Exploration of Abalone game-playing agents

Ture Claußen, 202132027, ture.claussen@stud.hs-hannover.de

Hochschule Hannover Fakultät IV

Abstract. Perfect information games provide a good environment for artificial agents to navigate in, as they have a clear performance measure for comparison with each other and humans. Their determinism removes some of the engineering problems of agents in the physical world. In the following we implement and compare alpha-beta pruning and Monte Carlo Tree Search for the game Abalone, to come to a conclusion about their resource-consumption and performance.

Keywords: AI · Alpha-beta · Monte-Carlo-Tree-Search · Abalone · Intelligent Agents

1 Introduction

1.1 Motivation

Overall, this degree of complexity makes the game a good project for the design of a game playing-agent, as it is meant to be an opportunity to apply the fundamental principles and algorithms learned in the class, as opposed to being distracted by the engineering aspects. This matches my personal background on the subject matter, as I have no prior (formal) exposure to the design of artificial intelligence. In addition, this project is only created for the purpose of this class.

Over the course of my current study of applied computer science I gained versatile proficiency in programming and the handling of data which will help implement the algorithms efficiently and provide the empirical foundation for the paper. The project will also be a valuable training for my upcoming bachelor thesis.

1.2 Related work

Considering the existing landscape of papers, there is unquestionably a wide array of papers exploring the application of minimax and alpha-beta pruning on the game of Abalone. Some of the most prominent include:

1. "Algorithmic fun-abalone" (2002) Considers foundational heuristics for the game and analyzes minimax and its refinements in the form of (heuristic) alpha-beta pruning. Furthermore it sheds light on the performance differences between those. [1]

2. "A Simple Intelligent Agent for Playing Abalone Game: ABLA" (2004) Implementation of a game-playing agent with minimax, alpha-beta pruning and some custom heuristics. The evaluation of the performance is done by comparing the agent to existing software in the form of ABA-PRO and RandomSoft. [4]
3. "Constructing an abalone game-playing agent" (2005) Provides a very thorough explanation and analysis of the game's fundamentals, such as the state space, rules and positions. In regards to the alpha-beta pruning it also explains strategies for ordering the nodes and performance concerns. [3]
4. "Implementing a computer player for abalone using alpha-beta and monte-carlo search" (2009) This master thesis is a very exhaustive analysis of the game, alpha-beta pruning and Monte Carlo tree search, conferring many of the previous results. [2]

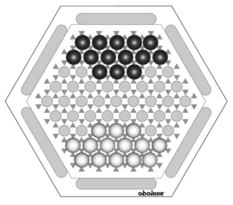
These resources give great insight into the classical approaches, but they are lacking certain qualities:

- Accessible and freely explorable code that underlies the analysis
- Comparison with modern approaches like Q-Learning that might reduce the resource demand on the client side

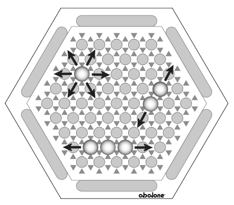
The proposed project seeks to build upon the given insight to improve upon these missing qualities.

1.3 Rules

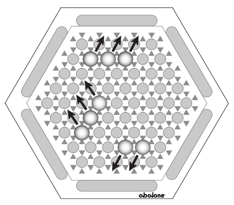
The goal of the game is to push six of the opponent's marbles off the playing field. The game's starting position is depicted in figure 1 (a). One, two, or three adjacent marbles (of the player's own color) may be moved in any of the six possible directions during a player's turn. We differentiate between broadside or "side-step" moves and "in-line" moves, depending on how the chain of marbles moves relative to its direction, which is shown in figure 1 (b) and (c).



