Un'Analisi Architetturale e Metodologica del Motore Scacchistico ShashChess

Sezione 1: Decostruzione Architetturale e Analisi a Livello di Codice di ShashChess

Questa sezione esegue un'analisi approfondita del codice sorgente e delle funzionalità dichiarate di ShashChess, valutando la traduzione della teoria scacchistica di Alexander Shashin in un motore funzionale e ad alte prestazioni. Verranno esaminate la novità, la qualità dell'implementazione e il potenziale impatto di ciascuna delle principali modifiche architetturali.

1.1 Il Nucleo Teorico: Implementazione di "Best Play" di Shashin in evaluate.cpp

L'obiettivo primario di ShashChess è implementare la teoria di Alexander Shashin, che classifica le posizioni in archetipi basati sullo stile di gioco di grandi campioni: Tal (attaccante), Capablanca (bilanciato) e Petrosian (difensivo), includendo anche tipologie miste.[1] Un'analisi delle risorse disponibili, in particolare da

chessprogramming.org, rivela un'implementazione diretta e pragmatica di questa teoria. Il motore assegna una "personalità" alla posizione basandosi su intervalli di punteggio della valutazione statica, derivati dal valore di un pedone nel finale (PawnValueEg = 208).[2] La logica è la seguente:

- Se valutazione < -74, la personalità è Petrosian.
- Se valutazione < 31, la personalità è Capablanca.
- Altrimenti, la personalità è Tal, con stati misti per i valori intermedi.

Questa modifica è fondamentale. A differenza della valutazione monolitica di Stockfish,

ShashChess introduce un modello di valutazione dipendente dallo stato, che mira a migliorare la consapevolezza situazionale del motore. La logica centrale nel file evaluate.cpp viene presumibilmente modificata per applicare pesi o euristiche differenti in base alla "personalità" assegnata. Le note di rilascio per la versione 39 menzionano un "Algoritmo di Resilienza Termodinamica" che calcola entropia ed energia interna, suggerendo un'estensione altamente astratta di questa idea centrale, nel tentativo di quantificare la stabilità e il potenziale posizionale in modo ispirato alla fisica.[4]

Questa architettura introduce un livello di controllo euristico al di sopra di una base neurale. ShashChess dichiara di utilizzare una "collaborazione tra Leela Chess Zero e Stockfish" per la sua funzione di valutazione [1], un riferimento quasi certo a un'architettura NNUE (Efficiently Updatable Neural Network), la stessa tecnologia adottata da Stockfish a partire dal 2020.[5] Tuttavia, le "personalità" di Shashin si basano su punteggi classici in centipedoni.[2] Ciò implica un processo a più stadi: in primo luogo, la rete NNUE fornisce una valutazione di base, un singolo numero altamente raffinato. Successivamente, la logica di Shashin utilizza questo numero per classificare la posizione in uno stato discreto (Tal, Capablanca, Petrosian). Infine, questo stato agisce probabilmente come un interruttore che attiva o disattiva altre logiche del motore, come estensioni di ricerca, euristiche di potatura o persino il modulo MCTS. Di conseguenza, ShashChess non sostituisce la NNUE, ma costruisce un livello di controllo euristico sopra di essa. Si tratta di un tentativo di reintrodurre la conoscenza esperta e la guida strategica in un sistema che è diventato sempre più dipendente da reti neurali "black-box". Il successo del motore dipende quindi dalla sinergia o dalla rottura tra questo strato euristico e i calcoli finemente sintonizzati della NNUE sottostante.

