A Decision Heuristic for Monte Carlo Tree Search Doppelkopf Agents

by Alexander Dockhorn, Christoph Doell, Matthias Hewelt, and Rudolf Kruse

Institute for Intelligent Cooperating Systems
Department for Computer Science, Otto von Guericke University Magdeburg
Universitaetsplatz 2, 39106 Magdeburg, Germany

Email: {alexander.dockhorn, christoph.doell, rudolf.kruse}@ovgu.de , matthias.hewelt@st.ovgu.de

Contents

I. Doppelkopf – the Card Game

II. Monte Carlo Tree Search (MCTS)

III. Adapting MCTS to Card Games

IV. Improving the Rollout Policy of MCTS

V. Conclusion, Limitations and Future Work

Doppelkopf – the card game

- Doppelkopf is a trick taking card game
- 4 players play a set of 12 tricks
- A shortened french deck containing 48 cards is used
  - Two instances of 10, Ace, King, Queen, Jack, 9
  - From the four suits clubs (♣), spades (♠), hearts (♥), and diamonds (♦)
- Different game modes are played depending on the initial card distribution
  - Normal game
  - (un-)announced marriage
  - Jack-/Queen-/Ace-/♣-/♠-/♥-/♦-Solo

Rules of a normal game

- In a normal game players holding the ♣ Q form the re-party. In case a player has both ♣ Q, he can either play a solo or a marriage (not discussed here)

- In a normal game all ♦ cards, all jacks, queens, as well as both ♥ tens form the trump suit

trump suit

♡ 10 → ♣ Q → ♠ Q → ♡ Q → ♢ Q →
♣ J → ♠ J → ♡ J → ♢ J → ♢ A → ♢ 10 → ♢ K → ♢ 9

non-trump suit

♣ A → ♣ 10 → ♣ K → ♣ 9   ♠ A → ♠ 10 → ♠ K → ♠ 9

♡ A → ♡ K → ♡ 9

Rules of a normal game

- Card pips are earned through winning tricks.
  - one player starts by playing a card
  - clockwise players need to add a card of the same suit
  - in case, they cannot follow the played suit (because they do not own an appropriate card) they can choose freely
  - the player who plays the highest card wins the trick and starts the next trick

- The re-party wins if it can secure at least 121 points.

- The winning threshold can be shifted through announcements, which also increase the number of points awarded for winning the game.

Doppelkopf – State Space

- When all players were dealt 12 cards, the number of possible games can be approximated by

$$\sum_{i=0}^{48} \prod_{j=0}^{3} \binom{12}{\lfloor (i+j)/4 \rfloor} \approx 2.4 \cdot 10^{13}$$

- Cards of our opponents are unknown. During a single game the player needs to guess, which cards our opponents have:

$$\binom{36}{12} \cdot \binom{24}{12} \cdot \binom{12}{12} \approx 3.4 \cdot 10^{15}$$

Monte Carlo Tree Search (MCTS)

- MCTS is a heuristic search algorithm
- Future game states are evaluated using random simulations
  - Number of wins and loses are used for rating the node
- Converges to minimax search!
- Does not need an explicit game state evaluation function!
- Has been used for a wide range of board games as well as video games
  - Most recent remarkable achievement is AlphaGo

