Performance Engineering of Chess Programs

Don Dailey

November 13, 2012
Section 1

A Brief History of Time
1948 - Astounding Claim

- UNIVAC, strongest computer in the world!
- No human can beat it at chess
- No human can beat it at gin rummy
- No actual chess program produced
1957 - Alex Bernstein at MIT

- First program to play a full game
- IBM 704 - 42,000 instructions per second
- 4 ply search in 8 minutes!
1957 - Herbert Simon

- Computer will be world champion within 10 years
1962 - Alan Kotok of MIT

- B.S. thesis project chess program
- Assisted by John McCarthy of Stanford
- Played “credible” chess
- 1100 positions per second
1966 - Richard Greenblatt.

- Mac Hack program
- Enters 1966 Massachusetts Amateur Championship
- 1 draw, 4 losses for provisional rating of 1243
1967 - Mac Hack rules!

- Mac Hack VI becomes first program to beat human
- Human rated 1510
- By end of 1967 record is 3 wins, 12 losses and 3 draws
- Becomes honorary member of US Chess Federation.
- First widely distributed chess program, running on man PDP machines.
- First to have an opening book distributed with it.
1968 $3,000 bet
No program will beat him in 10 years.
Levy wins this bet 10 years later.
1970 - Computer Chess Era Begins

- In 1970 the first all-computer championship was held in New York.
- Won by Chess 3.0 on a CDC 6400.
- Written by Slate, Atkin and Gorlen at Northwestern.
- 6 programs had entered.
1977 - Microcomputers appear

- Chess Challenger is marketed in 1977
- ICCA founded by Computer Chess Programmers.
Levy bet results

- 1977 David Levy beat Kaissa as part of bet.
- Score of Chess 4.7 match is 3 wins and one draw
- First time computer draws IM in serious game
- Experts once again predict that computers will beat world champion in 10 years
Master Level

- 1981 Cray Blitz wins Mississipi State Championship
- 5-0 score
- Beats 2262 player (master is 2200+)
- Attains master performance rating
Belle

- Authored by Ken Thompson of Unix fame
- Created in Late 70’s
- Attains masters rating of 2203 in 1983
- PDP-11/23 with many custom boards
- 4 custom boards for evaluation
- 3 custom boards for move generation
- 1 megabyte of memory for transposition tables
1988 Deep Thought

- Achieves 2745 performance rating in US Open
- In November gets official rating of 2550
Levy Prize

- In late 1970’s David Levy offers $1,000 prize
- First program to defeat him in 4 or 6 game match
- Omini magazine adds $4,000 to this.
- In 1989 Deep Blue collects prize
1994 - Kasparov

- Fritz beats Kasparov in Blitz tournament.
- Kapsarov demands rematch and beats Fritz 4-2
8th WCCC

- May, 1995 in Hong Kong
- Leiserson program humiliated by Fritz in playoff
- Leiserson program running on 1824 node Paragon
- Fritz running on 90 MHZ Penium
Kasparov beats Deep Blue in 1996

- But Deep Blue wins first game
- First victory by computer against Champ
Section 2

Era Of Machine Dominance
Deep Blue Revenge

- 1997, Deep Blue defeats Garry Kasparov
- 6 game match in New York
- 200 million nodes per second
GM Joel Benjamin plays match with Rybka
Rybka plays without one of it pawns as handicap
Rybka wins match 4.5 - 3.5.
2 wins, 1 loss, 5 draws
GM Joel Benjamin vs Rybka
Benjamin gets White in every game
All draws scored as a win for Benjamin
Rybka defeated GM Joel Benjamin with a 6-2 score
Section 3

Software Engineering for Chess
perft - a useful unit test

- Count end nodes to depth N from a specified position
- $\text{perft(1)} = 20$
- $\text{perft(2)} = 400$
- $\text{perft(3)} = 8902$
- $\ldots$
- $\text{perft(13)} = 1,981,066,775,000,396,239$
- Almost 2 quintillion
The result isn’t very large. When printed to paper, it’s only about five centimeters long.
ELO Rating System

- How chess strength is measured
  - expectation = $1 / (1 + 10^{((R0 - R1) / 400)})$
  - 1500 vs 1400 = 0.640
  - 1500 vs 1499 = 0.501
  - 2000 vs 1500 = 0.947
  - 3000 vs 2000 - 0.9968
Automated Testing

<table>
<thead>
<tr>
<th>Rank</th>
<th>ELO</th>
<th>+/-</th>
<th>Games</th>
<th>Score</th>
<th>Player</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3000.0</td>
<td>17.7</td>
<td>802</td>
<td>51.683</td>
<td>4507.00</td>
</tr>
<tr>
<td>2</td>
<td>2988.3</td>
<td>17.7</td>
<td>802</td>
<td>48.317</td>
<td>Beta-5+2R</td>
</tr>
</tbody>
</table>

