7. Windows

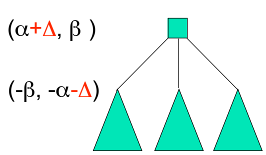
Jonathan Schaeffer
jonathan@cs.ualberta.ca
www.cs.ualberta.ca/~jonathan

Playing with Search Windows

- There are many ways that the search window can be altered
- Use this to:
  - Improve search efficiency
  - Answer questions
  - Low overhead exploratory searches

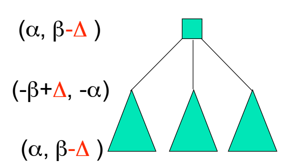
Improving α

Improve α means that the children have a smaller threshold for achieving a cutoff



Improving β

Improve β means that the grand children have a smaller threshold for achieving a cutoff



Aspiration Search

- Normally search the root using (-∞,+∞)
- What if you have a good idea of where the root value really is?
- Assume you believe the root value is close to V, plus or minus Δ?
- Search the root using (V-Δ, V+Δ)
- But…

Aspiration Search

- If result of the search is in (V-Δ, V+Δ), then everything is OK and you have obtained the root value with less search
- What if the value, *v*, is <= V-Δ?
  - Fail low
  - Re-search using window (-∞, *v*)
- What if the value, *v*, is >= V+Δ?
  - Fail high
  - Re-search using window (*v*, +∞)

Aspiration Search

- What is a good guess for the value of a search?
  - Use the value returned for the depth *d* search as the guess for the *d+1* search
- How big should Δ be?
  - Smaller Δ means less search
  - Use experimentally
- Note: aspiration search can build trees smaller than the minimal tree -- why?

Aspiration Search

```
guess = 0;
for( depth = 1; TimeAvailable(); depth++ ) {
    alpha = guess - Δ; beta = guess + Δ;
    /* Search moves: maximize score */
    if( score >= beta ) {
        alpha = score; beta = ∞;
        /* Search with new window */
    } else if( score <= alpha ) {
        alpha = -∞; beta = score;
        /* Search with new window */
    }
    guess = score;
}
```

Narrow Windows

- The idea of narrowing a window can be taken to the extreme!
- What are the semantics of (v,v+1)?
- Minimal or null window
- Answers a Boolean question:
  - Is the value <= v or is it >v

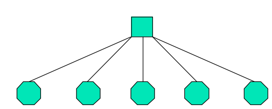
CUT Nodes Revisited

- We know that at CUT nodes, with high probability the best move is being searched early
- That implies that the rest of the moves are inferior to the best
- Can we reduce search effort by exploiting this observation?

NegaScout [1] and PVS [2]

- Search best move with the full window
- Search the remaining moves with a minimal window attempting to prove them inferior to the best move
  - Could be wrong and may need to re-search
  - Re-search may not be expensive because of saved TT results
  - With good move ordering, number of re-searches is usually very small

Example



Window	(0,9)	(2,3)	(2,3)	(2,3)	(7,8)
Result	2	2	0	3	5
				(3,9)	
				7	

NegaScout

```
score = -AlphaBeta( Successor( 1 ), -beta, -alpha, d );
if( score < beta ) {
   for( child = 2; child <= NumSuccessors( s ); child++ ) {
      lbound = MAX( score, alpha ); ubound = lbound + 1;
      result = -AlphaBeta( Successor( child ), -ubound, -lbound );
      if( result >= ubound && result < beta ) {
          result = -AlphaBeta( Successor( child ), -beta, -result );
      }
      if( result > score ) score = result;
      if( result >= beta ) break;
   }
}
return( score );
```


Taking MWs to the Extreme

- Pearl first proposed the idea of doing minimal window searches (Scout) [3]
- Used them to answer a Boolean question about the search tree
- Can we couch the entire alpha-beta search as a series of Boolean questions?

Scout Searches

- To determine the value of root do a series of minimal window searches to narrow in on the value.
- Could do a divide an conquer approach, using each search to cut the possible alpha-beta range in half

Scouting

- Assume value lies in range [-100,100]
    - Possible values are -100..100
  - Search with window [0,1] gives result of 1
    - Possible values are 1..100
  - Search with window [50,51] gives result 48
    - Possible values are [1..48]
  - Search with window [24,25] gives result 24
    - Possible values are [1..24]
  - And so on…

MTD(f)

- Divide-and-conquer is too slow.
- We have a good idea where the true value is… start from there.
- Use the value of the previous iteration to see the starting minimal window.

9/9/02 17

MTD(f) [4,5]

```
lbound = -∞; ubound = +∞;
score = ResultOfPreviousIteration();
repeat {
   if( score == lbound )      window = score;
   else                       window = score - 1;
   score = AlphaBeta( s, window, window + 1, d );
   if( score < window )       ubound = score;
   else                       lbound = score;
} until ( lbound >= ubound );
```

9/9/02 18

MTD(f)

- Note that all searches go down (except the last one), or all go up (except the last one)
- This only makes sense with a TT so that results of previous searches can be reused
- For chess an average of 3-5 iterations were needed to converge
- Node reductions of 5-15%

9/9/02 19

MTD(f)

- By changing the initial guess and the choice of bounds, SSS* and DUAL* can be derived.
- There are subtle points that may result in MTD(f) not converging
  - Bugs in your algorithm
  - Search depth changes
  - TT side effects

9/9/02 20

References

[1] Alexander Reinefeld. “An Improvement to the Scout Tree Search Algorithm”, ICGA Journal, vol. 6, no.4, pp. 4-14, 1983. See also [4].

[2] Murray Campbell and Tony Marsland. “A Comparison of Minimax Tree Search Algorithms”, *Artificial Intelligence*, vol. 20, pp. 347-367, 1983.

[3] Judea Pearl. “Scout: A Simple Game-Searching Algorithm with Proven Optimal Properties”, AAAI National Conference, 1980.

[4] www.cs.vu.nl/~aske/mtdf.html

[5] Aske Plaat, Jonathan Schaeffer, Wim Pijls, and Arie de Bruin. “Best-first Fixed-depth Minimax Algorithms”, *Artificial Intelligence*, vol. 87, no. 1-2, pp/ 1-38, 1996.