Bagh Chal – Game of Goats and Tigers

Project by: Raj Surya (rajen064@umn.edu)

Description

Bagh Chal is a strategic two-player board game. The game is asymmetric in that one player controls four tigers while the other controls up to twenty goats. The tigers 'hunt' the goats while the goats attempt to block the tigers' movements. The game is played on a 5x5 board. Pieces are positioned at the intersection of the lines and not inside the areas delimited by them. Lines connect directions of valid movement between these points. A variation of this game is prevalent in my native town of Aruppukkotai in India, and I have grown up watching the elders especially senior citizens playing this game. This motivated me to develop an AI agent to play this game.

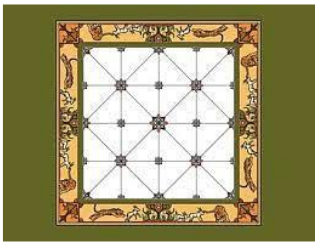

Figure 1: Board of Bagh Chal [Source: Wikipedia]

Rules for tigers:

- They can move to an adjacent free position along the lines.
- They can capture goats during any move and do not need to wait until all goats are placed.
- They can capture only one goat at a time.
- They can jump over a goat in any direction, as long as there is an open space for the tiger to complete its turn.
- A tiger cannot jump over another tiger.

Rules for goats:

- Goats cannot move until all goats have been positioned on the board.
- They must leave the board when captured.
- They cannot jump over tigers or other goats.

The game is over when either the tigers capture five goats, or the goats have blocked the tigers from being able to move.

Literature Review:

This game has not been widely studied by the research community in the context of game theory or artificial intelligence. The little study has shown that the complete database of all positions that arise consists of 88,260,972[1] entries. The placement phase where goats are introduced into the board involves a search whose game-tree complexity is estimated to be of the order 10. Partial results show that the Tigers appear to have the better part of the opening[2], but Goats may catch up in the long run. However, there is no research on heuristics for this game so all the heuristics used in this development are novel as well as the implementation of the game.

Game Implementation:

Requirements:

Python

Agents:

The game offers four different agents that can be used to play the game.

1. Human Agent: This agent allows the user to play interactively by choosing an action. The format and the type of input expected is given as a prompt to the user at every stage. Briefly,

when the user is playing as tiger, the user inputs two coordinates. The start location of the tiger in the grid and the end location. When the user is playing as goat, during the placement phase where goats are being placed, the user inputs a coordinate to place the goat. Once all the goats have been placed, the user gives two inputs, the start and end location of the goat. If any invalid move is given, the game discards that move and gives you another chance.

2. Random Agent: This agent chooses a random move from the list of all possible valid moves.
3. Alphabeta Agent: This agent uses alpha beta search and a heuristic to search the game tree for an optimal move. This agent can also be configured to use dynamic depth for alpha beta search by setting the variable dynamic to True when initializing. The dynamic depth allows the agent to choose a depth based on the current state. If the current state is favourable, then a small depth is chosen. But if the current state is bad, then a large depth is chosen, so that the agent can try and recover from it. The values have been tuned based on a large number of trials and analysis to improve performance and reduce time.
4. Monte Carlo Tree Search Agent: This agent uses pure monte carlo search to find the optimal move. The number of iterations or rollouts can be passed as parameter while initializing the agent.

All agents while initializing require a parameter 'T' or 'G' to be passed, which indicates whether the agent is playing as tigers ('T') or goats ('G').

Heuristic:

The heuristic used for alpha beta search is based on the following factors.

1. Number of goats captured.
2. Number of goats that are yet to be placed.
3. Number of valid moves available.
4. Number of tigers that don't have any valid moves i.e number of tigers captured.
5. Number of goats that cannot be captured immediately in the next move.
6. Number of goats that can be captured in the next move.

The final heuristic is a weighted sum of all these factors and the weights have been tuned by trial and error method. The number of goats/tigers captured has the highest weight.

The number of moves per match is limited to 50 as it has been found that the agents tend to repeat their moves and wait for the other to sacrifice. The average number of moves for the game is known[2] to be 39, hence 50 is a suitable choice.

To run the game, in the *main* function, choose two agents and allot their role along with any parameter if required. Then just run the game. No special instructions required.

Results:

Sl. No	Agent playing as Tigers	Agent playing as Goats	Total Number of Matches	Number of wins by Tigers	Number of wins by Goats	Number of draws	Average time per match (seconds)
1	Human	Random	10	10	0	0	NA
2	Random	Human	10	0	10	0	NA
3	Random	Random	100	99	1	0	0.44
4	Random	Alphabeta Depth=4	10	0	4	6	5.62
5	Alphabeta Depth=4	Random	10	1	0	9	7.78
6	Alphabeta Depth = 4	MonteCarlo Iterations=200	50	5	0	45	13.47
7	MonteCarlo Iterations=200	Alphabeta Dynamic Depth	50	0	23	27	57.20
8	Alphabeta Depth = 6	Random	10	9	1	1	79.24

9	Random	Alphabeta Depth = 6	10	0	9	1	225.79
10	Alphabeta Dynamic Depth	Random	10	10	0	0	40.35
11	Random	Alphabeta Dynamic Depth	10	0	9	1	9.33
12	MonteCarlo Iterations=200	Random	10	10	0	0	10.6
13	Random	MonteCarlo Iterations=200	10	8	0	2	10.15
14	Random	MonteCarlo Iterations=10000	1	0	0	1	315.6
15	Human (Beginner)	Alphabeta Dynamic Depth	1	0	1	0	NA
16	Alphabeta Dynamic Depth	Human (Beginner)	1	1	0	0	NA
17	Human (Intermediate)	Alphabeta Dynamic Depth	1	0	0	1	NA
18	Alphabeta Dynamic Depth	Human (Intermediate)	1	0	0	1	NA
19	Human (Beginner)	Monte Carlo Iterations = 200	1	1	0	0	NA
20	Monte Carlo Iterations = 200	Human (Beginner)	1	0	1	0	NA

Analysis:

Random Agent: Since the game is asymmetrical, it is seen that random agent performs way better when performing as tiger than goat. This is because, goats gain advantage as game progresses but tigers have initial advantage. Hence to play as goats, the ability to think ahead is required.

Alphabeta Search: The heuristics developed are quite robust. They are able to beat beginner human players even with small depth. For good performace, atleast a depth of 6 is required, however by using dynamic depth, the time can be improvised. When playing as goats, the heuristics direct the placement of goats towards the edges and corners to avoid being captured.

Monte Carlo Search: It has been seen that monte carlo search doesn't perform well. This is because the game is asymmetrical. Monte Carlo uses random agent to simulate the game, but since tigers have the upper hand while random agents play, the monte carlo agent doesn't have enough data to figure out the optimal move especially while playing as goats. This can be seen as a biased data being used for training a network.

References:

1. Lim Yew Jin and J. Nievergelt, "Computing Tigers and Goats", ICGA Journal, Sept 2004
2. Sakshi Agarwal and Hiroyuki Iida, "Analyzing Thousand-Year-Old Game: Tigers and Goat is Still Alive", Article in Information Technology and Tourism, Oct 2018