MCTS

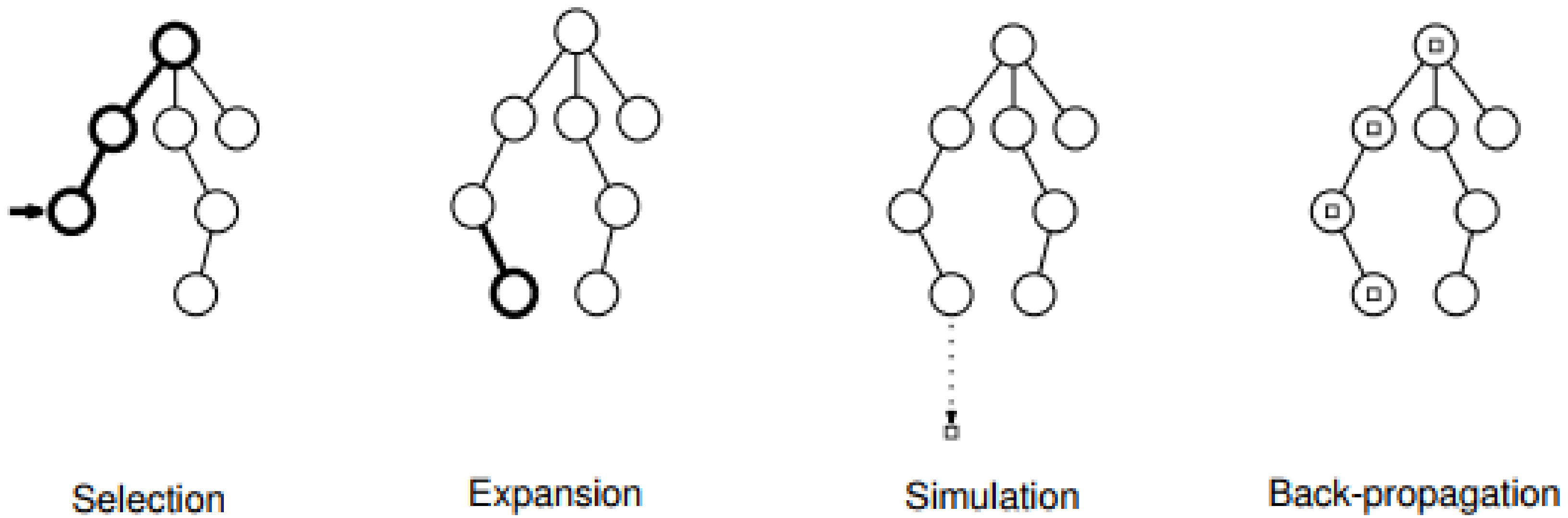


Diagram from: [Santos, A., Santos, P. A., & Melo, F. S. (n.d.). Monte Carlo Tree Search Experiments in Hearthstone.]

Upper Confidence Bounds applied to Trees

- Without any additions much time is lost on unpromising branches of the tree

- Upper confidence bounds represents the tradeoff between exploitation and exploration during the selection step

$$\underbrace{R(s')}_{\text{Exploitation}} + C \underbrace{\sqrt{\frac{log(V(s))}{V(s')}}}_{\text{Exploration}}$$

- R(s‘) = estimated value of node s‘ = average success rate
- V(s‘) = number of visits of node s during the search
- s = parent node of s‘

What is the problem with applying MCTS?

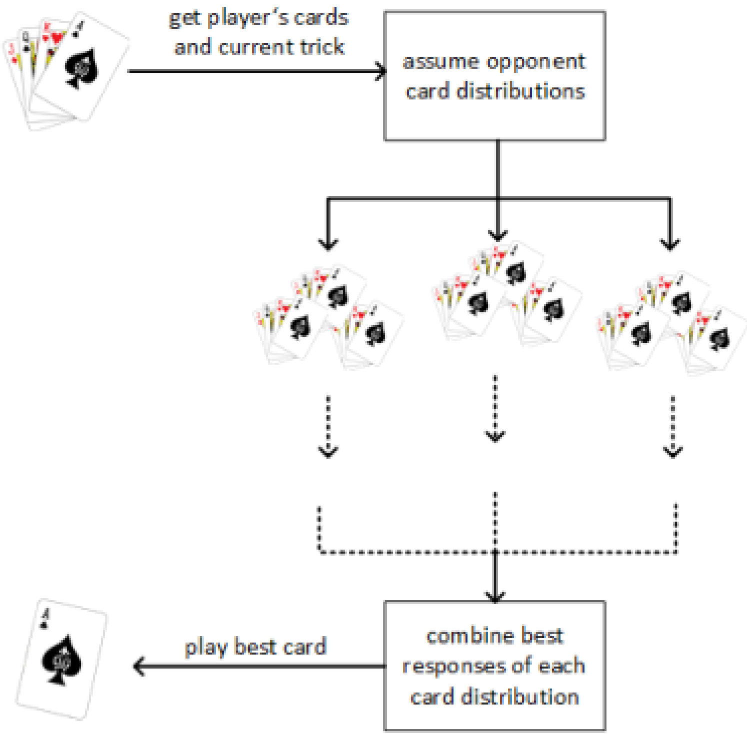
- MCTS needs a reliable forward model

- But we are possibly missing critical information:
  - What will our opponents do?
  - Who is our partner?
  - Which cards does a player hold in his hands?