1.2 Un Paradigma di Ricerca Ibrido: Integrazione Condizionale di MCTS in search.cpp

ShashChess include una sezione sperimentale di Monte Carlo Tree Search (MCTS), basata sul lavoro di Stephan Nicolet, che viene attivata specificamente per le posizioni di tipo "Petrosian alto, medio-alto e medio".[1] Questa funzionalità è controllata dall'opzione UCI

MCTS by Shashin. Quando abilitata, il motore commuta il suo algoritmo di ricerca dal predefinito alpha-beta pruning a MCTS in condizioni specifiche, determinate dalla logica di valutazione di Shashin. Ciò richiede modifiche significative al file search.cpp, creando una biforcazione condizionale nel processo di pensiero primario del motore. L'obiettivo dichiarato è sfruttare i punti di forza percepiti di Leela Chess Zero (un motore basato su MCTS) in posizioni tranquille e strategiche.[1]

Questa è probabilmente la caratteristica tecnica più ambiziosa del progetto. Alpha-beta e

MCTS sono paradigmi di ricerca fondamentalmente diversi. L'alpha-beta è profondo e stretto, eccellendo nel calcolo tattico. MCTS è ampio e probabilistico, spesso migliore nella pianificazione strategica dove una valutazione precisa è difficile. L'ipotesi è che per le posizioni difensive/strategiche "Petrosian", l'approccio MCTS produrrà risultati migliori.

Tuttavia, questa integrazione rappresenta una scommessa architetturale ad alto rischio e alta ricompensa. La funzionalità MCTS è etichettata come "sperimentale" [1], e alcuni post sui forum indicano potenziali problemi di compilazione o integrazione.[7] Il motore Stockfish di base è un ricercatore alpha-beta iper-ottimizzato, con ogni componente (ordinamento delle mosse, potatura, ecc.) messo a punto per quello specifico algoritmo.[8] L'aggiunta di un modulo MCTS a un motore alpha-beta è architetturalmente complessa; le strutture dati, i selettori di mosse e le chiamate di valutazione non sono direttamente compatibili. Il sovraccarico legato al passaggio tra le modalità di ricerca potrebbe essere significativo, sprecando tempo di calcolo prezioso. Inoltre, l'efficacia della ricerca MCTS dipende fortemente dalla qualità della sua politica di playout, che in questo caso sembra essere legata alla valutazione del motore di base, creando un complesso ciclo di feedback. Il fatto che venga attivato solo in posizioni specifiche suggerisce il riconoscimento che MCTS non è universalmente migliore. Tuttavia, la condizione di attivazione (un punteggio di valutazione statica) è un proxy grezzo per "una posizione che si adatta a MCTS". Una valutazione posizionale veramente profonda sarebbe molto più complessa. Questa caratteristica è una fonte di immenso rischio tecnico e potenziale instabilità delle prestazioni, ma anche un'area privilegiata per future ottimizzazioni e ricerche.

1.3 Meccanismi Adattivi: BrainLearn e Integrazione della Conoscenza Live

ShashChess incorpora BrainLearn, un algoritmo di apprendimento per rinforzo di tipo Q/Self-learning che aggiorna una tabella hash quando vengono trovati punteggi migliori a una data profondità, e una funzione "LiveBook" che può interrogare database esterni come Lichess e ChessDB.[1]

BrainLearn è progettato per controlli di tempo non-bullet e richiede un file persistente per funzionare.[1] Agisce come una memoria a lungo termine, affinando le valutazioni basate sui risultati della ricerca del motore stesso. La funzionalità LiveBook, con le sue ampie opzioni UCI (

LiveBook Proxy Url, LiveBook Lichess Games, ecc.), consente al motore di attingere a enormi database di partite umane e di motori, espandendo di fatto il suo libro di aperture e la conoscenza del mediogioco in tempo reale.[1]

Queste caratteristiche spostano ShashChess da un motore di puro calcolo verso un sistema di apprendimento, particolarmente adatto ai controlli di tempo lunghi. Esiste un chiaro allineamento filosofico tra le caratteristiche implementate e la metodologia di test preferita. La funzione BrainLearn è esplicitamente indicata come efficace solo in partite più lunghe di 5 minuti.[1] La funzione LiveBook introduce una latenza di rete, che è meno impattante in partite più lunghe. Un tipico test

fishtest a 10+0.1 renderebbe BrainLearn quasi inutile e il LiveBook una potenziale passività. Al contrario, in una partita in stile corrispondenza (1 minuto per mossa o più), queste caratteristiche diventano potenti risorse, consentendo al motore di imparare all'interno di una singola partita e di attingere a vaste conoscenze esterne. Ciò rivela una filosofia di design coerente. La presunta superiorità di ShashChess potrebbe non derivare solo dalla valutazione di Shashin, ma da una combinazione sinergica di caratteristiche che eccellono nelle specifiche condizioni di test utilizzate. Questo solleva una domanda critica: il motore è globalmente più forte, o è uno "specialista della resistenza" ottimizzato per un meta-game a lungo controllo di tempo?

