Repertoire Builder Guide

AIChessGM's Repertoire Builder is a transposition-aware repertoire workstation for serious chess players. Build opening trees from databases and PGNs, repair weak branches with engine workflows, split one master repertoire into opening-specific files, and turn opponent prep into focused training.

Tournament prep | Split-tree workflows | Fix Queue + Backsolve | Opponent intersections

OVERVIEW

More Than an Opening Tree

The Repertoire Builder is not just a place to store opening moves. It is where your database work, engine analysis, comments, split files, and training workflows come together. The same position is tracked as one node even when move orders transpose, which makes it useful for real tournament preparation rather than just browsing lines move by move.

Build from Real Data

Create `.otb` trees from your own games, model games, filtered databases, PGNs, and imported opening material.

Repair and Maintain

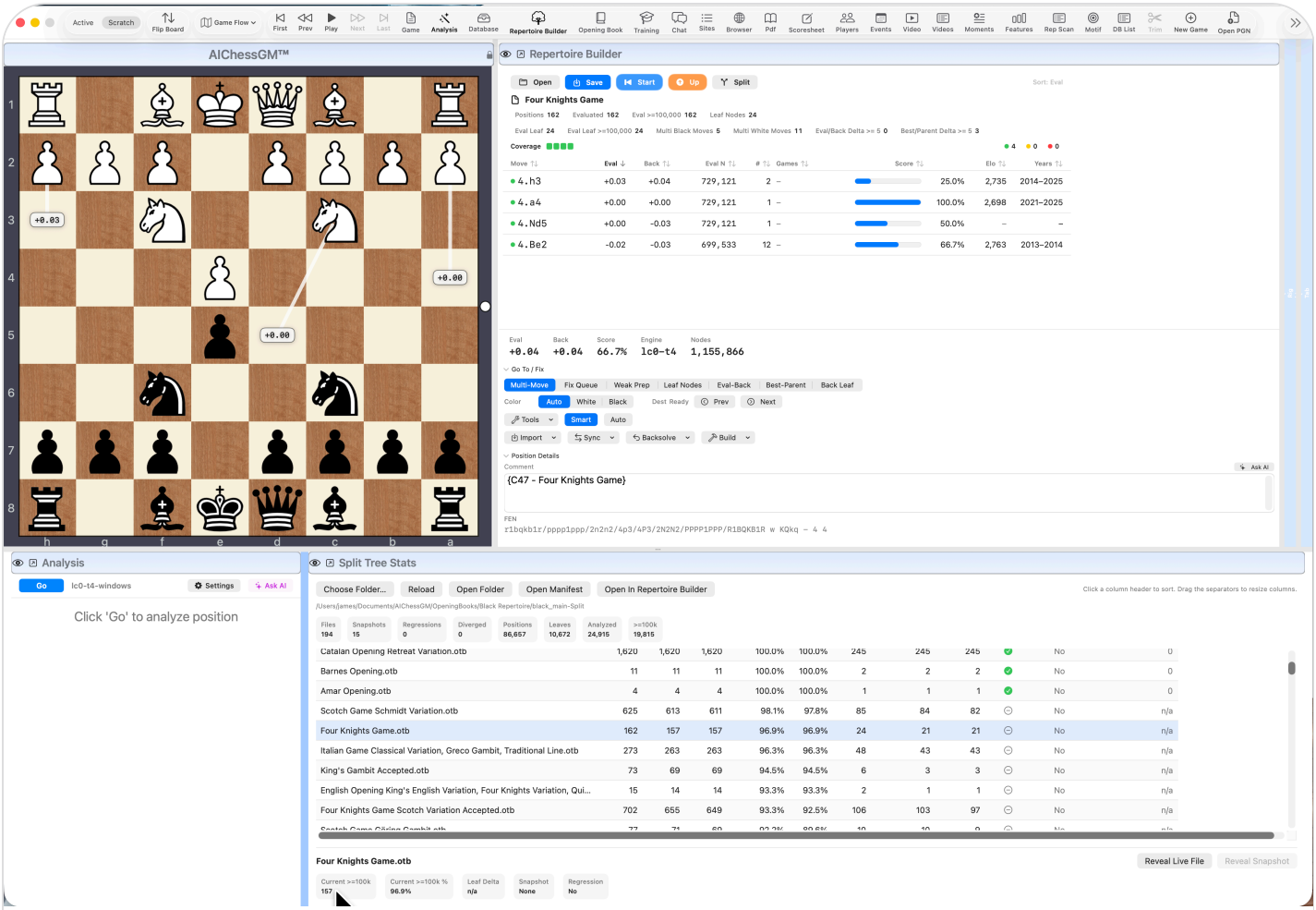
Use Eval/Back, Best/Parent, Fix Queue, Import Engine Lines, and whole-tree propagation to keep the repertoire healthy over time.