MCTS – for an unknown card distribution

- Since we do not know the true card distribution, we estimate is as best as possible.
  - If a player could not play cards of a kind, he does not own such a card
  - Previously played cards cannot be distributed
  - Queens are distributed according tc the game mode

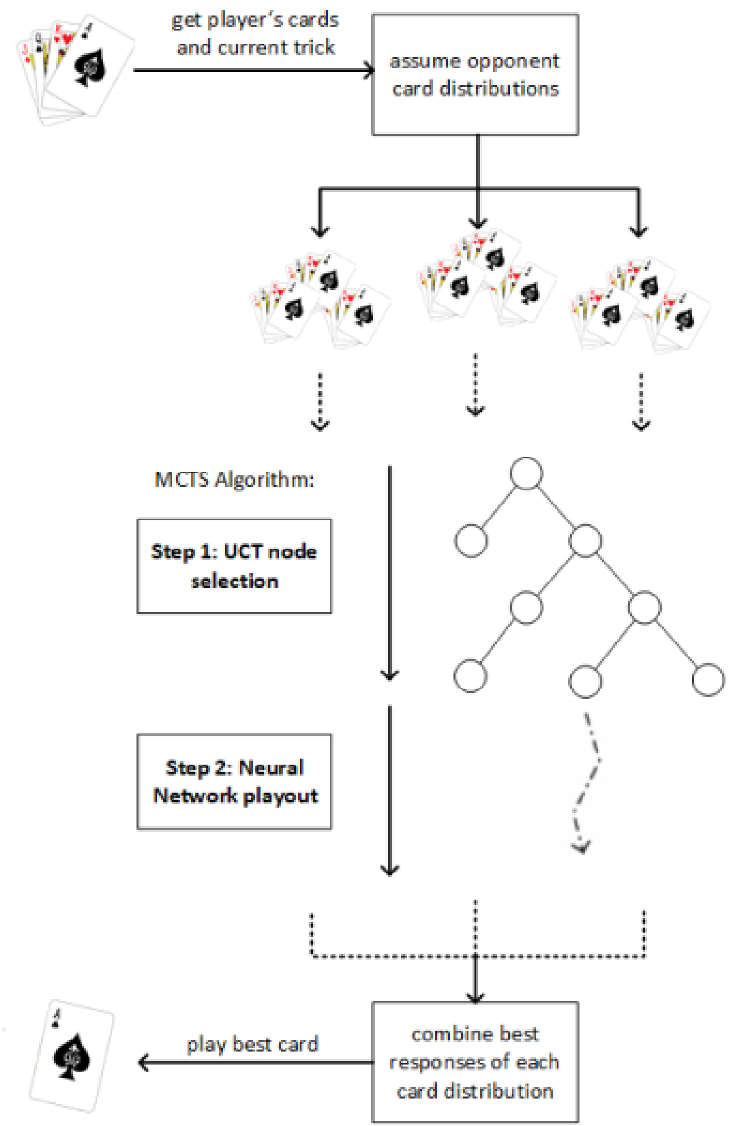
- We create an ensemble of MCTS agents which search for the best card given one card distribution
  - the overall best will be played



(Made by Siever and Helmert)

Learning a rollout policy

- A neural network was trained to predict player moves.
- We used a database of game-histories by human players.
  - (31 448 games, 1 509 504 game states)
- The network was trained to predict the next card by the available information at the moment of the players decision.
- During the rollout the network simulates the moves of the three other players.



The Database

- Data was collected on a German Doppelkopf online-platform.

