

ShashChess

Introduction

ShashChess is a free UCI chess engine derived from Stockfish family chess engines. The goal is to apply Alexander Shashin theory exposed on the following book : <https://www.amazon.com/Best-Play-Method-Discovering-Strongest/dp/1936277468> to improve

- base engine strength
- engine's behaviour on the different positions types (requiring the corresponding algorithm) :
 - Tal
 - Capablanca
 - Petrosian
 - the mixed ones
 - Tal-Capablanca
 - Capablanca-Petrosian
 - Tal-Capablanca-Petrosian

Terms of use

Shashchess is free, and distributed under the **GNU General Public License (GPL)**. Essentially, this means that you are free to do almost exactly what you want with the program, including distributing it among your friends, making it available for download from your web site, selling it (either by itself or as part of some bigger software package), or using it as the starting point for a software project of your own.

The only real limitation is that whenever you distribute ShashChess in some way, you must always include the full source code, or a pointer to where the source code can be found. If you make any changes to the source code, these changes must also be made available under the GPL.

For full details, read the copy of the GPL found in the file named *Copying.txt*.

Files

This distribution of ShashChessPro consists of the following files:

- Readme.md, the file you are currently reading.
- Copying.txt, a text file containing the GNU General Public License.
- src, a subdirectory containing the full source code, including a Makefile and the compilation scripts makeAll.bat (Windows) and makeAll.sh (Linux).

Uci options

Hash Memory

Hash

Integer, Default: 16, Min: 1, Max: 131072 MB (64-bit) : 2048 MB (32-bit)

The amount of memory to use for the hash during search, specified in MB (megabytes). This number should be smaller than the amount of physical memory for your system. A modern formula to determine it is the following:

$(T \times S / 100)$ MB where T = the average move time (in seconds) S = the average node speed of your hardware A traditional formula is the following: $(N \times F \times T) / 512$ where N = logical threads number F = clock single processor frequency (MB) T = the average move time (in seconds)

Clear Hash

Button to clear the Hash Memory. If the Never Clear Hash option is enabled, this button doesn't do anything.

Threads

Integer, Default: 1, Min: 1, Max: 512 The number of threads to use during the search. This number should be set to the number of cores (physical+logical) in your CPU.

Ponder (checkbox)

Boolean, Default: True Also called "Permanent Brain" : whether or not the engine should analyze when it is the opponent's turn.

Usually not on the configuration window.

MultiPV

Integer, Default: 1, Min: 1, Max: 500 The number of alternate lines of analysis to display. Specify 1 to just get the best line. Asking for more lines slows down the search. Usually not on the configuration window.

UCI_Chess960 (checkbox)

Whether or not ShashChess should play using Chess 960 mode. Usually not on the configuration window.

Move overhead

Default 30, min 0, max 5000 In ms, the default value seems to be the best on Linux systems, but must be increased for slow GUI like Fritz. In general, on Windows system it seems a good value to be 100.

Slow mover

Default 84, min 10, max 1000 "Time usage percent": how much the engine thinks on a move. Many engines seem to move faster and the engine is behind in time clock. With lower values it plays faster, with higher values slower - of course always within the time control.

Handicap mode

UCI_LimitStrength

Activate the handicap mode and the related following options: in this case, the evaluation function is always the classical one.

UCI_Elo

Default 2850, min 1350, max 2850 UCI-protocol compliant version of Strength parameter. A very refined handicap mode based on the four famous sovietic chess school levels: Internally the UCI_Elo value will be converted to a Strength value according to the following table:

- *beginner: elo < 2000*
- *intermediate: 2000 <= elo < 2200*
- *advanced: 2200 <= elo < 2400*
- *expert: elo > 2400*

Every school corresponds to a different evaluation function, more and more refined. The UCI_Elo feature is controlled by the chess GUI, and usually doesn't appear in the configuration window.

Syzygy End Game table bases

Download at <http://olympuschess.com/egtb/sbases> (by Ronald De Man)

SyzygyPath

The path to the Syzygy endgame tablebases. This defines an absolute path on your computer to the tablebase files, also on multiple paths separated with a semicolon (;) character (Windows), the colon (:) character (OS X and Windows) character. The folder(s) containing the Syzygy EGTB files. If multiple folders are used, separate them by the ; (semicolon) character.

SyzygyProbeDepth

Integer, Default: 1, Min: 1, Max: 100 The probing tablebases depth (always the root position). If you don't have a SSD HD, you have to set it to maximize the depth and kn/s in infinite analysis and during a time equals to the double of that corresponding to half RAM size. Choose a test position with a few pieces on the board (from 7 to 12). For example:

- Fen: 8/5r2/R7/8/1p5k/p3P3/4K3/8 w -- 0 1 Solution : Ra4 (=)
- Fen: 1R6/7k/1P5p/5p2/3K2p1/1r3P1P/8 b -- 1 1 Solution: 1...h5 !! (=)

SyzygyProbeLimit

Integer, Default: 6, Min: 0, Max: 6 How many pieces need to be on the board before ShashChess begins probing (even at the root). Current default, obviously, is for 6-man.

Advanced Chess Analyzer

Advanced analysis options, highly recommended for CC play

Full depth threads

Integer, Default: 0, Min: 0, Max: 512 The number of settled threads to use for a full depth brute force search. If the number is greater than threads number, all threads are for full depth brute force search.

MonteCarlo Tree Search section (experimental: thanks to original Stephan Nicolet work)

MCTS

Default is Off: no MonteCarlo Tree Search algorithm. The other values are "Single" and "Multi", where "Single" means only main thread does MCTS and "Multi" means all threads but main one does MCTS

Multi Strategy

Integer, Default: 20, Min: 0, Max: 100 Only in multi mcts mode, for tree policy.

