History of Computer Chess

- “The Turk” – Baron Wolfgang von Kempelen – Early 19th Century
- First real algorithm – Turing (1947)
- 1948 – UNIVAC was supposedly “unbeatable”!
- 1952 – Turing postulated that computers would eventually become more powerful than humans
- 1958 – First victory for computer vs. human
- 1966 – Russian program beats US program
- 1968 – David Levy’s famous 10 year bet
- 1970 – First ever all-computer tournament
History (continued…)

• 1977 – ICCA founded
• 1977 – GM Michael Stean loses in blitz to a computer
• 1981 – Cray Blitz wins Mississippi state championship with perfect 5.0 score
• 1988 – DEEP THOUGHT shares top place in US Chess Championship
• 1992 – Fritz 2 defeats (W-C) Kasparov in speed chess
• Feb 1996 – Kasparov beats IBM Deep Blue (4 – 2)
• May 1997 – Deep Blue defeats Kasparov (3.5 – 2.5)
• Oct 2002 – Deep Fritz draws with (W-C) Kramnik 4-4 in “Brains in Bahrain” match
• Jan 2004 – ChessBrain plays 1st match against GM Nielsen
How Chess Programmes Work

Board representation

Tree search

Board Evaluation

Precalculated Data

http://www.chessbrain.net/

http://www.frayn.net/beowulf/theory.html
(1)

Board Representation
# Algebraic Board Notation

<table>
<thead>
<tr>
<th>1</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Information to be Stored

- Board position (location of all pieces)
- En-passant square, if any
- Castling permissions
- Draw by repetition / 50-move stats (often stored outside the board structure)
- Side to move
Board Storage

• 8*8 integer array with piece keys
• [-6 <= p <= 6] e.g. empty=0, wpawn=1, bpawn=-1
• Extended board representation
• 12*12 integer array with border to save branches when testing move legality
• Bitboard representation – Now used by all major commercial programmes:
  • Represent board using a set of 64-bit numbers.
Bitboard Example

White Pawns :
0000000000000000000000000000000000000000000000000000000000000000

Black Pawns :
0000000000000000000000000000000000000000000000000000000000000000

Etc…

• For each board position : 12 Bitboards (12 * 64 bits) +
  Castling (4 bits) + side to move (1 bit) + E-P square (6 bits) =
  779 bits (98 bytes) [c.f. other methods]

• Can improve this e.g. no need to store BB for kings
Bitboards

• Complex, but fast (esp. on 64-bit arch.)
• Board is stored using 12 64-bit numbers
  (One per colour per piece-type)
• Move generation is now much quicker
  e.g. Pseudo-legal pawn moves:
  \[ \text{SimplePawnMove} = (\text{PawnWhite} >> 8) \& \sim(\text{AllPieces}) \]
• Sliding moves are more complicated, but still very fast
Bit Operations

- **OR** – Combine two boards
- **AND** – Use a mask on a board
- **XOR** – Flip bits on a board

```c
#define Remove(a,b)  ((a) = (a^(1<<b)))
#define RemoveFirst(a)   ((a) = ((a) & ((a)-1)))

int FirstPiece(BITBOARD B) {
    if (B&TwoFullRanks) return first_piece[B&TwoFullRanks];
    if (B&FPMask1) return first_piece[(B >> 16)&TwoFullRanks] + 16;
    if (B&FPMask2) return first_piece[(B >> 32)&TwoFullRanks] + 32;
    return first_piece[B >> 48] + 48;
}
```
/* Generate White Knight Moves */

Knights = B->WhiteKnights;

/* ‘Knights’ holds a board of all possible knights */
while (Knights) {
    /* Get the first available knight */
    from = FirstPiece(Knights);
    /* Mask out illegal moves */
    Dest = KnightMoves[from] & ~(B->WhitePieces);
    /* Add potential moves to global movelist */
    AddMovesToList(from,Dest);
    /* Remove this knight from the list */
    RemoveFirst(Knights);
}

000000000000000000000000010100001000100000000000000000000000001010000
/ * Generate Black Rook Moves */
Rooks = B->BlackRooks;
/* ‘Rooks’ holds a board of all possible rooks */
while (Rooks) {
    from = FirstPiece(Rooks);
    /* First generate horizontal moves */
    mask = (B->All >> (Rank(from)*8)) & FullRank;
    Dest = MovesRank[from][mask];
    /* Next generate vertical moves */
    mask = (B->R90 >> (File(from)*8)) & FullRank;
    Dest |= MovesFile[from][mask];
    /* Mask out illegal moves */
    Dest &= ~(B->BlackPieces);
    /* Add potential moves to global movelist */
    AddMovesToList(from,Dest);
    /* Remove this rook from the list */
    RemoveFirst(Rooks);
}
Making a Simple Knight Move