Split Into Working Files

Turn one broad master repertoire into opening-specific split trees with coverage tracking and archived analyzed snapshots.

Train What Matters

Push opening prep into Repertoire Training, Book vs Book, or narrow intersected round-prep files before a game.



The Repertoire Builder pane combines move statistics, engine evaluation, propagated backsolve values, quick maintenance controls, and navigation tools for finding weak or inconsistent positions.

TOURNAMENT PREPARATION

How This Helps Tournament Players

The strongest use of the Repertoire Builder is not “store my openings.” It is turning broad chess data into a practical pre-round workflow. AIChessGM can help you build a tree for a potential opponent, intersect it with your own repertoire, filter game collections by the current tree, and then drill only the lines that are actually likely to occur.

High-Value Tournament Workflows

1. Filter games for a likely opponent, then build a Repertoire Builder from those games.
2. Build or open your own White or Black repertoire.
3. Use `Intersect Two Trees...` to isolate the shared battleground between your prep and the opponent’s habits.
4. Filter the database by the current repertoire tree so you keep only games that stay inside your prep long enough to matter.
5. Train from the intersected file rather than the whole repertoire.

A lot of players over-prepare. Intersections and repertoire-aware filtering reduce wasted study by cutting away lines that never realistically overlap with the opponent’s choices.

Why This Is Better Than a Flat Game List

- You see branching decisions position by position.
- You can measure how deeply a prep line is covered.
- You can run engine cleanup directly on the positions that matter.
- You can move from scouting to training without rebuilding the prep from scratch.

Build an Opponent Tree

Use a filtered database to create a tree for one player, one side, or one date range. This gives you the opponent's actual branching map rather than a stack of disconnected games.

Train from the Intersection

Intersect the opponent tree with your own repertoire and train the resulting file before a round. That turns scouting into usable rehearsal.

Filter Databases by Repertoire

Use repertoire filtering to keep only games that remain inside the current tree for a chosen number of plies. This builds a much cleaner study set than ECO-only filtering.

Round-Specific Prep Files

Create a small focused file for tomorrow's game instead of dragging your entire opening system into every prep session.

SPLIT TREES

Split Repertoires as Full Working Systems

Split trees are one of the most powerful parts of the Repertoire Builder. Once your master repertoire gets broad, splitting by opening gives you smaller opening-specific workspaces, better integrity tracking, better opponent matching, and cleaner training workflows.

Split by Opening

Create one `.otb` per opening family inside a `-Split` folder, while preserving the ancestry required to keep those files usable as real trees.

Split Tree Stats Pane

Inspect the entire split set at once, sort by coverage and deep-analysis counts, and spot regressions without opening every file individually.

Analyzed Snapshots

When a split tree reaches the deep-analysis threshold, AIChessGM can preserve a snapshot in `Analyzed/` so completed opening work is not silently lost.

Split-Aware Resolution

AIChessGM can resolve split files from active games, scan records, side-specific settings, and opening names, then fall back intelligently when an exact file is missing.

Split Features Worth Knowing

- Find split files containing the current position.
- Jump to the split's named opening position.
- Open Scratch directly at the split anchor.
- Open the best matching Black split file for the active game.
- Merge split trees back into a new parent repertoire.
- Auto-import saved split changes into the parent repertoire.

How Tournament Players Use Split Trees

- Keep one broad master tree for long-term repertoire work.
- Maintain opening-family split trees for practical analysis sessions.
- Intersect only the relevant split with an opponent tree.
- Train from the split or intersected split instead of the whole repertoire.

MAINTENANCE

Keep the Tree Healthy Over Time

Serious repertoires need maintenance. AIChessGM gives you tools to find contradictory or weak positions, refresh engine work, propagate changes across the tree, and automate large repair passes without casually overwriting mature analysis.

Fix Queue

Cycle through the highest-value cleanup targets: missing evals, multi-move issues, Best/Parent problems, and Eval/Back mismatches.

Back Source Navigation

Jump to the immediate back source or the responsible leaf for an Eval/Back mismatch, then return directly to the original mismatch position.

Quick (Propagate Whole Tree)

Refresh propagated backsolve values across the whole tree after leaf or edge analysis changes, without needing another leaf-analysis pass.

Fix+Loop Overnight

Let the app work through routine repair targets overnight. Million-node Smart-protected cases are skipped for manual review instead of burning the session.

Smart update is the right default for serious repertoire work. The automatic fix loop is designed to harvest easy and medium repairs while leaving million-node judgment calls to the user.

RESOURCES

Keep Reading

This page focuses on the Repertoire Builder as a standalone competitive workflow. For the rest of the app, use the main user guide and the other setup guides.

Download PDF Guide | Open Main User Guide | Install Lc0 Locally

Remote Leela Guide | Back to Homepage