Multi MinVisits

Integer, Default: 5, Min: 0, Max: 1000 Only in multi mcts mode, for Upper Confidence Bound.

Live Book section (thanks to Eman's author Khalid Omar for windows builds)

Live Book (checkbox)

Boolean, Default: False If activated, the engine uses the livebook as primary choice.

Live Book URL

The default is the online chessdb https://www.chessdb.cn/queryc_en/, a wonderful project by noobpwnftw (thanks to him!)

<https://github.com/noobpwnftw/chessdb> <http://talkchess.com/forum3/viewtopic.php?f=2&t=71764&hilit=chessdb>

The private application can also learn from this live db.

Live Book Timeout

Default 5000, min 0, max 10000 Only for bullet games, use a lower value, for example, 1500.

Live Book Retry

Default 3, min 1, max 100 Max times the engine tries to contribute (if the corresponding option is activated: see below) to the live book. If 0, the engine doesn't use the livebook.

Live Book Diversity

Boolean, Default: False If activated, the engine varies its play, reducing conversely its strength because already the live chessdb is very large.

Live Book Contribute

Boolean, Default: False If activated, the engine sends a move, not in live chessdb, in its queue to be analysed. In this manner, we have a kind of learning cloud.

Live Book Depth

Default 100, min 1, max 100 Depth of live book moves.

Full depth threads

Default 0, min 0, max 512 The number of threads doing a full depth analysis (brute force). Useful in analysis of particular hard positions to limit the strong pruning's drawbacks.

Opening variety

Integer, Default: 0, Min: 0, Max: 40 To play different opening lines from default (0), if not from book (see below). Higher variety -> more probable loss of ELO

Concurrent Experience

Boolean, Default: False Set this option to true when running under CuteChess and you experiences problems with concurrency > 1. When this option is true, the saved experience file name will be modified to something like experience-64a4c665c57504a4.bin (64a4c665c57504a4 is random). Each concurrent instance of BrainLearn will have its own experience file name, however, all the concurrent instances will read "experience.bin" at start up.

Use NNUE

Toggle between the NNUE and classical evaluation functions. If set to "true", the network parameters must be available to load from file (see also EvalFile), if they are not embedded in the binary.

Persisted learning

Default is Off: no learning algorithm. The other values are "Standard" and "Self", this last to activate the [Q-learning](#), optimized for self play. Some GUIs don't write the experience file in some game's modes because the uci protocol is differently implemented

The persisted learning is based on a collection of one or more positions stored with the following format (similar to in memory Stockfish Transposition Table):

- *best move*
- *board signature (hash key)*
- *best move depth*
- *best move score*
- *best move performance* , a new parameter you can calculate with any learning application supporting this specification. An example is the private one, kernel of SaaS part of [ChessProbe](#) AI portal. The idea is to calculate it based on pattern recognition concept. In the portal, you can also exploit the reports of another NLG (virtual trainer) application and buy the products in the digishop based on all this. This open-source part has the performance default. So, it doesn't use it. Clearly, even if already strong, this private learning algorithm is a lot stronger as demonstrate here:

[Graphical result](#)

This file is loaded in an hashtable at the engine load and updated each time the engine receive quit or stop uci command. When BrainLearn starts a new game or when we have max 8 pieces on the chessboard, the learning is activated and the hash table updated each time the engine has a best score at a depth ≥ 4 PLIES, according to Stockfish aspiration window.

At the engine loading, there is an automatic merge to experience.bin files, if we put the other ones, based on the following convention:

`<fileType><qualityIndex>.bin`

where

- *fileType*="experience"/"bin"
- *qualityIndex* , an integer, incrementally from 0 on based on the file's quality assigned by the user (0 best quality and so on)

N.B.

Because of disk access, to be effective, the learning must be made at no bullet time controls (less than 5 minutes/game).

Read only learning

Boolean, Default: False If activated, the learning file is only read.

Shashin section

Default: no option settled The engine will determine dynamically the position's type starting from a "Capablanca/default positions". If one or more (mixed algorithms/positions types at the boundaries) of the three following options are settled, it will force the initial position/algorithm understanding

Tal

Attack position/algorithm

Capablanca

Strategical algorithm (for quiescent positions)

Petrosian

Defense position/algorithm (the "reversed colors" Tal)

Acknowledgments

- Sergey Aleksandrovitch Kozlov for his very interesting patch and code on Sugar engine
 - Omar Khalid for his great experience in microsoft c/cpp programming environment
 - Alexei Chernakoff for his pretious suggestions about the android version and its contribution to it
 - Dariusz Domagala for the Mac version
 - The BrainFish, McBrain, CorChess, CiChess and Crystal authors for their very interesting derivative
 - Obviously, the chess theorician Alexander Shashin, whitout whom I wouldn't had the idea of this engine
- Stockfish community

ShashChess team

- engine owner and main developer: ICCF IM Andrea Manzo (<https://www.iccf.com/player?id=241224>)
- IM Yohan Benitah for his professional chess understanding and help in testing against neural networks
- official tester: ICCF CCE and CCM Maurizio Platino (<https://www.iccf.com/player?id=241094>)
- official tester: Maurizio Colbacchini, FSI 1N
- official tester and concept analyst: ICCF GM Fabio Finocchiaro (<https://www.iccf.com/player?id=240090>), 2012 ICCF world champion
- official tester Dennis Marvin (NDL) (overall the online learning)
- tester and concept analyst: ICCF GM Matjas Pirs (<https://www.iccf.com/player?id=480232>), for his great experience and tests on positions analysis in different game's phases

Sorry If I forgot someone.