Artificial Intelligence for the Card Game Durak

Author: Azamat Zarlykov | Supervisor: Adam Dingle, M.Sc | 2023.

Introduction:

Durak is a two-player card game that starts by dealing cards to each player until they have a hand of six cards. The final card of the deck is then placed face up to determine the trump suit for the game. The gameplay consists of a series of bouts where the attacker begins by placing a card, and the defender may attempt to beat it by playing a card of their own. After that, the attacker can continue the attack or end it, and the defender can continue to defend or pick up the cards. The winner of each bout becomes the attacker for the next bout. The aim of the game is to get rid of all one's cards. However, Durak is a game with hidden information, which makes it challenging to develop AI players. The goal of this work was to develop various AI players for the game, with a focus on achieving an average movement time of around 100 ms, and to compare their performance.

Framework and Selected Technology:

The game is implemented in C# for .NET 6. It includes a command-line interface with options for running experiments between AI players.

Artificial Intelligence Agents

The project features five distinct agents:

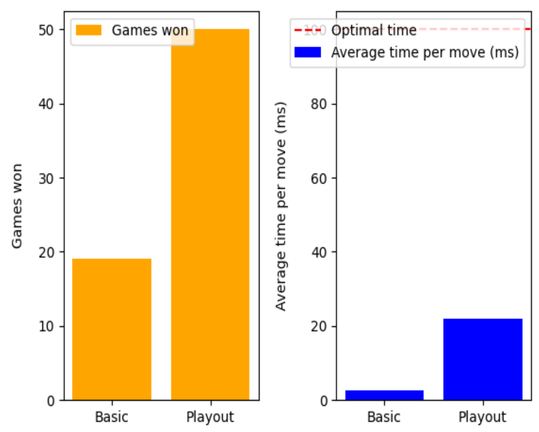
- **Random**: makes random moves without considering the game state.
- Rule-based agents:
  - **Greedy**: makes moves by selecting the lowest value cards.
  - **Smart**: uses strategies that take into consideration the opponent's hand.
- **Minimax**: a game-theoretic algorithm that evaluates the best move based on the minimax principle. It uses two heuristics to evaluate the state:
  -Basic, which is a traditional approach using the game state.
  -Playout, which generates a copy of the game state and plays out the remainder of the game.
- **MCTS (Monte Carlo Tree Search)**: an algorithm that uses simulated playouts of a game to estimate the value of each possible move.

Minimax Heuristic Comparison:

The basic heuristic function considers the following aspects when evaluation the state:

- Hand of the player
- Hand size of the player
- Existence of weakness, which are the ranks that are dominated by a card of the opponent

The following graph represents the playout heuristic being better than the basic.



Number of wins and average time taken per ply of basic and playout evaluation functions in the closed world

MCTS

Monte Carlo tree search consists of four main parts:

- **Selection**: The process of traversing the tree to select the most promising node to expand.
- **Expension**: Adding new child nodes to the selected node, representing unexplored moves.
- **Simulation**: Running random simulations from the expanded node to estimate the potential outcomes.
- **Backpropagation**: Updating the information in the tree, such as win/loss statistics, to improve future selections.

After the certain limit, the algorithm selects the best move from the generated game tree. This and selection part depends on The Upper Confidence Bounds applied to Trees (UCT) formula. It balances exploration and exploitation by considering the average reward and the number of visits of each child node.

Dealing with Hidden Information:

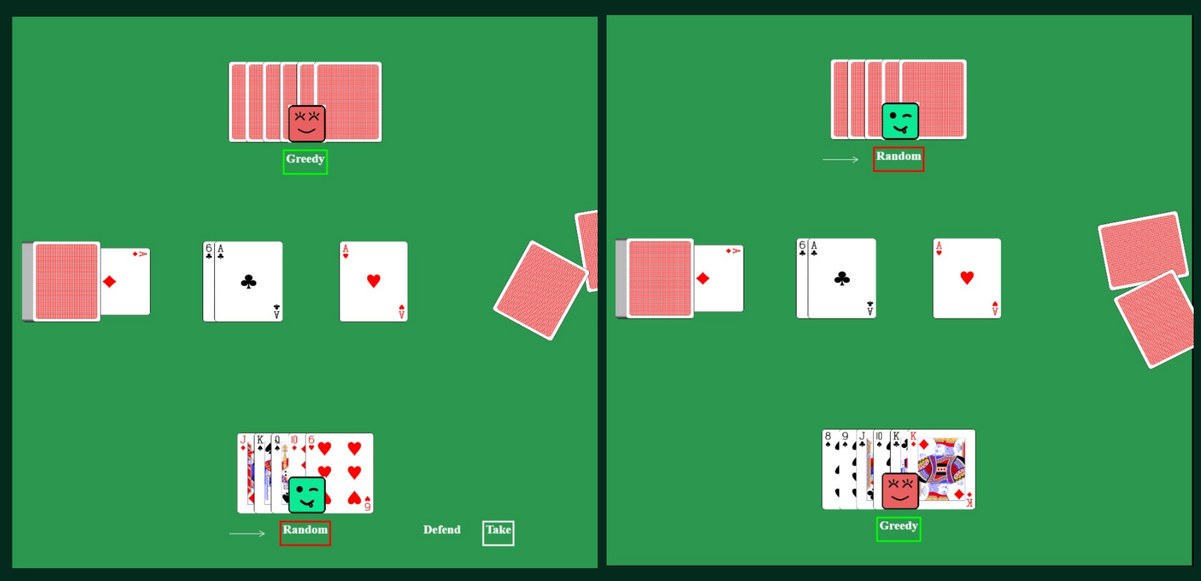
Durak poses challenges for the development of AI players with the hidden information. While some agents, such as Random, Greedy, and Smart, do not rely on this restriction as their move selection is based solely on their own cards, others, such as MCTS and minimax, cheat by exploring the game decision tree in the hidden information space. To overcome this limitation, they use sampling methods to randomly generate game states based on currently observable information. Through the iteration of many sampled games, these agents can select the optimal move while circumventing the hidden

