Tillämpad AI

Föreläsning 3, spelprogram I

Why Game-Playing?

- **Simple to formulate and describe**
- **Limited and clear domain**
- **Does not need lots of general knowledge**
- **Easy to test**
- **Supposed to demand intelligence**
- **Fun...**

Computer Chess

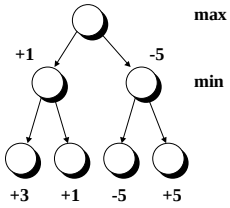
- **Branching factor ~36**
- **Depth > 40 moves (= 80 half moves or plies)**
- **N = 36^{80} > the number of atoms that would fill the visible universe**
- **Heuristics needed**

Parts of a Chess Program

- **Move generator(s)**
- **Make/retract - operators to make moves**
- **Evaluation function**
- **Search method**
- **High efficiency**

Minimax Search

- **Needs evaluation function**
- **Depth First**
- **Fixed depth**
- **Optimal solution within search horizon**
- **Efficient to implement**



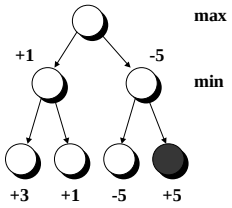
Minimax

```
int NegMax ()
{
  int v, Max = - MAXVALUE;

  if (Ply == MaxPly) return WhiteToMove ? Eval () : - Eval ();
  GenerateMoves ();
  loop (Move) {
    Make (Move); Ply ++;
    v = - NegMax ();
    Ply --; Retract (Move);
    if (v > Max) Max = v;
  }
  return Max;
}
```

Alpha-Beta Algorithm

- Equivalent to Minimax
- N' = 2 * sqrt (N)
- Improvement depends on move ordering
- More heuristics needed



Alpha-Beta

```
int AlphaBeta (Alpha, Beta)
{
  if (Ply == MaxPly) return WhiteToMove ? Eval () : - Eval ();
  GenerateMoves ();
  loop (Move) {
    Make (Move); Ply ++;
    v = - AlphaBeta (- Beta, - Alpha);
    Ply --; Retract (Move);
    if (v > Alpha) {
      Alpha = v;
      if (Alpha >= Beta) break;
    }
  }
  return Alpha;
}
```

Alpha Beta at Root

```
Value = AlphaBeta (- Limit, Limit);
```

Heuristic Improvements

- Move sorting
  - A priori knowledge
  - Evaluation function
  - Captures first
  - Principal variation
  - Killers
  - History
- Staged alpha-beta
  - Iterative depth-first
- Incremental evaluation
- Incremental generation
- Make/retract

Search Improvements

- Aspiration window for root node
- Scout search, minimal window search
- Hash table
- Quiescence search
- Null moves
- Singular extensions
- Knowledge-based selectivity

Aspiration Window

```
Value = AlphaBeta (- Limit, Limit);
if (Value <= - Limit) {
  Value = AlphaBeta (- MAXINT, Limit);
} else if (Value >= Limit) {
  Value = AlphaBeta (- Limit, MAXINT);
}
```

Scout

```
int Scout (Alpha, Beta)
{
  if (Ply == MaxPly) return WhiteToMove ? Eval () : - Eval ();
  GenerateMoves ();
  ScoutBeta = Beta;
  loop (Move) {
    Make (Move); Ply ++;
    v = - Scout (- ScoutBeta, - Alpha);
    if (v > Alpha && ScoutBeta != Beta) v = - Scout (- Beta, - Alpha);
    Ply --; Retract (Move);
    if (v > Alpha) {
      Alpha = v; if (Alpha >= Beta) break; ScoutBeta = Alpha + 1;
    }
  }
  return Alpha;
}
```

Evaluation

- **Material, 1000, 3000, 3000, 5000, 9000**
- **Piece location value tables**
  - `Value += Plv [Piece] [To] - Plv [Piece] [From];`
- **Second-order evaluation**
  - **Mate, stalemate, draw by repetition**
  - **Centrality, king proximity, mobility, pawn structure**
- **Staged evaluation**
  - **if (Alpha - T < Value && Value < Beta + T) Evaluate ();**

Shannon's Classification

- **Type A: full width, fixed depth**
- **Type B: fixed width, fixed depth**
  - **Width may be function of depth, w = f (d)**
- **Type C: variable width, variable depth**
- **Current standard**
  - **A plus captures (quiesence)**
  - **B plus captures**

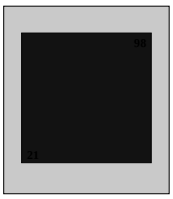
More Parts

- **Opening library**
  - **Databases of 20 000 to 500 000 games**
- **Endgame**
  - **Special evaluation parameters**
  - **Rule-based evaluation, KPK**
  - **Endgame databases, KQKR**

Mailbox Representation

```
#define OFF   -1
#define EMPTY  0
#define WP     1
#define WN     2

int Board [120];
```




```
if (Board [From + 11] == EM) {
  Move.From = From; Move.To = From + 11;
}
```

Bitboard Representation

```
uint64 wp, wn, wb,wr,wq,wk,bp,bn,bb,br,bq,bk,all;

...

ToPattern = KingMoves [FirstOne (wk)] & AllBlack;
Move.From = FirstOne(wk);
Move.To = FirstOne (ToPattern);

...

ToPattern = (wp >> 7) & DiagLeftMask & AllBlack;
```

Weaknesses

- Limited search depth
- Difficult to perform reliable selectivity
- Strategical evaluation
- Today's chess programs are tactical monsters, but weak in planning and strategy

Other Methods

- Specific methods (Nim)
- Complete databases
  - Colin Vout's car game
  - Mintman
  - The L Game
- Expert systems
  - No success so far...

Nim

- Three piles of matches
- Take n matches from one of the piles
- The one who takes the last match(es) wins
- Explicit methods to win
- Split numbers into groups of 2^n, the so called Nim sum
- Make an even Nim sum and your are certain to win the game

Nim

```
5, 3, 1        4 + 1, 2 + 1, 1        2, 3, 1
0, 3, 1        0, 2 + 1, 1            0, 1, 1
0, 1, 0        0, 1, 0                0, 0, 0
```

The Nimotron, Condon et al., 1940

Algorithmic Solutions

- KPK
  - Small rule base can play well
- KRK
  - Leonardo Torres y Quevedo, 1890

Inlämningsuppgift I

- Ett program för spelet Kalaha
- Baserat på sökning med standardtekniker
  - Representation?
  - Värdering
  - Alfabeta
  - Iterativ fördjupning
- Diskussion på nästa föreläsning