura

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

[2] International Chess Federation - strona główna, https://www.fide.com/, [25.05.2022].

[3] V. V. Vučković, Realization of the Chess Mate Solver Application., Yugoslav Journal of Operations Research 14 (2004) 273-288, https://doi.org/10.2298/YJOR0402273V.

[4] W. B. Putra, L. Heryawan, Applying Alpha-beta Algorithm In A Chess Engine, Jurnal Teknosains UGM 6 (2016) 37-43.

[5] H. Zang, Z. Yu, X. Wan, Automated chess commentator powered by neural chess engine, arXiv (2019), https://doi.org/10.48550/arXiv.1909.10413.

[6] M. Block, M. Bader, E. Tapia, M. Ramírez, K. Gunnarsson, E. Cuevas, D. Zaldivar, R. Rojas, Using Reinforcement Learning in Chess Engines, Research in Computing Science 35 (2008) 31-40.

[7] N. Hesham, O. Abu-Elnasr, S. Elmougy, A New Action-Based Reasoning Approach for Playing Chess, Computers, Materials and Continua 69 (2021) 175-190.

[8] S. K. Bimonugroho, N. U. Maulidevi, A Hybrid Approach to Representing Chessboard using Bitboard and Compact Chessboard Representation, IOP Conference Series: Materials Science and Engineering 803 (2020), https://doi.org/10.1088/1757-899X/803/1/012018.

[9] S. Maharaj, N. Polson, A. Turk, Chess AI: Competing Paradigms for Machine Intelligence, Entropy 24 (2022) 550, https://doi.org/10.48550/arXiv.2109.11602.

[10] Strona internetowa programu Lucas Chess, https://lucaschess.pythonanywhere.com/home, [25.05.2022].

[11] Magiczne tablice bitów - definicja, https://www.chessprogramming.org/Magic_Bitboards, [25.05.2022].

[12] Leniwe SMP - definicja, https://www.chessprogramming.org/Lazy_SMP, [25.05.2022].

[13] Okno aspiracji - definicja, https://www.chessprogramming.org/Aspiration_Windows, [25.05.2022].

[14] Mistrzostwa ACCA World Computer Rapid Chess Championship 2016, https://www.chessprogramming.org/WCRCC_2016, [25.05.2022].

[15] B. Steinbach, M. Werner, XBOOLE-CUDA -- Fast Boolean Operations on the GPU (2014).

[16] Double bongcloud: why grandmasters are playing the worst move in chess, https://www.theguardian.com/sport/2021/mar/18/bongcloud-meme-opening-carlsen-nakamura, [20.06.2022]

[17] Biblioteka psutil - dokumentacja, https://psutil.readthedocs.io/en/latest/, [25.05.2022].