Game Mode	Total Games	Share
normal game	24 548	$\approx 70.32\%$
announced marriage	6900	$\approx 19.76\%$
jack solo	1263	$\approx 3.62\%$
queen solo	763	$\approx 2.19\%$
ace solo	1086	$\approx 3.11\%$
$\diamondsuit$ solo	88	$\approx 0.25\%$
$\clubsuit$ solo	85	$\approx 0.24\%$
$\spadesuit$ solo	85	$\approx 0.24\%$
unannounced marriage	51	$\approx 0.15\%$
$\heartsuit$ solo	43	$\approx 0.12\%$

Coding the current state of the game

- The following information was encoded
  a) the currently played game mode
  b) the current position in the trick
  c) cards played during the current trick
  d) history of previous tricks
  e) *cards per player
  f) *the party the player belongs to
  g) *the parties of other player

- Using n-hot encoding a total of 406 inputs were neccessary.
- 24 output neurons were used to predict the next card to be played.

* => might not be available to the player

Evaluating the prediction accuracy

- Context-Free (CF): directly compare the highest ranked card predicted by the neural network with the true card in the test sample

- Context-Sensitive (CS): only the highest rated card, which also needs to be playable, is compared to the true outcome

Network Architecture	All Positions		Position 1		Position 4	
	CF	CS	CF	CS	CF	CS
1 hidden layer	0.3294	0.4157	**0.3537**	**0.4986**	**0.3501**	**0.4908**
2 hidden layers	**0.4066**	**0.4767**	0.3498	0.4696	0.2855	0.4469
3 hidden layers	0.4044	0.4701	0.2686	0.4160	0.2346	0.4063
4 hidden layers	0.3479	0.4252	—	—	—	—
5 hidden layers	0.2969	0.3994	—	—	—	—

Optimizing the Model

- Switching to Rectified Linear Units drastically sped up learning time
- New networks achieved much better restults
- Dropout rate assured that we can limit overfitting

Network Architecture	dropout rate = 0 CF	dropout rate = 0.2 CF	dropout rate = 0.5 CF
1 hidden layer	0.5997 ± 0.0148	0.6135 ± 0.0022	0.4965 ± 0.0044
2 hidden layers	$\mathbf{0.7159} \pm 0.0036$	$\mathbf{0.7293} \pm 0.0030$	$\mathbf{0.7194} \pm 0.0023$
6 hidden layers	0.7136 ± 0.0004	0.7186 ± 0.0129	0.7069 ± 0.0095
7 hidden layers	0.7125 ± 0.0026	0.7240 ± 0.0139	0.7176 ± 0.0027

Network Architectures and prediction rates

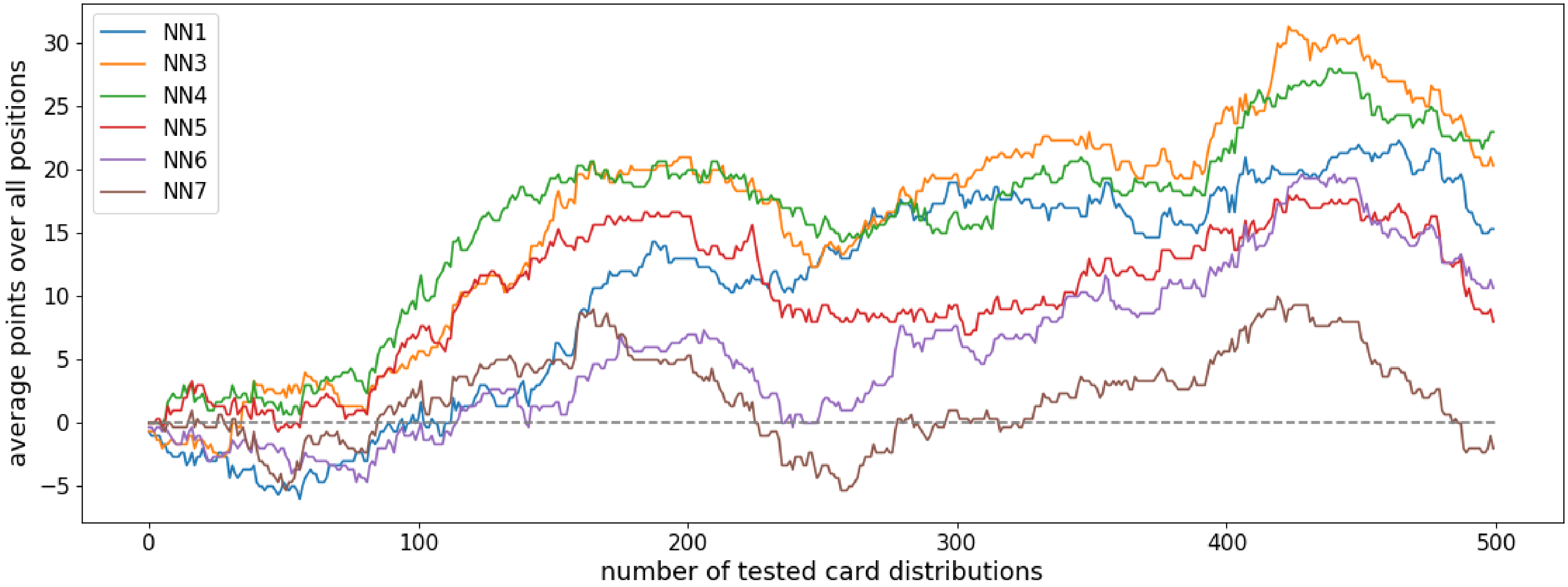
- Multiple network parameters were varied:
  - Depth and width of the network
  - Dropout rates and batch normalization

