A Doppelkopf Player Based on UCT

Silvan Sievers     Malte Helmert

University of Basel
Basel, Switzerland

September 25, 2015
Doppelkopf: card game with similarities to skat, but larger state space

Unique feature: parties usually only revealed during card play!

UCT: state-of-the-art algorithm for many applications of acting under uncertainty
Outline

1. Doppelkopf
2. The UCT Algorithm
3. The Card Assignment Problem
4. Experiments
Game Rules

- 4 players, 2 changing parties
- 48 cards: double deck from nines to aces
- Total of 240 card points
- Goal: collect 121 card points
Game Rules

- 4 players, 2 changing parties
- 48 cards: double deck from nines to aces
- Total of 240 card points
- Goal: collect 121 card points
- Normal game:
  - Trump suit: ♥10, queens, jacks, remaining ♦
  - Off suits: remaining ♠, ♦, ♥
  - Two parties: re and kontra (players with and without ♠Q)
Tricks: Example
Tricks: Example
Tricks: Example
Tricks: Example
Tricks: Example
Game Rules

Announcements:

- All reveal party of the announcing player
- All increase the game value
- Some increase card points required for winning
Game Rules

- **Announcements:**
  - All **reveal party** of the announcing player
  - All increase the game value
  - Some increase **card points required for winning**

- **Game evaluation:** **score points**
  - +1 for winning
  - +2/+1 for different announcements
  - +1 for every 30 card points achieved above the threshold required for winning
  - +1 through winning special tricks
Outline

1. Doppelkopf
2. The UCT Algorithm
3. The Card Assignment Problem
4. Experiments
Why UCT?

- Goal: determine the “best” move
- Problem: computing all possibilities infeasible
- UCT (Kocsis and Szepesvári 2006):
  - Monte Carlo tree search algorithm based on sampling
  - State of the art for scenarios with uncertainty
The UCT Algorithm

- Repeatedly perform rollouts of the game:
  - Assume a fixed card assignment
  - Traverse the game tree, choosing successors by balancing exploration and exploitation
  - From leaf nodes on: Monte Carlo simulation
  - End at terminal game states: compute rewards
  - Propagate back information
Variations

- Varying the number of card assignment used:
  - **Single-UCT**: regular UCT, each rollout with a different card assignment
  - **Ensemble-UCT**: several UCT computations, each with a different fixed card assignment
Outline

1. Doppelkopf
2. The UCT Algorithm
3. The Card Assignment Problem
4. Experiments
Problem Statement

The Card Assignment Problem (CAP)

- Given a game state and a player to move:
  - Assign all remaining cards to all other players
  - Respect all available information from the game history
The Card Assignment Problem (CAP)

- Given a game state and a player to move:
  - Assign all remaining cards to all other players
  - Respect all available information from the game history
- Goal for an unbiased UCT player: compute solutions to the CAP uniformly at random
- Requirement: computing the number of solutions of the CAP, i.e. solving #CAP
- Complexity of #CAP: #P-complete
The Card Assignment Algorithm

While there are cards left to be assigned:
The Card Assignment Algorithm

While there are cards left to be assigned:

If a card can be assigned to exactly one player:
    Assign that card to that player
The Card Assignment Algorithm

While there are cards left to be assigned:
   If a card can be assigned to exactly one player:
       Assign that card to that player
   If a player requires as many cards as he can have:
       Assign those cards to that player
**The Card Assignment Algorithm**

While there are cards left to be assigned:

If a card can be assigned to exactly one player:
    Assign that card to that player

If a player requires as many cards as he can have:
    Assign those cards to that player

If a player requires a ♣Q:
    Assign a ♣Q to that player
The Card Assignment Algorithm

While there are cards left to be assigned:

- If a card can be assigned to exactly one player:
  - Assign that card to that player
- If a player requires as many cards as he can have:
  - Assign those cards to that player
- If a player requires a ♣Q:
  - Assign a ♣Q to that player
- Otherwise:
  - Assign a random card to a random player
Properties of the Algorithm

- Only generates **consistent** card assignments
- Terminates after at most as many iterations as cards need to be assigned to players
- Solutions **not generated uniformly at random:**
  - Number of card slots of players not considered
  - Assignment of ♣Q prioritized over other cards
  - Number of possible assignments not considered
Outline