Sezione 2: Una Valutazione Critica della Metodologia di Test e della Validità Statistica

Questa sezione analizza rigorosamente il protocollo di test dell'utente e la sua critica al framework fishtest. Verranno convalidati i punti legittimi sollevati, fornendo al contempo una valutazione statisticamente fondata dei dati del match forniti, affrontando la questione centrale se la superiorità dichiarata sia statisticamente provata.

2.1 Il "Dilemma Fishtest": Convalida della Critica allo Status Quo

L'utente sostiene che fishtest sia imperfetto perché la sua dipendenza da controlli di tempo ultra-rapidi e dalla posizione di partenza standard porta a un'alta percentuale di patte, mascherando i progressi reali. La ricerca supporta ampiamente questa premessa. Ai massimi livelli dei motori, con controlli di tempo lunghi e partendo dalla posizione iniziale, il tasso di patta è straordinariamente alto, spesso citato come >99%.[11] Questo fenomeno, noto come "morte per patta" ("draw death"), è la ragione principale per cui le metodologie di test si sono evolute. Anche al più alto livello del gioco umano, i tassi di patta sono significativi, superando il 50% negli eventi classici e aumentando ulteriormente nei match di campionato.[13]

Questa convalida è cruciale. La motivazione principale dell'utente per sviluppare una metodologia di test alternativa si basa su un problema corretto e ampiamente riconosciuto negli scacchi computerizzati. L'uso da parte del framework fishtest di controlli di tempo rapidi [16] e di libri di apertura curati e sbilanciati [17] non è una svista, ma un

compromesso pragmatico per sfuggire alla morte per patta e generare un numero sufficiente di partite decisive per raggiungere la significatività statistica in un tempo ragionevole.[18]

Sia la metodologia dell'utente che quella di fishtest cercano di risolvere lo stesso problema: come ottenere un "segnale" chiaro (la differenza di Elo) attraverso il "rumore" (la varianza casuale e gli alti tassi di patta). L'approccio dell'utente mira ad aumentare la *forza del segnale* per partita. Scegliendo posizioni "pepite" (acute), la probabilità di un risultato decisivo (un segnale forte) è molto più alta. L'approccio di fishtest, d'altra parte, diminuisce il *livello di rumore* del test complessivo. Giocando decine di migliaia di partite per una singola patch, la legge dei grandi numeri attenua la varianza casuale, permettendo di rilevare in modo affidabile anche un segnale molto debole.[5] La critica dell'utente a

fishtest è quindi valida nella diagnosi del problema ma incompleta nella comprensione della soluzione. Fishtest non è "non scientifico"; è un approccio statistico diverso e su scala massiccia alla stessa sfida. Il metodo dell'utente è valido ma opera su una scala molto più piccola, il che ha profonde implicazioni per la confidenza statistica.

2.2 Significatività Statistica e Confidenza: Un'Analisi Rigorosa del Match di 300 Partite

I risultati forniti indicano che ShashChess ha ottenuto +67 vittorie, 175 patte e 58 sconfitte contro Stockfish in 300 partite. Ciò corrisponde a un punteggio di 154.5 / 300, ovvero il 51.50%. Si sostiene che questo sia sufficiente per eliminare il rumore statistico, con una stima di un guadagno di +10 Elo. Questa affermazione deve essere sottoposta a rigorosi test statistici.

