Analisi Approfondita del Motore Scacchistico ShashChess: Architettura, Prestazioni e Metodologia di Validazione

Sommario Esecutivo

Il presente rapporto fornisce un'analisi tecnica esaustiva del motore scacchistico ShashChess, un derivato di Stockfish. La valutazione si concentra su tre aree principali: l'architettura concettuale basata sulla teoria di Alexander Shashin, la validità statistica delle affermazioni sulle prestazioni e un'analisi comparativa della sua metodologia di test rispetto al framework fishtest della comunità di Stockfish.

L'architettura di ShashChess rappresenta un'innovazione paradigmatica significativa. L'introduzione di archetipi posizionali dinamici ("Tal", "Capablanca", "Petrosian") per adattare l'algoritmo di ricerca al carattere della posizione è un allontanamento teoricamente valido dal design monolitico dei motori convenzionali.[1] Questa specializzazione, unita a una funzione di valutazione ibrida che combina elementi di Leela Chess Zero e Stockfish, e a un'integrazione sperimentale del Monte Carlo Tree Search (MCTS), conferisce al motore un potenziale notevole per eccellere in tipologie di posizioni specifiche. Tuttavia, il successo di tale architettura dipende in modo critico dall'efficienza e dall'accuratezza del meccanismo di classificazione posizionale, che costituisce il principale punto di complessità e potenziale vulnerabilità.

L'analisi statistica dei risultati forniti rivela un quadro sfumato. L'aumento del 7,7% nel numero di posizioni complesse risolte rispetto a Stockfish è un indicatore forte e convincente di una superiore capacità di calcolo in contesti tattici e concreti. Al contrario, il risultato del match di 300 partite, che mostra un guadagno stimato di +10 Elo con un punteggio del 51,5%, non raggiunge la soglia di significatività statistica del 95% comunemente accettata nello sviluppo di motori scacchistici. La probabilità di superiorità (LOS) calcolata si attesta a circa il 79%, indicando che il risultato è promettente ma non conclusivo.

La metodologia di test adottata da ShashChess, basata su tempi di riflessione lunghi e posizioni "pepite" accuratamente selezionate, è un approccio valido per mitigare il problema

della "morte per patta" che affligge i test tra motori di alto livello.[2] Tuttavia, questa metodologia introduce un potenziale rischio di bias di selezione, ottimizzando le prestazioni del motore in un dominio specifico a scapito della forza di gioco generale. Si contrappone al framework

fishtest, progettato per la validazione rigorosa di miglioramenti incrementali e generalizzabili attraverso un volume massiccio di partite a tempi brevi.[4]

Le raccomandazioni strategiche si concentrano su tre assi principali. In primo luogo, per una validazione definitiva, si suggerisce un approccio di test ibrido che includa suite di posizioni diversificate (sia tattiche che strategiche) e un match basato su un libro di aperture standard e bilanciato. In secondo luogo, lo sviluppo futuro dovrebbe dare priorità all'ottimizzazione del modulo di classificazione posizionale. Infine, la raccomandazione più critica è quella di garantire la piena accessibilità e verificabilità del codice sorgente. L'attuale inaccessibilità di file chiave come search.cpp e evaluate.cpp rappresenta un ostacolo significativo alla credibilità e all'accettazione del progetto da parte della comunità open-source.

Sezione 1: Una Revisione Architettonica del Framework Ispirato a Shashin

Questa sezione analizza le innovazioni concettuali fondamentali di ShashChess, valutando la solidità teorica e il potenziale del suo design unico e sensibile al contesto.

