Pruning properties

- Techniques covered so far only prune if the sum of 2 players scores adds up to maxsum
- Consider Hearts as a maximization game:
  - Score is points taken by other players
  - maxp = 26; maxsum = 52; minp = 0; minsum = 52
  - The only way 2 players scores sum to 52 is if they both get 26; but then we just do immediate pruning!
  - Shallow pruning never occurs!
  - Depends on properties of game, not order of tree

Multi-player games have more than 2 players

- All pruning that we’ve seen so far only involves bounds for 2 players
  - It is highly unlikely that 2 players have their bounds add to maxsum
  - Effective pruning techniques involve bounds from all players

Last branch pruning (full)

![Diagram](image)
Speculative Pruning

- When $b$ gets large, the gains of speculative pruning will be minimal; can we do better
- If we analyze where deep pruning failed, (depth 3 tree) there were two conditions:
  - Unseen leaf at P2 node has better score for both P2 and P1
  - Unseen leaf at P3 node is better for P3 and P2
- This corresponds to having the move ordering wrong
- Solution: prune, and then verify these two conditions

Speculative Pruning Analysis

- Speculative pruning depends on the order of nodes in the tree, not the values in the game
- Asymptotically still must provide a strategy for each player, so bounded by $b^{(n-1)/nd}$
  - Can verify this lower-bound via recurrence equations

Speculative Pruning Example

- Speculative pruning depends on the order of nodes in the tree, not the values in the game
- Asymptotically still must provide a strategy for each player, so bounded by $b^{(n-1)/nd}$
- Can verify this lower-bound via recurrence equations

Generalized Pruning

- In practice our evaluation functions are points in 3d space
  - Each bound drops a plane in 3d space
    - If a point is guaranteed to be outside the current 3d bounded space, we have the potential to prune
  - Separates the pruning decision from the algorithm that implements correct pruning
Generalized Pruning

- Suppose that the evaluation function is a function of the underlying outcomes of the game
  - eg outcomes far fewer than strategies for play
  - Then, we can iterate over outcomes to see if pruning is possible

Example of complex non-zero sum game

<table>
<thead>
<tr>
<th>Possible Tricks</th>
<th>Game Score</th>
<th>Evaluation Fcn</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0, 0, 3)</td>
<td>(-20, ...)</td>
<td>(0, ...)</td>
</tr>
<tr>
<td>(0, 1, 2)</td>
<td>(-20, ...)</td>
<td>(0, ...)</td>
</tr>
<tr>
<td>(0, 2, 1)</td>
<td>(-20, ...)</td>
<td>(0, ...)</td>
</tr>
<tr>
<td>(0, 3, 0)</td>
<td>(-20, ...)</td>
<td>(0, ...)</td>
</tr>
<tr>
<td>(1, 0, 2)</td>
<td>(-20, ...)</td>
<td>(0, ...)</td>
</tr>
<tr>
<td>(1, 1, 1)</td>
<td>(-20, ...)</td>
<td>(0, ...)</td>
</tr>
<tr>
<td>(1, 2, 0)</td>
<td>(-20, ...)</td>
<td>(0, ...)</td>
</tr>
<tr>
<td>(2, 0, 1)</td>
<td>(20, ...)</td>
<td>(2, ...)</td>
</tr>
<tr>
<td>(2, 1, 0)</td>
<td>(20, ...)</td>
<td>(2, ...)</td>
</tr>
<tr>
<td>(3, 0, 0)</td>
<td>(21, ...)</td>
<td>(1, ...)</td>
</tr>
</tbody>
</table>

Generalized technique

- Generate points for the utility of each game outcome.
  0. Calculate current bound vector and heuristics.
  1. Check hash table for entry.
  2. Remove values that are inconsistent with heuristic.
  3. Check boundary of space.
  4. Add result to hash table.
### Other techniques?

- History Heuristic / move ordering
  - No problem...but...may change equilibrium strategy/payoff
- Transposition tables
  - No problem...but...may change equilibrium strategy/payoff

### Opponent modeling

- What if I don’t have the right opponent model?
  - Tie-breaking (shared eval function)
    - Break ties against player at root of tree
    - Will work poorly if there are lots of ties
  - Tie-breaking (bad evaluation function)
    - Need to compare values with epsilon difference
    - Or, simplify opponents’ model to represent uncertainty of their actions