Dorky Chess 4.3

This software is released under the MS-PL, and a copy of the license is included with this package in license.txt

Dorky 4.0 was the first release of a new version in over 7 years and was a rewrite of most major components. Below is a list of chess programming practices and features used in Dorky Chess

- PVS alpha-beta with quiescent search
- SMP search using Dr. Hyatt's DTS algorithm
- End game tablebases (Nalimov)
- Late move reductions
- Futility pruning
- Transposition hash table/Pawn evaluation hash table (using Hyatt's lockless approach)
- Null-move pruning
- Killer move heuristic
- Bit boards, rotated bit boards for move generation
- Static exchange evaluation
- Book creation and simple learning
- Support for conventional and common time controls

Changes since version 4.0

4.1 Estimated Strength increase (+20 ELO)

1. Fixed draw by repetition detection. Dorky would allow repeats in potentially winnable positions.
2. Fixed an SMP bug that was causing threads to spin-wait on opponents turn (in easy mode). This bug also had the effect of only using one thread to ponder during opening book. Dorky ponders (when not in easy mode) while in the opening book to fill hash tables.

3. Found a bug in evasion generator that caused evasions involving a 2-square pawn advance to be missed for white. Added perft command usage: perft [depth], optionally add “divide”
4. Improved my tree data structure slightly, slimming it down.

4.2 (bug fix update only)

1. Fixed an issue with multiple cores when using the ‘force’ command. Not all cores being utilized properly.

4.3 (bug fix update only)

1. Fixed time control issue which caused Dorky to flag in second time control session. This was caused by book learning. When replaying the moves to score for book learning, the move count was not being reset properly. Also, the book file is no longer required for Dorky to play.

Using Dorky

Dorky is a Winboard compatible engine and supports most of the features of Winboard. There is some simple help within the application that will list all of the commands dorky accepts and a brief description of what they do. Commands that require parameters will describe the parameters if they are not provided. Dorky uses a .ini file, Dorky.ini, to configure a few properties. A sample is included with this distribution and below:

Sample Dorky.ini File:

```
n_cpus=1
resign_score=-1100

[Hash Table Section]
transposition_table=256
pawn_table=64

[EGTB Section]
tbpath=c:\tb
tbcache=64
```

n_cpus=n tells the Dorky to use **n** threads for searching. This can be changed on the fly using the winboard “cores” command.

resign_score=n tells the engine a negative value, below which to resign if search cannot find a higher value. Dorky scores pawns as 100, knights/bishops as 300, rooks as 500, and queens as 900.

transposition_table=n where n is the size, in megabytes, of the transposition table (hash table)

pawn_table=n where n is the size, in megabytes, of the pawn evaluation hash table. The actual amount of space allocated will vary based on addressing efficiency.

tbpath is the file path to nalimov end game tablebases

tbcache=n where n is the size, in megabytes, of the end game tablebase in-memory cache

Building an opening book

Dorky can create an opening book by reading a .PGN file. A simple one is included with this distribution that was generated from 2,400 games by Alekhine. Dorky considers how many times a move was played and how many times it led to a win, draw, or loss. If the move won more than twice as many times as it lost, it is stored. To build a book, at the console type “book create [path to .pgn file]”

As Dorky plays games, it will adjust the frequency with which moves are chosen based on scores from searches just after leaving the book line, and more greatly based on the result of the game.

Books created by a version of Dorky prior to 4.0 are incompatible with this version.

Thanks

I would like to thank Dr. Robert Hyatt, whom I consider to be the world's foremost expert on computer chess. I have learned the most about chess programming from Dr. Hyatt's writings and source. Thanks also to Eugene Nalimov for his egtb code. Thanks to H.G. Muller and Tim Mann for the Winboard software. Thanks to Tom Kerrigan for TSCP that interested and educated me, and lured me in, over a decade ago.