1.1. Il Paradigma degli Archetipi Posizionali: Tal, Capablanca e Petrosian

Il design di ShashChess si discosta radicalmente dall'approccio monolitico tipico dei moderni motori scacchistici, come Stockfish. Invece di utilizzare un singolo algoritmo di ricerca e valutazione ottimizzato per la performance media, ShashChess implementa un sistema dinamico che adatta il proprio comportamento in base alla natura della posizione sul tavoliere. Questo approccio si fonda sulla teoria di Alexander Shashin, che classifica le posizioni in tre archetipi principali:

- **Tal**: Posizioni caratterizzate da un attacco acuto e da complesse possibilità tattiche.
- **Capablanca**: Posizioni strategiche, tranquille e manovrate, dove la comprensione posizionale prevale sul calcolo puro.

- **Petrosian**: Posizioni difensive, che possono essere viste come l'inverso speculare delle posizioni di tipo "Tal".[1]

A questi si aggiungono posizioni miste che combinano elementi dei tre archetipi. Questa segmentazione concettuale permette, in teoria, di applicare euristiche di ricerca, estensioni e potature specializzate. Ad esempio, in una posizione "Tal", il motore potrebbe dare maggiore priorità alle estensioni di ricerca per i controlli e le minacce dirette, mentre in una posizione "Capablanca" potrebbe favorire una ricerca più ampia e meno profonda per esplorare piani strategici a lungo termine.

Il vero fulcro tecnico e la sfida più grande di questa architettura non risiedono tanto nella logica specifica di ogni archetipo, quanto nel meccanismo di *classificazione* che determina la natura della posizione. L'efficacia dell'intero sistema dipende dalla capacità del motore di identificare correttamente e rapidamente a quale archetipo appartiene una data posizione. Questo processo di classificazione deve essere eseguito potenzialmente a ogni nodo dell'albero di ricerca, o quantomeno a intervalli strategici, introducendo un inevitabile sovraccarico computazionale. Si crea così un "bilancio di complessità": il guadagno di Elo ottenuto grazie all'applicazione di logiche specializzate deve superare la perdita di Elo causata dal tempo di calcolo speso per la classificazione. Una classificazione errata, come trattare una posizione difensiva ("Petrosian") come una d'attacco ("Tal"), potrebbe portare a decisioni di gioco catastrofiche. Di conseguenza, l'implementazione del classificatore è l'elemento più critico e potenzialmente fragile dell'intero motore, rappresentando sia il maggior rischio che la più grande opportunità di ottimizzazione.

1.2. Valutazione Ibrida: Una Sintesi di Approcci Neurali e Classici

ShashChess adotta una funzione di valutazione collaborativa che integra le capacità di Leela Chess Zero (LCZero) e Stockfish.[1] Questa scelta è particolarmente interessante nel contesto dell'evoluzione dei motori scacchistici. Stockfish stesso ha attraversato una transizione significativa, passando da una funzione di valutazione classica, realizzata a mano (Hand-Crafted Evaluation, HCE), a una basata interamente su una rete neurale efficientemente aggiornabile (NNUE) a partire dal 2020.[5] LCZero, d'altra parte, utilizza un'architettura di rete neurale diversa e più grande, simile a quella di AlphaZero, che è spesso percepita come dotata di una comprensione più "intuitiva" o strategica del gioco.

La fusione di questi due approcci offre sinergie potenziali. La rete di LCZero potrebbe eccellere nella valutazione di sfumature strategiche a lungo termine, mentre la NNUE di Stockfish è rinomata per la sua velocità eccezionale e la sua precisione tattica. L'implementazione di questa valutazione ibrida potrebbe essere la chiave per sbloccare il pieno potenziale degli archetipi posizionali. Un'implementazione sofisticata non si limiterebbe

a una media ponderata dei due punteggi. Piuttosto, potrebbe creare un ciclo di feedback virtuoso: i componenti della valutazione potrebbero informare il classificatore posizionale (ad esempio, una valutazione ad alta entropia da parte di LCZero potrebbe suggerire una posizione di tipo "Tal"), e l'archetipo di ricerca attivato potrebbe a sua volta dare un peso diverso ai componenti della valutazione (ad esempio, una ricerca "Capablanca" potrebbe fare maggiore affidamento sul giudizio strategico di LCZero, mentre una ricerca "Tal" si affiderebbe alla velocità tattica della NNUE di Stockfish). Questa stretta integrazione tra i due pilastri innovativi di ShashChess—la ricerca adattiva e la valutazione ibrida—è fondamentale. Un'implementazione ingenua rischierebbe di generare un segnale di valutazione confuso, mentre un design avanzato potrebbe creare una risonanza architettonica in cui ogni componente amplifica l'efficacia dell'altro, portando a un motore con una comprensione del gioco più profonda e contestuale.

