Computer Chess
Anthony Cozzie
Computer Chess

- Quite possibly the nerdiest activity in the world
- But actually more fun than human chess
- Zappa
  - alpha-beta searcher, with a lot of tricks
  - World Computer Chess Champion 2005
  - Much, much better than Deep Blue
Nolot 1: Kasparov-Karpov

- Key move N×h6
- Novag Diablo: 7.5 months
  - 1991, 16MHz 16-bits
- Deep Blue: 6 hours
- Zappa: 2 minutes
Nolot 4: Bronstein-Kotov

- Key move Nxe6
- TASC R30: 1 month
  - 1993, 30MHz ARM
  - Chessmaster engine
- Deep Blue: 2.5 hours
- Zappa: 1 minute
Nolot 6: Malaniouk - Ivanchuk

- Key move axb5
- TASC R30: centuries
- Deep Blue: gave up
  - (more system time than they cared to use, they estimated at a few days)
- Zappa: 10 minutes
This Talk:

- Will feel like a collection of tricks
- Search (tweaks to alpha-beta)
  - Move ordering
  - Extensions
  - Forward Pruning
  - Branching factor 2-3 instead of $\sqrt{35}$
- Evaluation
  - Automatic tuning
When do we stop searching?
Depth 1: capture the biggest piece we can, regardless of whether opponent can recapture
Qsearch: try captures, checks, check evasions
  - No depth parameter
  - Can stand pat (terminate if eval>beta)
  - Prune stupid captures
Transposition table

- Big hash table storing the results of the search
- Prevents engine from researching the same position (still exponential time, though)
  - 1. e₄ d₆ 2. d₄
  - 1. d₄ d₆ 2. e₄
- 64-bit key Zobrist hash key
  - \( (\text{hashA}_1[\text{pieceA}_1] \oplus \text{hashA}_2[\text{pieceA}_2] \oplus \ldots) \)
Move Ordering

- The fine print of alpha-beta: “Thou must search thy best move first”
- Iterative deepening
- Store best move transposition (hash) table
- History heuristic:
  - Table[piece-type][start-square][dest-square]
  - Increment when move fails high
- Result: 93% of fail highs occur on the first move
What happens if we search at the root with a smaller window?

Alphabeta(a, a+1) \(\rightarrow\) is score > a?
- All over the place in chess engines

Aspirated alpha beta (change root score)

MTD(f): continue to try zero-window searches until a goal is reached
- (not in Zappa)
int NegaScout ( position p; int alpha, beta ) {
    /* compute minimax value of position p */
    int a, b, t, i;
    if ( d == maxdepth )
        return Evaluate(p); /* leaf node */
    determine successors p_1,...,p_w of p;
    a = alpha;
    b = beta;
    for ( i = 1; i <= w; i++ ) {
        t = -NegaScout ( p_i, -b, -a );
        if ( (t > a) && (t < beta) && (i > 1) && (d < maxdepth-1) )
            a = -NegaScout ( p_i, -beta, -t ); /* re-search */
        a = max( a, t );
        if ( a >= beta )
            return a; /* cut-off */
        b = a + 1; /* set new null window */
    }
    return a; }
Null Move

- Having the move is usually good in chess
- If we can skip our turn and still win, we will win bigger without skipping
- Depth reduction $R$, usually $2/3$ or $3$
  - Cut a depth 10 search if we could skip our turn and still win with a depth 6 search
- Worth a few plies of search depth
Forward Pruning ?!

- Short-Timman
- Search looks like:
  Kh2 NULL Kg3 NULL
  Kf4 NULL Kg5 NULL
  Kh6 NULL Qg7 (10 + 5R = 25 ply)
When the evaluation is wrong

- Suppose Zappa searches
  - A B C D E F G with score 0.1
  - A B C D E F G H with score -1
- Can play different moves instead of A C E G
  - → Deeper search implies more places to change!
  - Alpha-beta is completely blind to whether alternatives are OK or terrible
One move is much better than the alternatives
- Classic example; QxQ PxQ
- Check with additional searches of (d-m, m usually 4-5 ply)
  - Costs nodes because normally at cut nodes we only search one move
- Originally done by Deep Blue team
Evaluation: Material

- Count pieces
  - Standard chess weights are 1/3/3/5/9
  - Zappa uses 1/4/4/6/12
- Chess is easier than Go
  - Smaller branching factor (35 moves vs 200)
  - Material is the #1 term in the evaluation
Evaluation: Mobility

- Count squares a piece can attack
- Queen > Bishop because queen attacks more squares
- Potentially move wins
- Random eval works!
Pretty chess specific
Two connected passed pawns on the 6th rank are worth about a rook to Zappa
A king with no pawn protection is about 1-2 pawns
  - Rises to 5+ when attacked by a bunch of pieces
About 1000 heuristics, 5,000 terms.
Very annoying to tune by hand
  - Fix one position, break another
Have some test, use genetic programming or simulated annealing
Problem is it takes ~50,000 games to determine a few elo improvement
Open problem
Also: automatically generating new terms
Questions?

- Just ideas discussed here gives you a super-GM strength program
- http://chessprogramming.wikispaces.com/
- But be careful, because computer chess is surprisingly addictive