Chapter 23
Planning in the Game of Bridge

Dana S. Nau
University of Maryland
Fall 2009
Computer Programs for Games of Strategy

Connect Four: solved
Go-Moku: solved
Qubic: solved
Nine Men’s Morris: solved
Checkers: solved
Othello: better than humans
Backgammon: better than all but about 10 humans
Chess: competitive with the best humans

Bridge: about as good as mid-level humans
Computer Programs for Games of Strategy

- Fundamental technique: the minimax algorithm

\[
\text{minimax}(u) = \begin{cases} 
\max\{\text{minimax}(v) : v \text{ is a child of } u\} & \text{if it’s Max’s move at } u \\
\min\{\text{minimax}(v) : v \text{ is a child of } u\} & \text{if it’s Min’s move at } u 
\end{cases}
\]

- Largely “brute force”
- Can prune off portions of the tree
  - cutoff depth & static evaluation function
  - alpha-beta pruning
  - transposition tables
  - …
- But even then, it still examines thousands of game positions
- For bridge, this has some problems …
How Bridge Works

- Four players; 52 playing cards dealt equally among them
- Bidding to determine the trump suit
  - Declarer: whoever makes highest bid
  - Dummy: declarer’s partner
- The basic unit of play is the trick
  - One player leads; the others must follow suit if possible
  - Trick won by highest card of the suit led, unless someone plays a trump
  - Keep playing tricks until all cards have been played
- Scoring based on how many tricks were bid and how many were taken
Game Tree Search in Bridge

- Bridge is an *imperfect information* game
  - Don’t know what cards the others have (except the dummy)
  - Many possible card distributions, so many possible moves
- If we encode the additional moves as additional branches in the game tree, this increases the branching factor $b$
- Number of nodes is exponential in $b$
  - worst case: about $6 \times 10^{44}$ leaf nodes
  - average case: about $10^{24}$ leaf nodes
- A chess game may take several hours
- A bridge game takes about 1.5 minutes

Not enough time to search the game tree
Reducing the Size of the Game Tree

- One approach: HTN planning
  - Bridge is a game of planning
  - The declarer plans how to play the hand
  - The plan combines various strategies (ruffing, finessing, etc.)
  - If a move doesn’t fit into a sensible strategy, it probably doesn’t need to be considered
- Write a planning procedure procedure similar to TFD (see Chapter 11)
  - Modified to generate game trees instead of just paths
  - Describe standard bridge strategies as collections of methods
  - Use HTN decomposition to generate a game tree in which each move corresponds to a different strategy, not a different card

<table>
<thead>
<tr>
<th></th>
<th>Brute-force search</th>
<th>HTN-generated trees</th>
</tr>
</thead>
<tbody>
<tr>
<td>Worst case</td>
<td>$\approx 6 \times 10^{44}$ leaf nodes</td>
<td>$\approx 305,000$ leaf nodes</td>
</tr>
<tr>
<td>Average case</td>
<td>$\approx 10^{24}$ leaf nodes</td>
<td>$\approx 26,000$ leaf nodes</td>
</tr>
</tbody>
</table>
Methods for Finessing

- LeadLow(P₁; S)
- PlayCard(P₁; S, R₁)
- Finesse(P₁; S)
  - StandardFinesse(P₂; S)
  - FinesseTwo(P₂; S)
    - EasyFinesse(P₂; S)
    - StandardFinesseTwo(P₂; S)
    - StandardFinesseThree(P₃; S)
    - FinesseFour(P₄; S)
      - PlayCard(P₂; S, R₂)
      - PlayCard(P₃; S, R₃)
      - PlayCard(P₄; S, R₄)
      - PlayCard(P₄; S, R₄')

- 1st opponent
- declarer
- 2nd opponent
- dummy
- task
- method
- time ordering

possible moves by 1st opponent
Instantiating the Methods

**Task**
- Finesse($P_1; S$)

**Method**
- LeadLow($P_1; S$)
- FinesseTwo($P_2; S$)
- PlayCard($P_1; S, R_1$)
- EasyFinesse($P_2; S$)
- StandardFinesse($P_2; S$)
- BustedFinesse($P_2; S$)
- PlayCard($P_2; S, R_2$)
- StandardFinesseTwo($P_2; S$)
- StandardFinesseThree($P_3; S$)
- FinesseFour($P_4; S$)
- PlayCard($P_3; S, R_3$)
- PlayCard($P_4; S, R_4$)
- PlayCard($P_4; S, R_4'$)

**Time Ordering**
- Us: East declarer, West dummy
- Opponents: defenders, South & North
- Contract: East – 3NT
- On lead: West at trick 3

