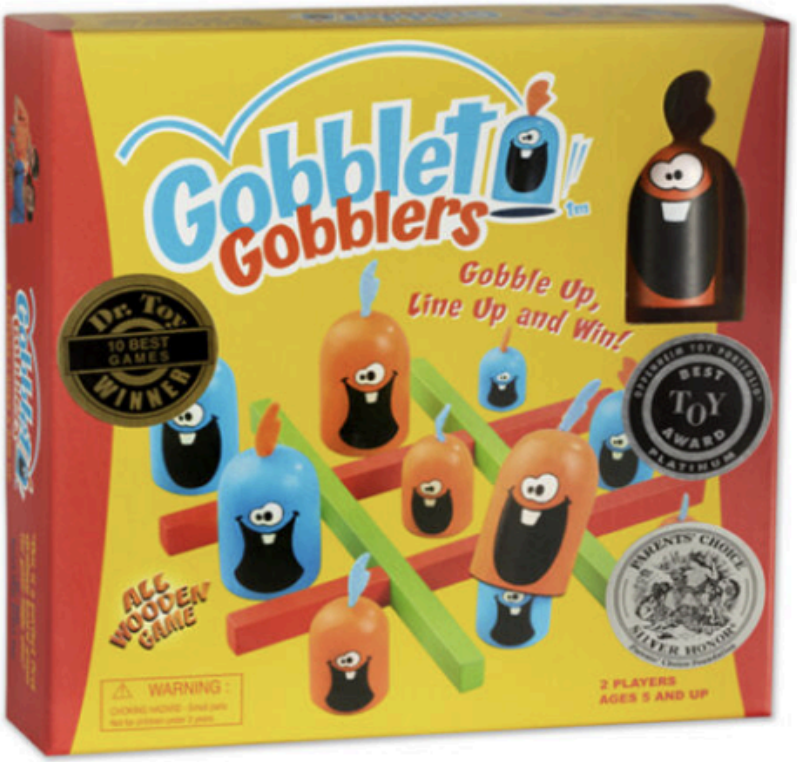
Strategy Selection Based on AI Agents

Gobblet Gobblers as a test case



Matan Kutz 205488679 matankutz@campus.technion.ac.il

Meitar Basson 208252452 meitarbasson@campus.technion.ac.il

Supervisor - Mr Mousa Arraf

Professor Shaul Markovitch

Introduction -

The main goal of our project was to investigate whether simple AI techniques can be used to discover and explain winning strategies for games. For example, in the classic game of Tic-Tac-Toe, a strategy such as controlling the center square or forming diagonal threats is well known. Our aim was not only to find a winning strategy and create a winning agent but also to ensure that the reasoning behind it would be understandable to humans. This contrasts with many modern AI approaches such as deep neural networks where the decision making process is often opaque.

Additionally, we wanted to provide strategy recommendations that adapt to different stages of a game. For instance, in chess, early moves focus on central control, while mid and late-game strategies vary depending on the board state.

To test our approach, we chose the game *Gobblet Gobblers*, a more advanced version of Tic-Tac-Toe (rules here). The game introduces additional complexity through the use of stackable pieces of varying sizes, while still maintaining an accessible rule set. This made it a suitable candidate for exploring human-understandable heuristics and strategies.

Problem Description -

We encountered several challenges early in the project. Our initial idea was to train a neural network and use interpretability techniques like FiLM to extract and explain strategies in a human readable way. However, training the network required a dataset of play games, which unfortunately we could not find online. While Tac-Tac-Toe datasets are available, they are not applicable due to the increased complexity of Gobblet Gobblers.

As a result, we decided to generate our own dataset by implementing the game and simulating over 5000 games using heuristics we wrote ourselves. Having our dataset we extracted a list of features that we wanted our network to be based upon and the final heuristic generated by the network was tested against human players, our original heuristics and also ChatGPT.