1.3. Integrazione Sperimentale del Monte Carlo Tree Search (MCTS)

Il README di ShashChess menziona una sezione sperimentale che permette di attivare un algoritmo di Monte Carlo Tree Search (MCTS), applicandolo specificamente a posizioni classificate come "Petrosian high, high middle and middle".[1] La logica dichiarata è quella di sfruttare la forza di LCZero, che utilizza MCTS, in queste particolari tipologie di posizioni complesse e difensive.

Questa è una scelta progettuale teoricamente solida e promettente. L'MCTS, come dimostrato da motori come AlphaZero e LCZero, eccelle in posizioni strategiche complesse dove la ricerca alfa-beta tradizionale può essere afflitta dall'effetto orizzonte, faticando a valutare correttamente piani a lungo termine. L'applicazione selettiva dell'MCTS alle posizioni difensive e strategiche ("Petrosian") è un tentativo intelligente di ottenere il "meglio di entrambi i mondi": la potenza tattica bruta e la velocità della ricerca alfa-beta di Stockfish nelle posizioni acute, e la profondità strategica e la robustezza dell'MCTS nelle fasi di gioco più tranquille e sfumate. Il successo di metodi simili, come l'UCT (Upper Confidence bounds applied to Trees) nel Go computerizzato, dimostra la potenza di questo approccio in giochi con spazi di ricerca vasti e complessi.[6] Questa funzionalità sperimentale rappresenta una direzione di ricerca affascinante che potrebbe ulteriormente differenziare ShashChess dai motori convenzionali, consentendogli di navigare con maggiore sicurezza in posizioni che richiedono una profonda comprensione strategica piuttosto che una mera forza di calcolo.

Sezione 2: Validazione Statistica delle Affermazioni

sulle Prestazioni

Questa sezione esamina con rigore statistico i dati forniti relativi al match e al benchmark di problem-solving per determinare il livello di confidenza del miglioramento prestazionale dichiarato.

2.1. Analisi del Risultato del Match di 300 Partite

I dati forniti indicano che ShashChess ha ottenuto un punteggio di 154,5 su 300 (51,5%) in un match contro una versione specifica di Stockfish, con un risultato di +67 vittorie, 175 patte e 58 sconfitte. Questo punteggio si traduce in una stima di performance superiore di +10 Elo [User Query].

Un punteggio del 51,5% è indubbiamente positivo. Tuttavia, nel campo dei test sui motori scacchistici, la domanda cruciale è se questo risultato sia statisticamente significativo o se possa essere attribuito alla varianza casuale ("rumore statistico"). Per quantificare ciò, è possibile applicare strumenti statistici standard, come il test della Probabilità di Superiorità (LOS, Likelihood of Superiority), comunemente utilizzato in questo dominio.[7]

L'affermazione di una superiorità di +10 Elo, basata su questi dati, risulta statisticamente debole e non raggiunge le soglie di confidenza standard. Il calcolo della LOS fornisce una misura quantitativa di questa incertezza. La formula approssimata per lo Z-score, da cui si deriva la LOS, è Z=W+LW−L, dove W è il numero di vittorie e L è il numero di sconfitte.

1. I dati del match sono: Vittorie (W) = 67, Sconfitte (L) = 58.
2. La differenza tra vittorie e sconfitte è W−L=9.
3. La somma delle partite decisive è W+L=125.
4. Applicando la formula, si ottiene uno Z-score di Z=1259≈11.189≈0.805.
5. Uno Z-score di 0.805 corrisponde a una probabilità cumulativa di circa il 79% in una distribuzione normale standard. Questa è la LOS.