(a) Starting position



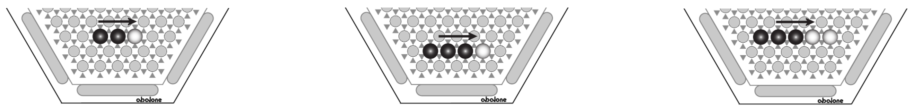
(b) "In-line" moves



(c) "Side-step" moves

Fig. 1: Basic moves [6]

A move pushing the opponent's marbles is called "sumito" and comes in three variations, as shown by figure 2. Essentially, the player has to push with superior numbers and the opponent's marbles can not be blocked. This is the game mechanic that allows for pushing the marbles out of the game and winning.



(a) "2-push-1" sumito (b) "3-push-1" sumito (c) "3-push-2" sumito

Fig. 2: Sumito positions allow pushing the opponent's marbles [6]

2 Project details

2.1 Agent design

Based on the PEAS framework we can analyze the task environment for the agent. [5, p.107]

Performance measure Win/loss, number of moves, time to deliberate
Environment Digital playing board
Actuators Move marbles, display text to CLI
Sensors Position of marbles

If we look at the environment more closely we see that it is fully observable, two-agent, competitive, sequential, static and discrete.

2.2 Complexity

An important characteristic of a game environment is its complexity, which can be described in two relevant dimensions.

State space complexity The state space is the collection of all possible states the agent can be in. [5, p. 150] For Abalone this means we have to consider all possible board configurations with different numbers of marbles present. Additionally, we would have to correct duplicates that arise from the symmetries of the board. Ignoring this fact the following gives a good upper bound:

$$\sum_{k=8}^{14}\sum_{m=9}^{14} \frac{61!}{k!(61-k)!} \times \frac{(61-k)!}{m!((61-k)-m)!}$$

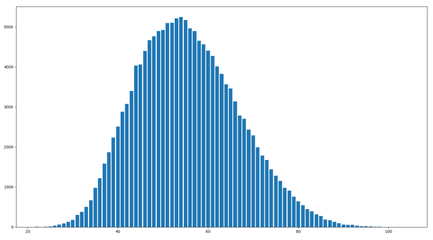


Fig. 3: Counts of moves available for random for random player in 5 games

Game tree complexity The game tree defines the dependencies between board positions (nodes) and moves (edges). First we consider the branching factor (how many moves are possible in one position) of the game tree, which is on average 60. We combine that number with the height of the tree to get the total number of leaves. As the length of a game varies greatly, we use the average length of a game which is 87: 60^{87} [3]

Putting Abalone's complexity in relation with other popular games, its state space complexity is on the same level as Reversi, whilst its game tree surpasses chess in complexity (c.f. table 1)

Game	state-space complexity (log)	game-tree complexity (log)
Tic-tac-toe	3	5
Reversi	28	58
Chess	46	123
Abalone	24	154
Go	172	360

Table 1: Abalone in comparison with other games [2]

3 Conclusion

Overall, the implementation of the agents posed a much greater engineering challenge than expected. The basic algorithms were implemented quickly, but two tweak them until they have acceptable move-times required a lot of effort. It is interesting that the basic implementation of the Monte Carlo Search agents performed so poorly. Especially, combined with more modern techniques the MCTS agent still is extremely promising and worth further investigation.

References

1. Aichholzer, O., Aurenhammer, F., Werner, T.: Algorithmic fun-abalone. Special Issue on Foundations of Information Processing of TELEMATIK **1**, 4–6 (2002)
2. Chorus, P.: Implementing a Computer Player for Abalone Using Alpha-Beta and Monte-Carlo Search. Master Thesis, Citeseer (2009)
3. Lemmens, N.: Constructing an abalone game-playing agent. In: Bachelor Conference Knowledge Engineering, Universiteit Maastricht. Citeseer (2005)
4. Özcan, E., Hulagu, B.: A Simple Intelligent Agent for Playing Abalone Game: ABLA (Jan 2004)
5. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Pearson Education, Inc, fourth edn. (2021)
6. S.A., A.: Abalone rulebook. https://cdn.1j1ju.com/medias/c2/b0/3a-abalone-rulebook.pdf