- Prediction accuracies step-wise increase from Position 1 to Position 4

ID	Network Architecture	All Positions		Position 1		Position 4	
		CF	CS	CF	CS	CF	CS
NN1	406-100-0.2-24	0.6135	0.6410	0.5703	0.5751	0.6806	0.7145
NN2	406-812-0.2-406-24	0.6675	0.7293	0.6014	0.6022	0.7657	0.7698
NN3	406-3248-0.2-406-24	0.6896	0.6952	0.5913	0.5934	0.7450	0.7562
NN4	406-6496-0.2-406-24	0.6910	0.6954	0.5917	0.5939	0.7439	0.7554
NN5	406-1624-0.2-812-406-203-100-50-24	0.7186	0.7206	0.5945	0.5975	0.7577	0.7667
NN6	406-3248-0.2-1624-812-406-203-100-50-24	0.7240	0.7261	0.5900	0.5943	0.7601	0.7710
NN7	406-700-0.2-bn-406-bn-24	**0.7376**	**0.7378**	**0.6044**	**0.6050**	**0.7869**	**0.7887**

Evaluating the strength of the system

- Best performing model in prediction: NN7
  - Now the worst performing network → Overfitting
- Shallow networks with a huge width performed best during simulation



Conclusions

- Neural Networks proved to provide a powerful rollout-policy
- Our system on average beats the previous state of the art by Sievers and Helmert
- Motivated by the success: we are currently in the process in extending our work to other better known card games
  - e.g. **Hearthstone AI Competition** -> Official Announcement in January
  - In case you want to learn more about our future plans just talk to me after the session!

Limitations and Open Research Questions

- Current neural networks are restricted to a snap-shot of currently and previously played cards. The order in which cards were played is lost due to our encoding.
  - Recurrent neural networks could be applied using a time-dependent code
  - Other network structures will be analyzed in the future

- Support more game modes:
  - our current database does not include enough games for certain game types, such as soli and announced marriages

- Making announcements is currently not included in our prediction since they are made in-between the tricks

Thank you for your attention!

Check on Updates on our project at:
http://fuzzy.cs.ovgu.de/wiki/pmwiki.php/Mitarbeiter/Dockhorn
(Download of our project files will be made available soon)

by Alexander Dockhorn, Christoph Doell, Matthias Hewelt and Rudolf Kruse

Institute for Intelligent Cooperating Systems
Department for Computer Science, Otto von Guericke University Magdeburg
Universitaetsplatz 2, 39106 Magdeburg, Germany

Email: {alexander.dockhorn, christoph.doell, rudolf.kruse}@ovgu.de , matthias.hewelt@st.ovgu.de