Una LOS del 79% significa che c'è una probabilità del 79% che ShashChess sia effettivamente più forte della versione di Stockfish testata. Sebbene questo risultato sia incoraggiante, lo standard scientifico per accettare un'ipotesi, specialmente in contesti rigorosi come il framework fishtest, richiede tipicamente un livello di confidenza del 95% o superiore. Pertanto, sulla base di questo campione di 300 partite, l'affermazione di superiorità è *probabile* ma non *statisticamente provata*. Il valore di +10 Elo è una stima puntuale che si trova all'interno di un intervallo di confidenza relativamente ampio, che potrebbe

realisticamente includere anche un valore di 0.

2.2. Valutazione del Benchmark di Problem-Solving

I dati del benchmark indicano che ShashChess ha risolto 140 posizioni su una suite di 256, mentre Stockfish ne ha risolte 130 [User Query]. Questo rappresenta un aumento del 7,7% nel numero di problemi risolti, un risultato notevole.

L'uso di suite di test-posizioni è una metodologia di valutazione consolidata, utile per verificare specifiche capacità del motore, come la ricerca di tattiche complesse o la comprensione di finali difficili. È importante notare che un'alta performance su queste suite non si correla sempre in modo diretto con un aumento della forza di gioco complessiva in un match, ma fornisce indicazioni preziose su aree specifiche di miglioramento o regressione.[8]

In questo caso, il risultato è un forte indicatore che le modifiche architettoniche di ShashChess hanno un impatto tangibile e positivo in posizioni concrete e complesse. Suggerisce che il motore è più abile nel navigare attraverso complicazioni tattiche, il che è coerente con l'obiettivo di utilizzare "posizioni pepite" (sharp) per i test. Questo risultato è, per certi versi, più convincente del marginale punteggio del match, poiché indica un'area specifica e misurabile di superiorità, probabilmente legata alla capacità del motore di calcolare più a fondo o in modo più intelligente in situazioni critiche.

2.3. La Sfumatura della Potenza Statistica a Tempi di Riflessione Lunghi

Viene asserito che un campione di 300 partite a tempi di riflessione lunghi (1 minuto per mossa in media) sia sufficiente per eliminare il rumore statistico, una conclusione che si dice essere stata confermata da un'intelligenza artificiale [User Query]. Questa affermazione richiede un'analisi più approfondita.

È corretto affermare che tempi di riflessione più lunghi riducono una specifica fonte di rumore: la variabilità delle prestazioni del motore dovuta a fattori come la casualità del sistema operativo o la gestione del tempo in situazioni di pressione.[8] Permettono al motore di raggiungere profondità di ricerca maggiori e valutazioni più stabili. Tuttavia, la potenza statistica di un test—ovvero la sua capacità di rilevare un effetto reale—è una funzione sia della dimensione del campione che dell'entità dell'effetto che si sta misurando.

La metodologia adottata è ben congegnata per rilevare differenze di Elo *grandi*, ma è

intrinsecamente inadatta a provare con alta confidenza differenze *piccole*. Per dimostrare in modo conclusivo una piccola differenza di Elo (come +10), è necessario un numero molto elevato di partite, indipendentemente dal tempo di riflessione. Il framework fishtest, ad esempio, esegue decine di migliaia di partite proprio per questo motivo: rilevare segnali molto deboli (piccoli guadagni di Elo) in mezzo al rumore statistico.[9] Il campione di 300 partite di ShashChess avrebbe la potenza statistica per rilevare con sicurezza un grande guadagno di Elo (ad esempio, +50), ma come dimostrato dal calcolo della LOS, è insufficiente per provare un guadagno di +10. La conferma generica da parte di un'IA probabilmente non ha tenuto conto del contesto specifico dei test sui motori scacchistici, dove il "segnale" da misurare è estremamente debole. L'errore sta nell'identificare la fonte primaria di incertezza non nella varianza legata al tempo di riflessione, ma nell'errore di campionamento, che può essere ridotto solo aumentando in modo significativo il numero di partite.