1. **Calcolo della Differenza di Elo**: Utilizzando la formula Elo standard, $\Delta = -400 \cdot \log_{10}(1/\text{punteggio} - 1)$, un punteggio del 51.50% corrisponde a una differenza di Elo di **+10.4 Elo**. La stima dell'utente è accurata.[19]
2. **Calcolo della Probabilità di Superiorità (LOS - Likelihood of Superiority)**: La LOS misura la probabilità che ShashChess sia veramente più forte, dato il risultato. La formula è $\text{LOS} = \frac{1}{2}[1 + \text{erf}((\text{vittorie} - \text{sconfitte})/2 \cdot (\text{vittorie} + \text{sconfitte}))]$.[21]
    - Vittorie = 67, Sconfitte = 58.
    - vittorie - sconfitte = 9.
    - vittorie + sconfitte = 125.

    - LOS=½[1+erf(9/2·125)]=½[1+erf(9/15.81)]≈0.83.
    - La LOS è di circa l'**83%.** Ciò significa che c'è un'83% di probabilità che ShashChess sia più forte e un 17% di probabilità che il risultato sia dovuto alla fortuna (cioè che Stockfish sia di forza uguale o superiore). Sebbene promettente, questo valore è ben al di sotto del livello di confidenza del 95% tipicamente richiesto per la significatività statistica.
3. **Calcolo del Margine di Errore (MoE)**: Utilizzando un calcolatore Elo con un livello di confidenza del 95% per il risultato del match dato (V=67, S=58, P=175), il margine di errore è di circa **±21 Elo**.[23]

La conclusione dell'utente è plausibile ma statisticamente non provata. Il guadagno di Elo calcolato è di +10.4, ma l'intervallo di confidenza al 95% è +10.4±21.0 Elo, che corrisponde a un intervallo da **-10.6 Elo a +31.4 Elo**. L'affermazione centrale secondo cui il test di 300 partite è sufficiente per "eliminare il rumore statistico" è quindi errata. Il margine di errore è più del doppio della differenza di Elo misurata. Poiché l'intervallo di confidenza al 95% include lo zero (e anche valori negativi), non è possibile, con una confidenza del 95%, rifiutare l'ipotesi nulla che non vi sia alcuna differenza di prestazioni tra i motori. L'intelligenza artificiale che ha "confermato" la sufficienza di 300 partite era probabilmente un modello linguistico generico privo della conoscenza specifica del dominio delle statistiche dei motori scacchistici, dove le dimensioni degli effetti (guadagni di Elo) sono minuscole e richiedono campioni di dimensioni enormi per essere provate.[25] Il test ha generato un

segnale promettente, ma non un *risultato statisticamente significativo*. Fornisce prove per *giustificare ulteriori test su larga scala*, ma non una prova definitiva di superiorità.

Tabella 1: Margine di Errore in Elo vs. Numero di Partite Giocate (Livello di Confidenza del 95%)

Numero di Partite (N)	Punteggio Assunto	Tasso di Patta Assunto	Guadagno Elo Calcolato	Margine di Errore (± Elo)	Intervallo di Confidenza al 95%	Statisticamente Significativo?
300 (Test ShashChess)	**51.5%**	**58.3%**	**+10.4**	**± 21.0**	**-10.6 a +31.4**	**No**

1,000	51.5%	58.3%	+10.4	± 11.5	-1.1 a +21.9	No (limite)
5,000	51.5%	58.3%	+10.4	± 5.1	+5.3 a +15.5	**Sì**
15,000 (Tipico Fishtest)	51.5%	58.3%	+10.4	± 2.9	+7.5 a +13.3	**Sì**
50,000	51.5%	58.3%	+10.4	± 1.6	+8.8 a +12.0	**Sì**

2.3 La Suite di Risoluzione Problemi: Un Segnale Più Forte

ShashChess ha risolto 140 posizioni, mentre Stockfish ne ha risolte 130 da una suite di 256 posizioni complesse. Questo rappresenta un aumento del 7.7% nelle posizioni risolte (140/130). A differenza del gioco in partita, che è soggetto all'elevata varianza dei risultati, la risoluzione di problemi è una misura più diretta della pura capacità di calcolo e della precisione di valutazione in domini specifici.