**Card Setup**
- East: ♠KJ74
- West: ♠A2
- Out: ♠QT98653

**Players**
- Us: East declarer, West dummy
- Opponents: defenders, South & North

**Contract**
- East – 3NT

**On Lead**
- West at trick 3

**Card Setup**
- East: ♠KJ74
- West: ♠A2
- Out: ♠QT98653
Generating Part of a Game Tree

Finesse\((P_1; S)\)

LeadLow\((P_1; S)\)

PlayCard\((P_1; S, R_1)\)

EasyFinesse\((P_2; S)\)

StandardFinesse\((P_2; S)\)

BustedFinesse\((P_2; S)\)

FinesseTwo\((P_2; S)\)

\(\ldots\)

StandardFinesseTwo\((P_2; S)\)

StandardFinesseThree\((P_3; S)\)

FinesseFour\((P_4; S)\)

\(\ldots\)

PlayCard\((P_2; S, R_2)\)

PlayCard\((P_3; S, R_3)\)

PlayCard\((P_4; S, R_4)\)

PlayCard\((P_4; S, R_4')\)

The red boxes are the leaf nodes

North—\(\spadesuit 3\)

West—\(\spadesuit 2\)

(North—\(\spadesuit Q\))

East—\(\heartsuit J\)

South—\(\spadesuit 5\)

South—\(\spadesuit Q\)
Game Tree Generated using the Methods

... later stratagems ...

FINESSE

+600

W♠2

N♣2

N♣Q

0.9854

+265

E♠J

E♠K

0.0078

0.0078

0.5

+265

+630

S♣Q

S♠3

0.5

0.5

S♠5

S♠3

+600

+600

+600

CASH OUT

W♠A

N♣3

E♣4

S♣5

+600

+600

+600

-100

+630

+630

+630

+600

+600

+600
Implementation

- Stephen J. Smith, then a PhD student at U. of Maryland
  - Wrote a procedure to plan declarer play
- Incorporated it into Bridge Baron, an existing commercial product
  - This significantly improved Bridge Baron’s declarer play
  - Won the 1997 world championship of computer bridge
- Since then:
  - Stephen Smith is now Great Game Products’ lead programmer
  - He has made many improvements to Bridge Baron
    - Proprietary, I don’t know what they are
  - Bridge Baron was a finalist in the 2003 and 2004 computer bridge championships
    - I haven’t kept track since then
Other Approaches

- Monte Carlo simulation:
  - Generate many random hypotheses for how the cards might be distributed
  - Generate and search the game trees
    » Average the results

- This can divide the size of the game tree by as much as $5.2 \times 10^6$
  » $(6 \times 10^{44})/(5.2 \times 10^6) = 1.1 \times 10^{38}$
    • still quite large
  » Thus this method by itself is not enough
Other Approaches (continued)

- AJS hashing - Applegate, Jacobson, and Sleator, 1991
  - Modified version of transposition tables
    - Each hash-table entry represents a set of positions that are considered to be equivalent
    - Example: suppose we have ♠AQ532
      - View the three small cards as equivalent: ♠Aqxxx
  - Before searching, first look for a hash-table entry
    - Reduces the branching factor of the game tree
    - Value calculated for one branch will be stored in the table and used as the value for similar branches
- GIB (1998-99 computer bridge champion) used a combination of Monte Carlo simulation and AJS hashing
- Several current bridge programs do something similar
Top contenders in computer bridge championships, 1997–2004

<table>
<thead>
<tr>
<th>Year</th>
<th>#1</th>
<th>#2</th>
<th>#3</th>
<th>#4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1997</td>
<td>Bridge Baron</td>
<td>Q-Plus</td>
<td>Micro Bridge</td>
<td>Meadowlark</td>
</tr>
<tr>
<td>1998</td>
<td>GIB</td>
<td>Q-Plus</td>
<td>Micro Bridge</td>
<td>Bridge Baron</td>
</tr>
<tr>
<td>1999</td>
<td>GIB</td>
<td>WBridge5</td>
<td>Micro Bridge</td>
<td>Bridge Buff</td>
</tr>
<tr>
<td>2000</td>
<td>Meadowlark</td>
<td>Q-Plus</td>
<td>Jack</td>
<td>WBridge5</td>
</tr>
<tr>
<td>2001</td>
<td>Jack</td>
<td>Micro Bridge</td>
<td>WBridge5</td>
<td>Q-Plus</td>
</tr>
<tr>
<td>2002</td>
<td>Jack</td>
<td>Wbridge5</td>
<td>Micro Bridge</td>
<td>?</td>
</tr>
<tr>
<td>2003</td>
<td>Jack</td>
<td>Bridge Baron</td>
<td>WBridge5</td>
<td>Micro Bridge</td>
</tr>
<tr>
<td>2004</td>
<td>Jack</td>
<td>Bridge Baron</td>
<td>WBridge5</td>
<td>Micro Bridge</td>
</tr>
</tbody>
</table>

I haven’t kept track since 2004

For more information see http://www.jackbridge.com/ewkprt.htm