Sezione 3: Un'Analisi Comparativa delle Filosofie di Test dei Motori

Questa sezione valuta criticamente la strategia di test di ShashChess mettendola a confronto con il framework fishtest. L'obiettivo non è stabilire quale sia "giusta" o "sbagliata", ma inquadrarle come due strumenti distinti, progettati per scopi diversi.

3.1. Il Framework fishtest: Ottimizzazione Incrementale ad Alto Volume

Fishtest è un sistema di test distribuito che esegue un numero massiccio di partite—miliardi nel corso della sua esistenza—per testare le modifiche al codice di Stockfish.[5] Le sue caratteristiche principali sono:

- **Tempi di riflessione ultra-rapidi**: I test standard (STC e LTC) utilizzano tempi di riflessione molto brevi (es. 10 secondi + 0.1 secondi di incremento) per massimizzare il numero di partite giocate in un dato periodo.[4]
- **Test Sequenziale SPRT**: Utilizza il Sequential Probability Ratio Test (SPRT) per determinare la significatività statistica di un risultato nel modo più efficiente possibile, interrompendo il test non appena viene raggiunta una soglia di confidenza predefinita (solitamente il 95%).[4]
- **Piccole modifiche atomiche**: Il framework è progettato per testare piccole modifiche

isolate al codice, permettendo di attribuire con precisione qualsiasi variazione di performance a una specifica modifica.[4]

La filosofia alla base di fishtest è quella di rilevare con certezza quasi assoluta miglioramenti minuscoli e generalizzabili nell'algoritmo di base del motore. Risponde alla domanda: "Questa piccola modifica al codice rende il motore universalmente più forte, anche solo di +0.5 Elo?". Tratta gli scacchi come un enorme problema statistico da risolvere attraverso una mole schiacciante di dati.

3.2. La Metodologia ShashChess: Analisi Profonda e Curata

La strategia di test di ShashChess, come descritta, si basa su principi opposti:

- **Tempi di riflessione lunghi**: Un tempo medio di 1 minuto per mossa simula condizioni di gioco più simili a quelle di un torneo umano [User Query].
- **Basso volume di partite**: Un match di 300 partite e una suite di 256 posizioni [User Query].
- **Posizioni di partenza selezionate**: Le partite iniziano da posizioni "pepite" (sharp), scelte per essere tatticamente e strategicamente complesse, evitando le linee di apertura teoriche che spesso portano a patte [User Query].

Questa filosofia affronta un problema ben documentato negli scacchi moderni di alto livello: la "morte per patta" (draw death). Quando i motori più forti giocano dalla posizione di partenza standard, la stragrande maggioranza delle partite termina in parità, rendendo difficile discernere differenze di forza.[2] La metodologia di ShashChess è progettata per rispondere a una domanda diversa: "In una posizione complessa e decisiva, simile a quelle che si incontrano in un torneo, quale motore dimostra una forza pratica superiore?".

3.3. Sintesi: Due Metodologie per Risolvere Due Problemi Diversi

Il punto cruciale dell'analisi è che queste due metodologie non sono in competizione diretta, ma sono complementari. Fishtest è uno strumento per l'**ottimizzazione del codice**, mentre il metodo ShashChess è uno strumento per la **valutazione delle prestazioni in domini specifici**.