Questo risultato corrobora il design architetturale del motore. Il motore è progettato, tramite la teoria di Shashin, per comportarsi in modo diverso e (presumibilmente) migliore in posizioni tattiche complesse ("Tal") o strategiche ("Petrosian"). La suite di test è descritta come "posizioni complicate" e "pepite". Esiste una chiara coerenza tra l'architettura del motore, specializzata per posizioni complesse, la suite di test, composta da posizioni complesse, e il miglioramento delle prestazioni osservato su questa specifica suite. Questa è la prova più forte presentata. Mentre il risultato del match è statisticamente ambiguo, il risultato della suite di problemi suggerisce fortemente che le modifiche architetturali hanno migliorato con successo le prestazioni del motore nel suo dominio di riferimento: posizioni complesse e non standard. Questo è un risultato significativo.

Sezione 3: Sintesi delle Prestazioni e Raccomandazioni Strategiche

Questa sezione finale sintetizza le analisi architetturali e metodologiche per fornire una valutazione olistica delle prestazioni e un percorso chiaro e attuabile per il futuro del progetto ShashChess.

3.1 Interpretare la Presunta Superiorità: Uno Specialista, Non (Ancora) un Campione

Le prove non supportano un'affermazione di superiorità *generale e statisticamente provata* rispetto alla linea di base di Stockfish. Il +10 Elo del match è una stima plausibile della forza del motore, ma l'elevato margine di errore significa che potrebbe essere inferiore, o addirittura negativo. Tuttavia, le ottime prestazioni sulla suite di problemi curata (140 vs 130) forniscono prove convincenti che ShashChess è uno *specialista di successo*. Le modifiche architetturali derivate dalla teoria di Shashin sembrano aver migliorato tangibilmente la capacità del motore di navigare in posizioni tattiche e strategiche complesse.

In sintesi, il lavoro svolto ha creato con successo un motore con una "personalità" distinta che dimostra prestazioni superiori nei tipi di posizioni mirate. Il risultato del match è incoraggiante ma richiede più dati per essere considerato conclusivo. Lo scetticismo della comunità notato nei forum [26] deriva probabilmente proprio da questo problema: affermazioni forti basate su metriche personalizzate e test statisticamente piccoli, un modello comune per i motori derivati.

3.2 Percorsi Strategici per lo Sviluppo Futuro

Perfezionamenti Architetturali

- **Affinare l'Attivazione MCTS**: L'attuale meccanismo di attivazione è uno strumento grossolano (punteggio di valutazione statica). Si potrebbe sviluppare un trigger più sfumato, basato su una combinazione di fattori come l'attività dei pezzi, la struttura pedonale e la sicurezza del re, per identificare meglio le posizioni veramente adatte a MCTS. Ciò ridurrebbe il rischio di invocare la costosa ricerca MCTS in situazioni non ottimali.

- **Isolare e Ottimizzare le Euristiche di Shashin**: Invece di un interruttore globale di "personalità", identificare i termini di valutazione specifici o le estensioni di ricerca che forniscono il maggior beneficio all'interno di ciascun archetipo di Shashin. Questi cambiamenti più piccoli e mirati sono più facili da testare e mettere a punto.
- **Potenziare il Modulo BrainLearn**: Indagare tecniche di apprendimento per rinforzo più avanzate. L'attuale approccio Q-learning è efficace ma potrebbe essere migliorato con un modello più sofisticato che generalizzi meglio tra le posizioni, piuttosto che fare affidamento sulla memorizzazione meccanica in una tabella hash.