Acknowledgments:

I am grateful to my supervisor Adam Dingle for his patience, expertise, and willingness to go above and beyond to provide assistance.

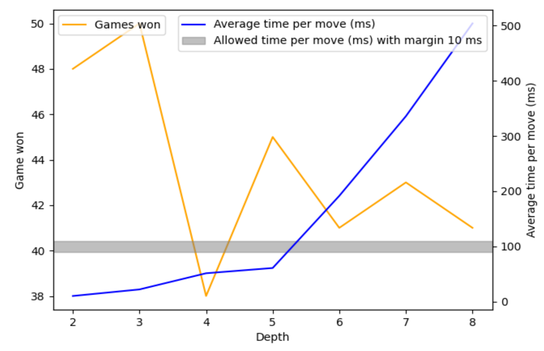
Branching Factor and Duration:

The maximum branching factor, the number of possible moves a player can make at each turn, is 35, but often players have only a few choices of cards to play, particularly the defender. An experiment was run to estimate the average branching factor in Durak by simulating 1000 random games played between two greedy agents and the result showed that the average branching factor, as computed using the geometric mean, was 2.17, which suggests that the branching factor of the game is low.
Regarding the duration with the same experiment, the average number of bouts per game between two greedy agents was 8.3, the average number of plies per bout was 5.3, and the average number of plies per game was 44.0. These results suggest that the behavior of the greedy agents led to relatively shorter games.

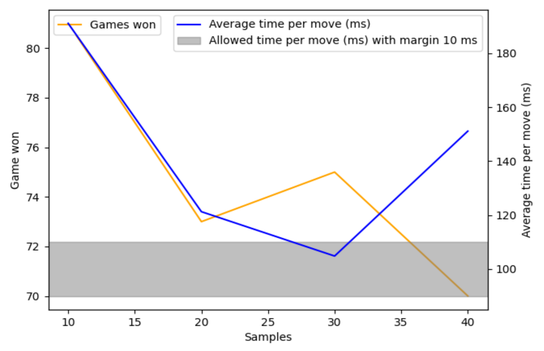


Minimax and MCTS Parameter Selection

The results of our experiments, as depicted in the accompanying figures, indicate that the optimal parameter settings for the Minimax and MCTS agents are a depth of 3 and an iteration limit of 30, respectively, with a fixed sample size of 30. It should be noted that while a higher iteration limit of 90 was found to yield the best results for MCTS using default parameter settings, the sample size was adjusted in order to maintain a constant computational time.



Performance and time per move by the Minimax agent with different depth parameters



Performance and time per move by the MCTS agent with differen limit parameters

Results:

In the closed-world environment, players do not have complete information about the game state and can only see certain elements such as their own cards and the trump card. To evaluate the performance of various agent strategies, the following agents were utilized, along with their parameters and respective average time to take a move:

- Random (~0.0218ms)
- Greedy (~0.0211ms)
- Smart (~0.0293ms)
- Minimax (depth=3,eval=playout,samples=30) (~169.1752ms)
- MCTS (limit=30,c=1.40,simulation=greedy,samples=30). (~45.1150ms)

The MCTS agent was the strongest, followed by the smart and minimax agents.

	random	greedy	smart	minimax	MCTS
random		8.4% ∓ 2.8%	6.2% ∓ 2.4%	6.9% ∓ 4.3%	1.5% ∓ 1.1%
greedy	91.5% ∓ 2.8%		50.6% ∓ 1.2%	47.4% ∓ 1.7%	19% ∓ 4.1%
smart	93.7% ∓ 2.4%	49.3% ∓ 1.2%		53.8% ∓ 3.7%	26.9% ∓ 4.7%
minimax	93.1% ∓ 2.6%	52.5% ∓ 1.7%	46.2% ∓ 3.7%		16.7% ∓ 3.8%
MCTS	98.4% ∓ 1.1%	80.9% ∓ 4.1%	73.1% ∓ 4.7%	82.8% ∓ 4.4%	

Win rates between agents in the open world (98% confidence intervals)

Conclusion:

This study developed an AI framework for the card game Durak and tested various agents. The playout heuristic in the minimax agent greatly outperformed the basic heuristic. Also, the sampling technique worked well for dealing with hidden information.In the tournament, the MCTS agent consistently outperformed the others in the game. The smart agent was the second most successful in the full game, though the minimax and greedy agents performed nearly as well as it.