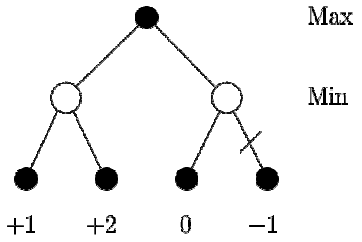
Multi-ProbCut Search

Michael Buro
Department of Computing Science
University of Alberta

mburo@cs.ualberta.ca
http://www.cs.ualberta.ca/~mburo

Background: Look-Ahead Search

- Decision based on evaluation of future states
- Compensates for evaluation function errors
- Problem: Combinatorial explosion + majority of actions are unproductive
- E.g. Alpha-Beta game-tree search

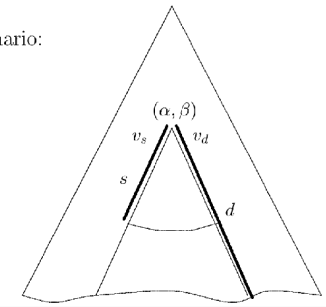


- #nodes is $\Theta(\sqrt{b^d})$
- small number of good moves

Selective Search: ProbCut

- Goal: save useless work by pruning probably irrelevant subtrees.
- Search results are highly correlated!
- Current subtree has no effect $\iff v_d \le \alpha \ \vee \ v_d \ge \beta$
- Idea: estimate deep search result and relax condition $\hat{v}_d \le \alpha - t \ \vee \ \hat{v}_d \ge \beta + t$

Search scenario:

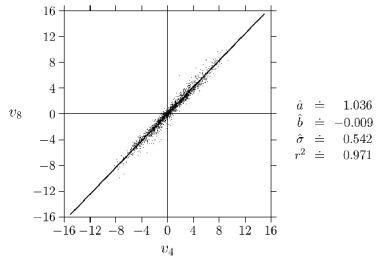


ProbCut Opinion-Change Model

$$v_d = a \cdot v_s + b + e \quad a, b \in \mathcal{R} \quad e \sim \mathcal{N}(0, \sigma^2)$$

$$\rightsquigarrow \quad \hat{v}_d := a \cdot v_s + b$$

Prune subtree iff $a \cdot v_s \notin (\alpha - b - T \cdot \sigma \ , \ \beta - b + T \cdot \sigma)$



Parameter Optimization

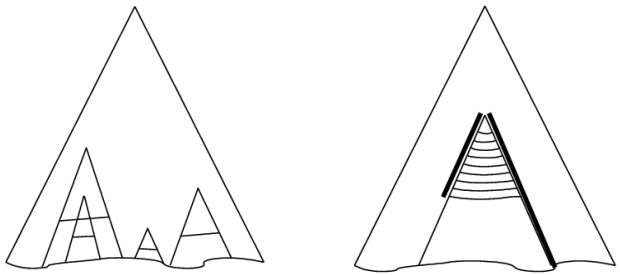
- Given: search depths s and d
- To be determined:
  - Model parameters: a, b, σ :
    - Generate set of representative positions
    - Compute depth s, t search results
    - Compute maximum likelihood estimates
  - Cut-confidence level T :
    - Play tournaments against a full-width player using different values, e.g. T=0.5(0.1)2.5
    - Pick T with the best result

ProbCut Results

- Application: Othello
- s=4, d=8
- T ~ 1.5 found to be best
  (other peak at ~1.0 almost as good, bimodal!)
- 4-8-ProbCut won 74% of the points in a 70-game tournament against the full-width player

Improvements: Multi-ProbCut (MPC)

- Game stage dependent cut thresholds
- a (s, d) pair for each search height d
- Multiple check searches of increasing depth



MPC Results

- Cut pairs up to (s=5, d=17)
- Two cut thresholds
  - opening T=1.0
  - middle game T=1.5
- Wins 72% of the points against 4-8-ProbCut in a 140-game tournament
- Wins 80% against the full-width player
- Equally strong if MPC only uses 4% of the full-width player's thinking time

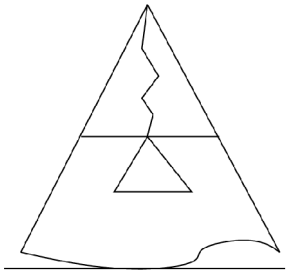
Selective Endgame Search

- Fast endgame search is crucial in games where the remaining number of moves can be accurately estimated (e.g. Othello, Ataxx, Hex, and Domineering)
- Programs play perfectly in positions close to the terminal horizon.
- How to improve decisions before the first game position can be solved?

MPC tailored to endgames +
increasing cut-confidence!

EndCut

- Start with iterative deepening MPC search
- Switch to iterative widening EndCut (increase confidence):
  - At specific heights perform shallow searches to estimate the final game result
  - Prune subtree if value falls outside enlarged window
  - Otherwise, solve position
- Refinements: 0-window search, no cuts on PV.



EndCut in Othello

- Selective endgame search very important. All top programs use it.
- "Selectively solves" most positions 4+ plies earlier than traditional solvers with high confidence.
- Adjustable anytime algorithm: finds winning moves quicker, doesn't get stuck in huge proof trees.
- Logistello switches to EndCut when #discs + MPC depth reaches 61. Usually at 37-38 discs, i.e. 27-26 plies to go.

Conclusion & Outlook

- MPC prunes probably irrelevant subtrees. Saves time for investigating relevant lines.
- MPC is game independent and its parameters are easy to tune.
- MPC is effective in Othello, amazons, shogi, and checkers. Ataxx, chess, hex, or go, anyone?
- Can the MPC idea be applied to single-agent search?
- Needed: a more sophisticated opinion-change model. Use positional features to model variance?