/* Sort out the new board state */
B->WhiteKnights ^= Mask[from] | Mask[to];

/* Test for Capture */
switch (CapturedPiece) {
    case (bpawn) : B->BlackPawns ^= Mask[to]; break;
    case (brook) : B->BlackRooks ^= Mask[to]; break;
    case (bknight) : B->BlackKnights ^= Mask[to]; break;
    case (bbishop) : B->BlackBishops ^= Mask[to]; break;
    case (bqueen) : B->BlackQueens ^= Mask[to]; break;
}

/* Check for alterations to castling permissions */
switch(from) {
    case (a1) : B->castle &= 13; break;
    case (h1) : B->castle &= 14; break;
    case (e1) : B->castle &= 12; break;
}
switch(to) {
    case (a8) : B->castle &= 7; break;
    case (h8) : B->castle &= 11; break;
}
(2)

Tree Search
Tree Search Fundamentals

- Computers work recursively, like humans
- Computers have no (little) intuition
  …so must work by (intelligent) brute force

(Q) Which is the best move in any position?
(A) The one with the weakest ‘best reply’

Note: This is not (always) the move with the best chances
Search Recursion
Termination of the Search

We could continue this recursive search forever BUT it is going to get prohibitively slow.

Average branching factor = 35

Average game length = 50 moves

Total Nodes = $35^{50*2} \approx 10^{154}$

(There have been $5 * 10^{17}$ seconds so far…)

So we must terminate the search prematurely
Quiescence Search

We can’t just “stop searching”

- The Horizon effect
- Perform a quiescence search until the board is ‘calm’!
- Search only ‘non-quiescent’ moves (to a maximum depth)

Definitions vary. I use:

- Captures
- Pawn promotions (or near-promotions)
- Checking moves
Iterative Deepening

• Start with a shallow search
• Using information from this search, look slightly deeper
• Keep increasing the depth until we run out of time, or find a definite outcome
• This is the most efficient way to search
• May look wasteful, but safeguards against extremely damaging mistakes!
• Allows us to use knowledge to improve searches
# Tree Pruning

<table>
<thead>
<tr>
<th>Depth</th>
<th>Node Count</th>
<th>Search Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 ply</td>
<td>900</td>
<td>0.005s</td>
</tr>
<tr>
<td>3 ply</td>
<td>27,000</td>
<td>0.14s</td>
</tr>
<tr>
<td>4 ply</td>
<td>810,000</td>
<td>4.05s</td>
</tr>
<tr>
<td>5 ply</td>
<td>24,300,000</td>
<td>2 mins</td>
</tr>
<tr>
<td>6 ply</td>
<td>729,000,000</td>
<td>1 hour</td>
</tr>
<tr>
<td>7 ply</td>
<td>21,870,000,000</td>
<td>30 hours</td>
</tr>
<tr>
<td>8 ply</td>
<td>656,100,000,000</td>
<td>38 days</td>
</tr>
</tbody>
</table>

Branching factor 30, evaluating 200,000 nodes/sec
Tree Pruning Methods

- Negamax Search
- Iterative Deepening
- Alpha-Beta Pruning
- Principal Variation Search
- Aspiration Windows
- Transposition Table
- Killer Moves
- History Heuristic
- Internal Iterative Deepening
- Null Move Heuristic
- Futility Pruning
- Razoring
- Search Extensions
- etc….
Alpha-Beta Search

• Enormously reduces search tree size
• Alpha-beta pruning is just a complicated-sounding name for “Don't check moves which cannot possibly have an effect on the outcome of your search.”
• Reduces the branching factor from 30-40 to 6-7. Approximately doubles search depth.
• Absolutely vital to any strong engine
Imagine we’re searching all the possible moves in a position and the best move so far scores +15 centipawns.