Tuttavia, la metodologia di ShashChess comporta un rischio significativo che non viene affrontato: il bias di selezione e l'overfitting. Selezionando accuratamente posizioni "pepite", il test potrebbe essere involontariamente orientato a favorire i punti di forza specifici

dell'architettura di ShashChess (ad esempio, il suo modo di attacco "Tal"). Il motore potrebbe mostrare un guadagno di +10 Elo in queste posizioni, ma potenzialmente subire una regressione in posizioni tranquille e strategiche ("Capablanca") che vengono escluse dalla suite di test. Fishtest, utilizzando libri di apertura bilanciati e giocando un'enorme varietà di posizioni, evita questo tipo di bias e misura una forza di gioco più generale e universale. In sostanza, esiste il rischio che ShashChess venga ottimizzato per superare il proprio benchmark, senza che ciò si traduca necessariamente in una superiorità globale.

La tabella seguente riassume le differenze filosofiche e pratiche tra i due approcci.

Caratteristica	Framework fishtest	Metodologia ShashChess
Obiettivo Primario	Validazione statistica di piccole modifiche al codice.	Valutazione della forza pratica in posizioni complesse.
Tempo di Riflessione	Molto breve (es. 10s + 0.1s).	Lungo (es. 1 min/mossa).
Volume di Partite	Molto alto (decine di migliaia per test).	Basso (centinaia).
Posizioni di Partenza	Libri di apertura bilanciati e standard.	Posizioni "pepite" selezionate (tattiche/strategiche).
Modello Statistico	SPRT (Sequential Probability Ratio Test).	Statistiche di base del match (punteggio, %).
Punti di Forza	Altissima confidenza statistica, evita l'overfitting, rileva guadagni minimi.	Simula condizioni di torneo, evita la "morte per patta", testa la forza in posizioni critiche.
Punti di Debolezza	Non rappresentativo delle condizioni di torneo, criticato per i tempi ultra-rapidi.	Bassa potenza statistica per piccoli guadagni di Elo, rischio di bias di selezione e overfitting.

Questa tabella illustra come i due approcci non siano mutuamente esclusivi. Fishtest è ideale

per lo sviluppo incrementale del "core" del motore, mentre la metodologia di ShashChess è eccellente per testare e dimostrare la sua efficacia in scenari di gioco specifici e impegnativi.

Sezione 4: Raccomandazioni Strategiche per l'Avanzamento del Progetto

Questa sezione finale fornisce raccomandazioni concrete e attuabili per migliorare il motore, rafforzarne la validazione e aumentarne l'impatto.

4.1. Percorso verso una Validazione Conclusiva: Un Gauntlet di Test Ibrido

Per dimostrare in modo definitivo la superiorità del motore e ottenere un più ampio riconoscimento, è necessario un approccio di test a più livelli che combini i punti di forza di entrambe le filosofie.

1. **Diversificare la Suite di Test**: Continuare i test a tempi lunghi, ma espandere la suite di 256 posizioni "pepite" per includere un set altrettanto numeroso di posizioni tranquille e strategiche. Questo permetterà di testare specificamente la logica "Capablanca" e "Petrosian" e di garantire che i miglioramenti in un'area non siano avvenuti a scapito di regressioni in altre.
2. **Eseguire un Gauntlet Standard**: Condurre un match separato (ad esempio, 400-600 partite) partendo da un libro di aperture standard e bilanciato (es. a 2 o 3 mosse). Questo affronterebbe direttamente le critiche sul potenziale overfitting e fornirebbe un risultato direttamente confrontabile con quelli delle principali liste di rating dei motori.
3. **Confrontarsi con fishtest**: Come obiettivo a lungo termine, considerare di sottoporre a fishtest una versione semplificata di ShashChess che implementi solo una delle idee centrali. Sebbene il framework sia stato criticato, superare un test LTC (Long Time Control) su fishtest rimane il gold standard indiscusso per provare un guadagno di Elo all'interno della comunità di Stockfish.[4]

4.2. Affinamento Algoritmico: Ottimizzazione del Motore Centrale

L'attenzione dello sviluppo dovrebbe concentrarsi sull'efficienza e l'accuratezza del modulo di classificazione posizionale, che è il cuore dell'innovazione di ShashChess.

