King Saud University
College of Computer and Information Sciences
Computer Science Department

361 Project
Spring 2021

Othello Agent Project

مهند الرشيد 439101298

عبدالملك العرجاني 439102162

عبدالعزيز الجمهور 439101644

Problem

Othello is a difficult game with a high branching factor and long search tree, building an AI that can tackle such game would require it to be 1- Adversarial (i.e. predicting the opponents moves and trying to outmaneuver them), 2- Fast on choice (since the agent wouldn't be given time in an adversarial game).

Data Structures

In order for us to design a fast agent, we must sacrifice the luxury of traversing the entire search tree and evaluate moves midway through the search, so we created an evaluation function which uses a **weighted sum** of various metrics and produces a result. We used a hashmap(Metric -> Weight) in order to represent the weights.

```
weights = new HashMap<String, Double>();
weights.put("Moves", 15.);
weights.put("Corner", 20.);
weights.put("Score", 5.);
weights.put("Winner", 99999.);
```

The agent also has a couple of attributes that dictate its behavior

```
final boolean USE_ALPHA_BETA = true;
final boolean USE_TOURNAMENT = false;
final int MAX_CUTOFF = 5; // Used if USE_TOURNAMENT == true
int cutoff = 6; // Used if USE_TOURNAMENT == false
```

Evaluation Function

For our evaluation function we considered many metrics, one of them the corner. If there is a corner move then the agent will tend to pick that move over others. We also considered the score at that move as a metric but gave it a low weight as it changes frequently throughout the game

Results

We tested the agent in 5 matches with different depths against a random agent and against itself, here are the results (Note, some of these examples are with **α, β** turned ON, it speeds up the search since it cuts the exponent by half):

α, β = False	MinMaxer			MinMaxer_2		
Winner	Depth	Time	Score	Depth	Time	Score
MinMaxer	1	0.033	37	1	0.049	27
MinMaxer	2	0.29	51	2	0.23	13
MinMaxer_2	3	2.45	25	3	1.92	39
MinMaxer_2	4	10.45	13	4	18.63	51
MinMaxer_2	1	0.025	26	4	27.5	38

α, β = False	MinMaxer			RandomAgent	
Winner	Depth	Time	Score	Score	Time
MinMaxer	1	0.045	37	27	0.036
MinMaxer	4	30.4	58	6	0.012
MinMaxer	2	0.2	44	20	0.02
RandomAgent	3	1.4	25	39	0.01
MinMaxer	2	0.49	36	28	0.033

α, β = True	MinMaxer			MinMaxer_2		
Winner	Depth	Time	Score	Depth	Time	Score
MinMaxer_2	1	0.03	25	6	7.8	39
MinMaxer	4	1.39	34	1	0.033	30
MinMaxer_2	2	0.083	23	2	0.089	41
MinMaxer	5	5.26	38	3	0.33	26
MinMaxer_2	6	14.355	28	6	14.26	36

α, β = True	MinMaxer			RandomAgent	
Winner	Depth	Time	Score	Score	Time
RandomAgent	1	0.057	27	37	0.03
MinMaxer	4	1.3	38	26	0.011
MinMaxer	2	0.83	38	26	0.025
MinMaxer	3	0.579	49	15	0.019
MinMaxer	2	0.097	44	20	0.027

Conclusion

Creating an intelligent agent that solves Othello can be quite hard given that the evaluation function dictates which moves are optimal, and that the MinMax algorithm relies on the other player playing optimally for its optimality.

Comparing agents is difficult given a handful of matches, since factors like which player plays first, and random chance might affect the performance of an agent.

α, β pruning can massively decrease the time needed to perform MinMax, e.g. see **Table 2 Row 2** and compare with **Table 4 Row 2**