We continue to the next move and start examining the opponent’s replies:

-20,-22,-19,-16,-11,-18,-20,-30,-70,-75,-55,-2,-628,-M3

We abort as soon as we get to the ‘-11’

Doesn’t matter what the rest of the moves score – the ‘-11’ is already good enough even though the opponent can actually do better (-2)

Best moves should be as near the front as possible!
Alpha-Beta Pseudo-code

Initially alpha = -INFINITY, beta=INFINITY

search(position, side, depth, alpha, beta) {

    best_score = -INFINITY

    for each move {
        do_move(position, move)

        if (depth is 0) move_score = static_score(position, side)
        else move_score = -search(position, opponent side, depth-1, -beta, -alpha)

        undo_move(position, move)

        if (move_score > best_score) best_score = move_score
        if (best_score > alpha) alpha = best_score
        if (alpha >= beta) return alpha
    }

    return best_score
}
Transposition Table

- To save calculation, there is no point evaluating positions that we’ve seen before!
- Every result is stored on a priority basis (most expensive first, most important first)
- Replacement schemes vary
- When we find a ‘hit’ we see if we can make use of the new information
- Must be careful about depths and bounds
- Makes sense with Iterative Deepening
- Individual key for each board position (we hope – clashes do happen)
- Key is generated using XOR of 64 random numbers
Other Tree Pruning Algorithms

- History Heuristic
- Killer Moves Heuristic
- Razoring
- Extensions
- Window search
- Null move search
(3) Board Evaluation
Static Board Evaluation

- A computer has no *a priori* chess knowledge
- Humans evaluate positions using numerous methods (‘intuition’)
- The goal of computer chess is to mimic the concept of intuition
- Brute force vs. Intelligence
- Diminishing returns in brute force?
- The intelligent way to proceed – add knowledge
A typical board evaluation

Current Position

White Points: 39
Black Points: 39
(Even Sides)

Lazy Eval

Game Stage = 0 [0=Opening, 5=Late Endgame]

Material Eval : 0
Positional Eval : 0
Total Lazy Eval : 0

Full Eval

Square a8 [r] : Blocked -3, HBlock 1, [-5]
Square b8 [n] : Opp.KTrop. -10, Bad Develop. -12, [-19]
Square c8 [b] : Bad Develop. -12, Mobility -12, [-24]
Square d8 [q] : KAtt 3, Trapped -10, Quarts 0 (-15), [-22]
Square e8 [k] : {bqnpnpppp DEF=6, Sh 3 [43]}, [43]
Square f8 [b] : Bad Develop. -12, Mobility -12, [-24]
Square g8 [n] : Opp.KTrop. -9, Bad Develop. -12, [-19]
Square h8 [t] : Blocked -3, HBlock 1, [-5]
Square a7 [p] : DEF=1, DefSc 4, [4]
Square b7 [p] : DEF=1, DefSc 4, [4]
Square c7 [p] : DEF=1, DefSc 4, [4]
Square d7 [p] : DEF=4, DefSc 16, [16]
Square e7 [p] : DEF=4, DefSc 16, [16]
Square f7 [p] : DEF=1, DefSc 4, [4]
Square g7 [p] : DEF=1, DefSc 4, [4]
Square h7 [p] : DEF=1, DefSc 4, [4]
Square a6 [.] : -3
Square b6 [.] : -4
Square c6 [.] : -4
Square d6 [.] : -4
Square e6 [.] : -4
Square f6 [.] : -4
Square g6 [.] : -4
Square h6 [.] : -3
Square a5 [.] : 0
Square b5 [.] : 0
Square c5 [.] : 0
Square d5 [.] : 0
Square e5 [.] : 0
Square f5 [.] : 0
Square g5 [.] : 0
Square h5 [.] : 0
Square a4 [.] : 0
Square b4 [.] : 0
Square c4 [.] : 0
Square d4 [.] : 0
Square e4 [.] : 0
Square f4 [.] : 0
Square g4 [.] : 0
Square h4 [.] : 0
Square a3 [.] : 3
Square b3 [.] : 4
Square c3 [.] : 4
Square d3 [.] : 4
Square e3 [.] : 4
Square f3 [.] : 4
Square g3 [.] : 4
Square h3 [.] : 3
Square a2 [p] : DEF=1, DefSc 4, [4]
Square b2 [p] : DEF=1, DefSc 4, [4]
Square c2 [p] : DEF=1, DefSc 4, [4]
Square d2 [p] : DEF=4, DefSc 16, [16]
Square e2 [p] : DEF=4, DefSc 16, [16]
Square f2 [p] : DEF=1, DefSc 4, [4]
Square g2 [p] : DEF=1, DefSc 4, [4]
Square h2 [p] : DEF=1, DefSc 4, [4]
Square a1 [R] : Blocked -3, HBlock 1, [-5]
Square b1 [N] : Opp.KTrop. -10, Bad Develop. -12, [-19]
Square c1 [B] : Bad Develop. -12, Mobility -12, [-24]
Square d1 [Q] : KAtt 3, Trapped -10, Quarts 0 (-15), [-22]
Square e1 [K] : {PPPPBQBN DEF=6, Sh 3 [43]}, [43]
Square f1 [B] : Bad Develop. -12, Mobility -12, [-24]
Square g1 [N] : Opp.KTrop. -9, Bad Develop. -12, [-19]
Square h1 [R] : Blocked -3, HBlock 1, [-5]

