Novità di ShashChess 38



ShashChess 38 rappresenta un'evoluzione significativa rispetto a Stockfish, introducendo innovazioni strategiche e tecniche che lo distinguono come un motore derivato piuttosto che un clone. Le zone Shashin, il modello WDL e un approccio dinamico alla gestione delle posizioni difficili lo rendono unico nel panorama dei motori scacchistici.

Analisi Tecnica Dettagliata delle Innovazioni Introdotte

# Isolamento della struttura Shashin

## Originalità

Il framework Shashin introduce la gestione avanzata delle zone posizionali, ora isolata in un pacchetto dedicato per migliorare modularità, manutenibilità e scalabilità.

## Attuazione

### File separati

Il framework è organizzato in file come `shashin\_helper.h`, `shashin\_manager.cpp` e `shashin\_types.h`, che gestiscono la classificazione delle zone, gli aggiornamenti dello stato e l'integrazione con il motore.

### Interfacce definite

API chiare come "getShashinRange", "initShashinValues" e "updateShashinValues" consentono una facile interazione con il framework.

### Indipendenza

Sebbene integrato nella pipeline di ricerca, il framework Shashin è autonomo, il che lo rende facilmente aggiornabile.

## Vantaggi

### Modularità

Consente miglioramenti al framework Shashin senza influenzare il resto del motore.

### Testabilità

Consente test separati per garantire una determinazione precisa della zona.

### Scalabilità

Facilita l'integrazione di nuove funzionalità o zone.

# Introduzione del modello WDL (Win-Draw-Loss).

## Innovazione

Il modello WDL supera i limiti della semplice probabilità di vincita, aggiungendo granularità e precisione nella classificazione posizionale. In base ai valori (w, d, l), il modello distingue posizioni con caratteristiche diverse anche a parità di probabilità di vincita.

## Attuazione

### Calcolo del WDL

* w: Percentuale di vincite previste;
* d: percentuale di pareggi attesi;
* l: percentuale delle perdite attese;
* Probabilità di vittoria = w + d/2;

### - \*\*Mappatura delle zone\*\*:

- Zone come Alto Tal, Capablanca, Medio Petrosian, ecc., sono definite da intervalli simmetrici che rispettano l'equilibrio posizionale (ad esempio, Petrosian ↔ Tal).

- Capablanca funge da centro posizionale, rappresentato da (w, d, l) = (0, 100, 0) con Probabilità di vittoria = 50.

- \*\*Simmetria mantenuta\*\*:

- Vittoria Petrosiana elevata = Perdita tal. elevata.

- Perdita Capablanca-Petrosian = Vittoria Capablanca-Tal.

- \*\*Integrazione dinamica\*\*:

- `WDLModel::get\_wdl(value, rootPos)` determina le probabilità di vincite, pareggi e perdite, classificate tramite `getShashinRange`.

## Vantaggi

### Precisione

Distingue accuratamente tra posizioni con probabilità di vincita simili ma caratteristiche diverse.

### Adattabilità

Consente al motore di rispondere efficacemente sia alle esigenze tattiche che strategiche.

### Resilienza

Gestisce in modo ottimale posizioni caotiche o critiche.

# Inizializzazione e aggiornamento dello stato di Shashin

## Principio guida

L'approccio Shashin si ispira al principio di massima resilienza in termodinamica, suggerendo di iniziare con un approccio aggressivo (es. High Tal) e adattarsi progressivamente alla posizione.

## Attuazione

### Inizializzazione dello stato

* Durante "initShashinValues", il motore inizia con uno stato aggressivo (ad esempio, Alto Tal o Caos) per esplorare opportunità tattiche.
* La struttura "RootShashinState" tiene traccia dei parametri chiave come profondità, zona corrente, conteggio delle mosse legali, materiale e sicurezza del re.

### Aggiornamenti dinamici

Durante `updateShashinValues`, il framework ricalcola la zona Shashin ad ogni profondità di ricerca, adattando la strategia, in base alla seguente tabella:

### 

### Gestione del caos

In posizioni incerte (ad esempio, Caos: Capablanca-Petrosian-Tal), il motore esplora un'ampia gamma di mosse, riducendo la potatura aggressiva.

### Vantaggi

#### Progressività

Transizione graduale dal gioco tattico a quello strategico, migliorando la comprensione dinamica.

#### Resilienza

Gestione ottimale di posizioni instabili o complesse.

#### Adattamento continuo

Evoluzione strategica durante il gioco.

Valore Aggiunto Rispetto allo Stoccafisso

ShashChess non è un clone di Stockfish ma un motore derivato che introduce innovazioni significative:

# Flessibilità dinamica

Il framework Shashin consente al motore di adattarsi dinamicamente tra approcci tattici (ad esempio Tal) e strategici (ad esempio Petrosian).

# Precisione nel calcolo

Il modello WDL migliora la capacità del motore di distinguere tra posizioni con caratteristiche simili, ottimizzando la gestione delle situazioni critiche.

# Gestione del caos

L'approccio di Shashin garantisce una gestione superiore di posizioni complesse rispetto alla potatura standard di Stockfish.

# Prestazioni nei controlli a lungo termine\*\*:

Grazie alla resilienza e all'adattabilità, ShashChess eccelle nelle partite di lunga durata pur mantenendo un'elevata competitività nelle partite veloci.

# Risultati del benchmark

ShashChess è stato sottoposto a test rigorosi per convalidare i suoi miglioramenti:

## Partita LTC su 10 posizioni "Pepite".

ShashChess ha vinto 4-3 contro Stockfish, con 17 pareggi, dimostrando la sua gestione superiore di posizioni complesse e tatticamente acute: <https://github.com/amchess/ShashChess/wiki/Match>

## Suite di 128 posizioni difficili

ShashChess ha risolto 22 posizioni in più rispetto a Stockfish, sottolineando la sua maggiore capacità di gestire efficacemente scenari difficili: <https://github.com/amchess/ShashChess/wiki/Battery>

Conclusione

L'integrazione del framework Shashin, del modello WDL e della gestione dinamica delle zone rappresenta un'idea innovativa ed eccezionale. ShashChess è un motore flessibile e adattabile, in grado di gestire qualsiasi scenario posizionale con intelligenza strategica. Queste innovazioni lo distinguono come un contributo significativo alla comunità degli scacchi e una derivazione davvero originale e innovativa di Stockfish.