Scalabilità e Convalida Metodologica

- **Abbracciare la Scala**: La priorità immediata è aumentare il numero di partite giocate. Per dimostrare il guadagno di +10 Elo, sarebbe necessario un match di almeno **5,000 partite** nelle stesse condizioni per raggiungere la significatività statistica (come mostrato nella Tabella 1). Si tratta di un'impresa significativa, ma è l'unico modo per rispondere definitivamente alla domanda sulla forza.
- **Diversificare le Suite "Pepite"**: Espandere il set di posizioni di apertura acute e di problemi complessi del mediogioco. Creare suite di test multiple e indipendenti per evitare l'"overfitting" del motore a un singolo set di problemi. Ciò rafforzerà l'affermazione di superiorità specializzata.
- **Adottare SPRT**: Per i test interni, adottare la metodologia Sequential Probability Ratio Test (SPRT) utilizzata da fishtest.[16] Questo è un metodo di test più efficiente che interrompe un match non appena si raggiunge una conclusione (superato, fallito o nullo) con una data confidenza, risparmiando un'enorme quantità di tempo CPU rispetto ai match a numero fisso di partite.

Unire le Metodologie: Il Percorso verso la Convalida Mainline

Il modo più efficace per convalidare le innovazioni di ShashChess è interagire con il framework fishtest, non come un avversario, ma come l'arbitro finale. L'immensa potenza di elaborazione del framework [5] e il suo rigoroso modello statistico [28] sono lo standard di riferimento per una ragione. La raccomandazione strategica è di isolare, creare una patch e sottoporla a test.

1. **Isolare**: Invece di testare l'intero motore ShashChess, identificare l'idea singola più promettente e autonoma. Ad esempio, un singolo termine di valutazione ispirato all'archetipo "Petrosian" di Shashin che premia il controllo dello spazio intorno al re.
2. **Creare una Patch**: Sviluppare una patch minima che introduca solo questo singolo cambiamento all'ultimo ramo master di Stockfish.

3. **Sottoporre a Test**: Inviare questa patch a fishtest per un test LTC (Long Time Control).[16]

Questa strategia è vantaggiosa per tutti. Il progetto Stockfish è open-source e accoglie contributi che dimostrano di aumentare l'Elo.[9] Se la patch supera

fishtest con un guadagno di Elo statisticamente significativo (ad es., +2 Elo con LOS > 95%), fornisce una prova inconfutabile che l'idea di base è valida. Questo risultato porterebbe un significativo riconoscimento allo sviluppatore e al valore delle teorie di Shashin, e l'innovazione verrebbe incorporata nel motore scacchistico più forte del mondo, a beneficio dell'intera comunità. Se la patch fallisce, fornisce dati preziosi, indicando che l'idea, in isolamento, non è vantaggiosa nelle condizioni di test standard, consentendo di affinarla o di cambiare focus. Questo percorso trasforma il progetto da un motore derivato di nicchia a una potenziale fonte di innovazione per l'intero campo degli scacchi computerizzati, rappresentando il modo più efficace e credibile per convalidare l'eccellente lavoro e l'esplorazione teorica che sono stati investiti in ShashChess.

Bibliografia

1. amchess/ShashChess: A try to implement Alexander Shashin's theory on a Stockfish's derived chess engine - GitHub, accesso eseguito il giorno settembre 22, 2025, https://github.com/amchess/ShashChess
2. ShashChess - Chessprogramming wiki, accesso eseguito il giorno settembre 22, 2025, https://www.chessprogramming.org/index.php?title=ShashChess&mobileaction=toggle_view_desktop
3. ShashChess - Chessprogramming wiki, accesso eseguito il giorno settembre 22, 2025, https://www.chessprogramming.org/ShashChess
4. Releases · amchess/ShashChess - GitHub, accesso eseguito il giorno settembre 22, 2025, https://github.com/amchess/ShashChess/releases
5. Stockfish (chess) - Wikipedia, accesso eseguito il giorno settembre 22, 2025, https://en.wikipedia.org/wiki/Stockfish_(chess)
6. Introducing NNUE Evaluation - Stockfish - Strong open-source chess engine, accesso eseguito il giorno settembre 22, 2025, https://stockfishchess.org/blog/2020/introducing-nnue-evaluation/
7. ShashChess - Page 2 - TalkChess.com, accesso eseguito il giorno settembre 22, 2025, https://talkchess.com/viewtopic.php?t=78738&start=10
8. sohamkorade/chess_engine: A basic chess engine made in pure C++. - GitHub, accesso eseguito il giorno settembre 22, 2025, https://github.com/sohamkorade/chess_engine
9. official-stockfish/Stockfish: A free and strong UCI chess engine - GitHub, accesso eseguito il giorno settembre 22, 2025, https://github.com/official-stockfish/Stockfish
10. ShashChess and Brainlearn - Page 2 - TalkChess.com, accesso eseguito il giorno settembre 22, 2025, https://talkchess.com/viewtopic.php?t=83198&start=10
11. What proportion of games played among the top chess engines end in a draw?,