Dataset Creation -

As previously stated, we simulated many games between different AI agents in order to create a dataset. We tried to create several agents, each using another heuristic in order to get some variance in our dataset. Our heuristics were as follows -

1. **Blocking agent** - this agent will win only if it is possible to win within the next move, otherwise it will prefer blocking the opponent or just playing randomly in the case where no block is possible.
2. **2 Steps agent** - the agent will try winning in the next move. In the case where it's not possible to win in the next move, the agent will look ahead for his next move and simulate whether he could win. It is balanced with the opponent's chance of winning in its turn.
3. **Undercover agent** - this agent will win only if it is possible to win within the next move, otherwise it will prefer *covering* the opponent's piece randomly.
4. **Proactive agent** - this agent will win only if it is possible to win within the next move. If not, this agent will deepen its search and determine its best move considering the opponents best options as well.
5. **Blocking only agent** - unlike the previous stated blocking agent, this agent focuses only on blocking the opponent without even trying to win.

For each simulation we created a log file (partial example attached) which we used in order to create the dataset excel sheet and also extract the relevant features.

```
winner:1
turn:1
pos0-0:
pos0-1:
pos0-2:
pos1-0:
pos1-1:
pos1-2:
pos2-0:0:L,
pos2-1:
pos2-2:
turn:2
pos0-0:
pos0-1:
pos0-2:
pos1-0:
pos1-1:1:L,
pos1-2:
pos2-0:0:L,
pos2-1:
pos2-2:
```

Each board state was encoded using a fixed 3x3 grid structure, with positions ordered from top-left to bottom-right using 0-indexing. For every occupied cell, we recorded both the player ID and piece size (could be plural). For example, the entry pos2-0 refers to the cell in the third row, first column, which contains the first player large piece (0:L).

We also wrote a script log_to_excel in order to create an excel file containing the data in a more readable version. Snippet from this excel file is attached below -

Game Inde	Agent Player 1	Agent Player 2	Winner	Turn	Cell(0,0)	Cell(0,1)	Cell(0,2)	Cell(1,0)	Cell(1,1)	Cell(1,2)	Cell(2,0)	Cell(2,1)	Cell(2,2)
8000	random_improved_large	proactive_gobbler_agent	1	1	-	-	-	-	0:L	-	-	-	-
8000	random_improved_large	proactive_gobbler_agent	1	2	-	-	-	-	0:L	-	-	-	1:L
8000	random_improved_large	proactive_gobbler_agent	1	3	-	-	-	0:S	0:L	-	-	-	1:L
8000	random_improved_large	proactive_gobbler_agent	1	4	-	-	-	1:L	0:L	-	-	-	1:L
8000	random_improved_large	proactive_gobbler_agent	1	5	-	-	-	1:L	0:L	-	0:S	-	1:L
8000	random_improved_large	proactive_gobbler_agent	1	6	-	-	-	1:L	0:L	-	1:L	-	-
8000	random_improved_large	proactive_gobbler_agent	1	7	0:M	-	-	1:L	0:L	-	1:L	-	-
8000	random_improved_large	proactive_gobbler_agent	1	8	0:M	-	-	1:L	0:L	-	1:L	-	1:S
8000	random_improved_large	proactive_gobbler_agent	1	9	0:M	-	-	1:L	0:L	-	1:L	-	0:L

Features extracted -

To train our neural network effectively, we designed a set of 25 features that represent each board state in a compact, and informative manner. These features aim to capture both spatial properties of the board and strategic elements relevant to the gameplay.

Below is a summary of the features we used -

1. Column counts - number of the pieces in each column
2. Row counts - number of the pieces in each column
3. Winning/Losing player pieces on board - how many pieces of each size
4. Winning/Losing player pieces captured - how many pieces of each size were captured
5. Average location (vertical / horizontal) of the player's pieces
6. Player spread - how spread out the player's pieces are across the board.
7. Total pieces on board - both players combined
8. Game progress - vector to encode the game stage, early, mid or late.

This feature set was designed to reflect meaningful aspects of the game that could help a neural network (and also a human observer) infer potential strategies. For example:

- Column and row counts give insight into imminent threats or opportunities to align three pieces.
- Piece sizes matter since larger pieces can cover smaller ones.
- Spatial features like average position and spread are helpful for understanding board control.
- The game progress is used in order to generate a strategy for each game stage as stated earlier.

Together, these features form the input model to the neural network.

Neural Network Architecture (maybe add images too)

Where AI was used during this project

Experiments and Results (also add images)