1. **Profiling delle Prestazioni**: Eseguire un'analisi di profiling del codice per determinare la percentuale di tempo CPU spesa per la classificazione rispetto alla ricerca e alla valutazione. Se questo sovraccarico è significativo (ad esempio, superiore al 5-10%), la sua ottimizzazione dovrebbe diventare la massima priorità.
2. **Sintonizzazione dei Parametri**: Le soglie e le condizioni che determinano il passaggio tra le modalità Tal, Capablanca e Petrosian sono probabilmente i parametri più sensibili dell'intero motore. È fondamentale implementare un processo di sintonizzazione rigoroso per questi parametri specifici. Un approccio come la SPSA (Simultaneous Perturbation Stochastic Approximation), menzionato nella documentazione di fishtest per la sintonizzazione dei parametri, potrebbe essere estremamente efficace.[4]

4.3. Coinvolgimento della Comunità e Verificabilità

La raccomandazione più importante per il futuro del progetto è quella di rendere il codice sorgente completamente accessibile e verificabile.

L'attuale inaccessibilità di file sorgente cruciali come search.cpp e evaluate.cpp è il più grande ostacolo alla credibilità del progetto. La natura open-source e la licenza GPL sono pilastri del successo e dello sviluppo di Stockfish.[1][2] Senza un codice aperto, la verifica esterna delle affermazioni architettoniche è impossibile, e i risultati, per quanto promettenti, saranno sempre accolti con scetticismo dalla comunità di sviluppatori. Aprire completamente il codice permetterebbe una revisione paritaria (peer review), l'identificazione di bug, suggerimenti per il miglioramento e potenziali collaborazioni, sfruttando la stessa potenza collettiva che ha reso Stockfish il motore di riferimento a livello mondiale.

Bibliografia

1. amchess/ShashChess: A try to implement Alexander ... - GitHub, accesso eseguito il giorno settembre 21, 2025, https://github.com/amchess/ShashChess
2. First-move advantage in chess - Wikipedia, accesso eseguito il giorno settembre 21, 2025, https://en.wikipedia.org/wiki/First-move_advantage_in_chess
3. What proportion of games played among the top chess engines end in a draw?, accesso eseguito il giorno settembre 21, 2025, https://chess.stackexchange.com/questions/46697/what-proportion-of-games-played-among-the-top-chess-engines-end-in-a-draw
4. Creating my first test - Stockfish Docs, accesso eseguito il giorno settembre 21, 2025, https://official-stockfish.github.io/docs/fishtest-wiki/Creating-my-first-test.html

5. Stockfish (chess) - Wikipedia, accesso eseguito il giorno settembre 21, 2025, https://en.wikipedia.org/wiki/Stockfish_(chess)
6. Statistical approach to chess? - Stack Overflow, accesso eseguito il giorno settembre 21, 2025, https://stackoverflow.com/questions/2717476/statistical-approach-to-chess
7. Match Statistics - Chessprogramming wiki, accesso eseguito il giorno settembre 21, 2025, https://www.chessprogramming.org/Match_Statistics
8. Engine Testing - Chessprogramming wiki, accesso eseguito il giorno settembre 21, 2025, https://www.chessprogramming.org/Engine_Testing
9. How does Stockfish ensure an incremental change made is beneficial?, accesso eseguito il giorno settembre 21, 2025, https://chess.stackexchange.com/questions/23620/how-does-stockfish-ensure-an-incremental-change-made-is-beneficial
10. official-stockfish/fishtest: The Stockfish testing framework - GitHub, accesso eseguito il giorno settembre 21, 2025, https://github.com/official-stockfish/fishtest
11. Chess engine, pt. 3: Elo, and rigorous SPRT testing, accesso eseguito il giorno settembre 21, 2025, https://www.dogeystamp.com/chess3/
12. official-stockfish/Stockfish: A free and strong UCI chess engine - GitHub, accesso eseguito il giorno settembre 21, 2025, https://github.com/official-stockfish/Stockfish