1. Doppelkopf
2. The UCT Algorithm
3. The Card Assignment Problem
4. Experiments
Setup

- Two UCT players against two random players
- 1000 games with random card deals
- Repeat every game in every possible permutation
- Total of 10000 rollouts for every decision
- Results: average score points per game with 95% confidence interval
### Ensemble-UCT Configurations

- **X/Y**: number of single UCT computations/rollouts

<table>
<thead>
<tr>
<th>Ensemble-UCT Configuration</th>
<th>Quality Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>ensemble-UCT (5/2000)</td>
<td>1.67 ± 0.12</td>
</tr>
<tr>
<td>ensemble-UCT (10/1000)</td>
<td>1.83 ± 0.11</td>
</tr>
<tr>
<td>Random</td>
<td>(−1.75 ± 0.05)</td>
</tr>
<tr>
<td>ensemble-UCT (10/1000)</td>
<td>2.10 ± 0.11</td>
</tr>
<tr>
<td>ensemble-UCT (20/500)</td>
<td>1.70 ± 0.10</td>
</tr>
<tr>
<td>Random</td>
<td>(−1.90 ± 0.05)</td>
</tr>
</tbody>
</table>
### Ensemble-UCT Configurations

- X/Y: number of single UCT computations/rollouts

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Value (Mean ± Standard Deviation)</th>
<th>Random Value (Mean ± Standard Deviation)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ensemble-UCT (5/2000)</td>
<td>1.67 ± 0.12</td>
<td>1.83 ± 0.11 (−1.75 ± 0.05)</td>
</tr>
<tr>
<td>ensemble-UCT (10/1000)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ensemble-UCT (10/1000)</td>
<td>2.10 ± 0.11</td>
<td>1.70 ± 0.10 (−1.90 ± 0.05)</td>
</tr>
<tr>
<td>ensemble-UCT (20/500)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

→ **trade-off** between the number of different card assignments and the quality of the computation per card assignment
## Influence of Announcement Making

<table>
<thead>
<tr>
<th></th>
<th>ensemble-UCT</th>
<th>random</th>
</tr>
</thead>
<tbody>
<tr>
<td>announcing</td>
<td>1.70 ± 0.07</td>
<td>0.48 ± 0.06</td>
</tr>
<tr>
<td>no announcing</td>
<td>0.79 ± 0.05</td>
<td>0.19 ± 0.05</td>
</tr>
<tr>
<td></td>
<td>(−1.25 ± 0.04)</td>
<td>(−0.33 ± 0.04)</td>
</tr>
</tbody>
</table>

Making announcements is crucial for performance.


## Influence of Announcement Making

<table>
<thead>
<tr>
<th></th>
<th>ensemble-UCT</th>
<th>random</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>announcing</strong></td>
<td>1.70 ± 0.07</td>
<td>0.79 ± 0.05 (−1.25 ± 0.04)</td>
</tr>
<tr>
<td><strong>no announcing</strong></td>
<td>0.48 ± 0.06</td>
<td>0.19 ± 0.05 (−0.33 ± 0.04)</td>
</tr>
</tbody>
</table>

→ making announcements **crucial** for performance
<table>
<thead>
<tr>
<th>ensemble-UCT</th>
<th>single-UCT</th>
<th>random</th>
</tr>
</thead>
<tbody>
<tr>
<td>$4.52 \pm 0.11$</td>
<td>$-1.25 \pm 0.08$</td>
<td>$(-1.63 \pm 0.05)$</td>
</tr>
</tbody>
</table>
## Ensemble-UCT versus Single-UCT

<table>
<thead>
<tr>
<th></th>
<th>ensemble-UCT</th>
<th>single-UCT</th>
<th>random</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$4.52 \pm 0.11$</td>
<td>$-1.25 \pm 0.08$</td>
<td>$-1.63 \pm 0.05$</td>
</tr>
</tbody>
</table>

→ using **few, but fixed card assignments** better than using many card assignments
Analysis of UCT Players

- Two sets of 24 games human vs. ensemble-UCT:
  - Too many solos (works well against random players)
  - Always makes announcements when playing solo, but rarely in normal games
  - The fewer options remaining, the stronger the game play
Possible Improvements

- Separate hand evaluation algorithm
- Analyze and reduce bias of card assignment algorithm
- Domain specific knowledge for simulation phase of rollouts
- Drop assumption that opposing players behave like UCT players
- Reuse information from decisions at previous game states
Conclusion

- Doppelkopf as a benchmark problem
- Baseline UCT players
- Card assignment algorithm
- Ensemble-UCT for more stable UCT computations
The End

Thank you!