Which program is stronger?
Error Margins

<table>
<thead>
<tr>
<th>Rank</th>
<th>ELO</th>
<th>+/-</th>
<th>Games</th>
<th>Score</th>
<th>Player</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3000.0</td>
<td>17.7</td>
<td>802</td>
<td>51.683</td>
<td>4507.00</td>
</tr>
<tr>
<td>2</td>
<td>2988.3</td>
<td>17.7</td>
<td>802</td>
<td>48.317</td>
<td>Beta-5+2R</td>
</tr>
</tbody>
</table>

- Error margin is a measure of confidence
- Not valid unless size of sample determined in advance
- 2 sigma means 95.45% runs would lie between range of 2970.6 - 3006.0
Bell curve

Figure: Bell curve
Determinism

- It is useful if your program is deterministic
- Run some fixed depth search from the opening.
- Note the final score and Nodes searched.
- Make a coding improvement
- Run again to same depth and compare
Dual routines

- It is useful to write same routine 2 different ways
- Put debug code to run both and compare.
- One way simple and slow, the other way sophisticated.
Example of Dual routines

- Incremental zobrist hashing
- Create routine to calculate hash from scratch.
- After each move compare with incremental hash
- If not identical, you have a bug

Principal: work incrementally and carefully. Check everything as you go.
Section 4

Performance
The Importance of CPU performance

Figure: Elo Graph
Graph data

- y-axis is rating in ELO
- x-axis is level in nodes, i.e. $2^N$ nodes
- fixed nodes
- approximately 100 points per doubling

Conclusion: CPU performance is critical!
Move Extensions

- High compulsion moves should be extended
- Some checks
- Some captures
- Singular moves
Iterative deepening

- Take advantage of information being built up
- Hash table benefits
- Good mechanism for time control
Null Move Observation

For the side to move, there is almost always a better alternative than doing nothing.

- Basis of null move pruning in modern programs
- Futility pruning
- Quies search
- Doesn’t hold in all games.
More On Futility pruning

- Don’t try moves unlikely to help
- Speculative
- Based on null move observation
- Limited to frontier nodes

\[
e = \text{static}\_\text{eval}(\text{pos});
\]

if \( e < \alpha - \text{MARGIN} \) then try only captures and checks
Killer Moves

- If it’s not broken, don’t fix it
- Moves that work well in one position usually work well in others
Incremental Zobrist Hashing

- Table driven, fast, incremental
- xor of random values from table
- Can be computed incrementally
- operations: remove and add are same

To remove: \( \text{sig} = \text{sig} \oplus \text{zob[who][where]} \)

To add: \( \text{sig} = \text{sig} \oplus \text{zob[who][where]} \)
Global Transposition Table

- Remember results of previous searches
- Not same as memoization but similar
- Search is not a “pure” function
- Alpha and Beta is a complication
Transpostion table records

- zobrist key
- score
- move
- quality or depth searched
- bounds (upper, lower or exact)
- age
Quies search

- Based on null move observation
- Each side tries captures only
- Each side has option to "stand pat"
Testing methodology

- Modern programs are hard to improve
- Progress is made one ELO at a time
- Massive automated testing
- Most improvements are transitive
- The days of finding a nice trick to add 100 ELO is long gone
- But not the case for Leiserchess!
Orthogonal multi-testing

- Proposed by H.G. Muller
- Test many changes simultaneously
- Assumes most changes are independent

http://komodochess.com/asdf/MULT/multi.html
Move ordering

- Move Ordering incredibly important
- alpha/beta cutoffs
- Late move reductions
Typical Chess move ordering

- hash table move
- internal iterative deepening
- non-losing capture in MVVLVA order
- Killers
- losing captures
- other moves (see next slide)
History Heuristic for move ordering

- Originally proposed by Schaeffer
- 64x64 butterfly boards
- if move is “best” increment counter
Probelms with Schaeffer Heuristic

- Move become irrelevant with deep searches
- Scores only increase
Improvement to History

- Make it piece specific (pce/to)
- Keep win/loss statistics
- Heavily weight recent entries
Late Move Reductions (LMR)

Observation:
With good move ordering a beta cutoff will either occur right away, or not at all.
LMR

- Search first N moves normally
- Reduce depth for later moves
- If fail high, research
- High compulsion moves
Forward Pruning

- Similar to LMR
- Only near frontier nodes
- Depends on good move ordering
- Look at N moves and throw out the rest
- Take care with high compulsion moves
SEE

- Static Exchange Evaluator
- Quies search without search
- Swapoff on a single target square
- Powerful pruning mechanism
- High error rate
Game Phase scoring

- Berliner observation about game transitions
- Many older programs have 2 evaluations
- Modern programs have 2 computed in parallel
- Pack 2 scores into single 64 bit integer
- Interpolation of scores
Time control

- Fischer time control
- Standard time control
- Important to allocate your time wisely
Time Allocation

- Use much more time early in game
- Finish iteration if possible
- Don’t start something you can’t finish
- Not all moves are created equal
- Node ratio between first move and rest
Summary

- Automated testing is crucial
- Interpretation of test results important
- Speed kills (the opponent)
- The only good bug is a dead bug
One final observation

- Your program is going to be 2000 ELO weaker than it could be
- Leiserchess is a new game - a lot is not known