Effective Castling: NONE
White Not Castled: -8
Black Not Castled: 8
White centre pawns: 6
Black centre pawns: 6
Positional Score: 0
White has a bishop pair: [+15]
Black has a bishop pair: [-15]
Final Score: 0 [Delta 0]

Static Analysis Score
Exactly Even Sides

Approx. 3,200 lines!
Evaluating Defences

(White +0.00)

Rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq -
Evaluating Defences 2

(White +0.41)

r1bqkb1r/1p2pppp/p1n1n2/1B6/3NP3/2N5/PPP2PPP/R1BQK2R w KQkq -

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-2</td>
<td>-2</td>
<td>-2</td>
<td>-2</td>
<td>-2</td>
<td>-2</td>
<td>0</td>
</tr>
<tr>
<td>-2</td>
<td>-1</td>
<td>-1</td>
<td>-4</td>
<td>-4</td>
<td>-1</td>
<td>-1</td>
<td>-2</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>-2</td>
<td>-1</td>
<td>-2</td>
<td>-2</td>
<td>0</td>
</tr>
<tr>
<td>-2</td>
<td>1</td>
<td>-1</td>
<td>1</td>
<td>-2</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>-1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Notice Bb5!
What else is in the Eval?

- Individual positional bonuses for each piece
- Specific bonuses for piece tactics
- King safety
- Board control
- Passed pawns
- Endgame evaluation
- Special cases / positional knowledge
Knowledge vs. Strength

• Trade off eval complexity versus tactical search speed
• We are beginning to see diminishing returns
• Way forward – Moore’s Law & advanced analysis
• Expert testing
• ‘Never Tests’
(4) Precalculated Data
Internal Precalculations

- Lots of algorithms require precalculated data
- E.g. sliding piece moves, knight moves, masks, corner avoidance
- King area, square control, positional weighting
- Penalties / bonuses based on game stage
- Hash table initialisation
Opening Books

- All professional programs use these
- The opening position is too complicated
- Often strategies prove good/bad after 20-30 moves
- Usually generated from GM game databases
- Saves computation time
- Can be tweaked by a GM-level player
Endgame Tablebases

- Allow perfect play for the very end of the game
- 2-piece is a trivial draw
- 3-piece is fairly simple (381k)
- 4-piece is significantly larger (40Mb compressed)
- 5-piece is huge (8.2Gb compressed)
- 6-piece (Computational time & size prohibitive – 1.2Tb?)
- Most games are over well before this stage (but allows computer to use obscure tactics)
- Longest forced CM (3-, 4-, 5-piece) = 56, 85, 254 moves. Unknown for 6-piece.
- More examples – e.g. longest forced mate from underpromotion = 199 plies (5-piece)
(5)
The ChessBrain Project
Project Overview

- ChessBrain was founded in January 2002 by Carlos Justiniano
- CMF joined in summer 2002
- ChessBrain played its first ever match just after Christmas 2002
- ChessBrain is the world’s leading distributed chess project

http://www.chessbrain.net/
How does it work?
World Record Attempt

- Friday 30th January, 2004 in Copenhagen, Denmark
- Opponent was Peter Heine Nielsen (#1 in Denmark, 2620 ELO)
- The game lasted 34 moves
- 2,070 individual contributors from 56 countries
- The result was an agreed draw
- Official Guinness World Record
Some game statistics

PeerNode distribution per 15 minute interval

Max per interval: 1923
Average nodes in window: 1471

PeerNode client distribution across IP address range
(6)
The End?
The Future

• Chess will almost certainly NEVER be solved
• EGTBs are already getting too large – it is unlikely that anyone will use more than 6-man TBs
• Computers are now as good as the world champion.
• Processing power is linked to playing strength (*2 speed = ~+75 ELO points)
• GM players only analyse 2 or 3 positions per second!
• Diminishing returns – the future is in creating more “human” evaluation functions
• They will be in a league of their own within 5 years
• Chess as a test for complex computational frameworks?
Questions