accesso eseguito il giorno settembre 22, 2025, https://chess.stackexchange.com/questions/46697/what-proportion-of-games-played-among-the-top-chess-engines-end-in-a-draw
12. chess.stackexchange.com, accesso eseguito il giorno settembre 22, 2025, https://chess.stackexchange.com/questions/46697/what-proportion-of-games-played-among-the-top-chess-engines-end-in-a-draw#:~:text=That%20said%3A%20since%20you've,extraordinarily%20high%2C%20%3E99%25.
13. Drawing Rates in Chess Pt.1 - Time Matters!, accesso eseguito il giorno settembre 22, 2025, https://www.chess.com/blog/ThomasJEvans/drawing-rates-in-chess-pt1---time-matters
14. Has the number of draws in chess increased? - ChessBase, accesso eseguito il giorno settembre 22, 2025, https://en.chessbase.com/post/has-the-number-of-draws-in-chess-increased
15. Draw rates in classical chess between elite players?, accesso eseguito il giorno settembre 22, 2025, https://chess.stackexchange.com/questions/24712/draw-rates-in-classical-chess-between-elite-players
16. Creating my first test - Stockfish Docs, accesso eseguito il giorno settembre 22, 2025, https://official-stockfish.github.io/docs/fishtest-wiki/Creating-my-first-test.html
17. What is the strongest chess engine in the world? — A Reflection - Page 2 - TalkChess.com, accesso eseguito il giorno settembre 22, 2025, https://talkchess.com/viewtopic.php?t=84490&start=10
18. Why does Fishtesting test Stockfish against itself, not other engines?, accesso eseguito il giorno settembre 22, 2025, https://chess.stackexchange.com/questions/24685/why-does-fishtesting-test-stockfish-against-itself-not-other-engines
19. Elo Calculator, accesso eseguito il giorno settembre 22, 2025, https://www.omnicalculator.com/sports/elo
20. Elo rating system - Wikipedia, accesso eseguito il giorno settembre 22, 2025, https://en.wikipedia.org/wiki/Elo_rating_system
21. Match Statistics - Chessprogramming wiki, accesso eseguito il giorno settembre 22, 2025, https://www.chessprogramming.org/Match_Statistics
22. Calculating the LOS (likelihood of superiority) from results - TalkChess.com, accesso eseguito il giorno settembre 22, 2025, https://talkchess.com/viewtopic.php?t=51003
23. Calculate ELO difference from record or winning percentage - 3DKingdoms, accesso eseguito il giorno settembre 22, 2025, https://3dkingdoms.com/chess/elo.htm
24. Elo Difference Calculator, accesso eseguito il giorno settembre 22, 2025, https://elocalculator.netlify.app/
25. Chess engine - Wikipedia, accesso eseguito il giorno settembre 22, 2025, https://en.wikipedia.org/wiki/Chess_engine
26. ShashChess Parameters - Next Chess Move, accesso eseguito il giorno

settembre 22, 2025, https://forums.nextchessmove.com/t/shashchess-parameters/1188
27. Chess engine, pt. 3: Elo, and rigorous SPRT testing, accesso eseguito il giorno settembre 22, 2025, https://www.dogeystamp.com/chess3/
28. Statistical Methods and Algorithms in Fishtest | Stockfish Docs - GitHub Pages, accesso eseguito il giorno settembre 22, 2025, https://official-stockfish.github.io/docs/fishtest-wiki/Fishtest-Mathematics.html