Min-Max Kalah

Haruki Gonai (hg2541) and David Cendejas (dc3448)

November 29, 2022

1 What is Kalah?

Kalah is a two-player game involving seeds and a board with pits. The board consists of 12 small pits, called houses, and two big pits, called stores, as shown in Figure 1. Initially, each house contains four seeds.

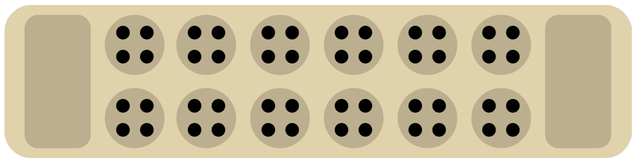

Figure 1: Initial board arrangement in Kalah

Each player owns a row of six houses and the store to their counter-clockwise direction. Each turn, the player picks one of their houses containing at least one seed. The player then removes the seeds from the house and deposits a seed into each pit (except the opponent's store), going counter-clockwise.

For example, suppose the six houses in the bottom row of Figure 1 belong to a player, and the player chooses their house third from the right. The player removes the four seeds from this house and deposits a seed into each pit, going counter-clockwise, as depicted in Figure 2.

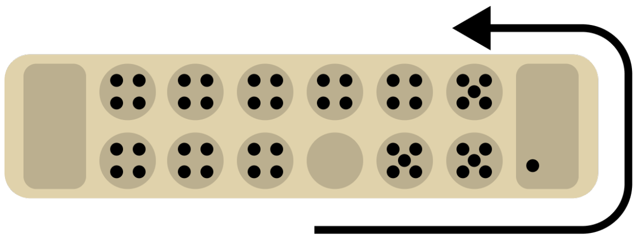

Figure 2: Kalah board after a move

If the last pit into which the player deposits is their own store, they get another move. If the last pit into which the player deposits is an empty house that belongs to them, and the house across from it is not empty, then the player moves all the seeds in these two houses to the player's store. The game ends when either player has no seeds in any of their houses, at which point the winner is whoever has more total seeds in their houses and store [2].

2 Kalah AI

The objective of this project is to create a fast Kalah AI using Haskell that computes the best move given the current board state. Since Kalah is a zero-sum game with perfect information, we will create an AI that uses the min-max algorithm. Since the min-max algorithm will traverse through many board states in DFS style, we can leverage parallelism to speed up the AI.

The basic algorithm we will use can be found in this GitHub repository, which contains Python code for a single-threaded alpha-beta pruning Kalah AI [1].

3 Goals

The goals of our project are as follows:

1. Design and implement a sequential, min-max Kalah AI (in Haskell, of course).
    - Here, we will determine if we need to impose a depth-limit to have the algorithm terminate in a reasonable amount of time. Why this is relevant will be explained in goal 5.
2. Design and implement a seqeuntial, alpha-beta pruning Kalah AI.
3. Identify the bottlenecks of the sequential min-max Kalah AI, and speed it up using parallelism to create the parallel min-max Kalah AI.
4. Run the sequential min-max Kalah AI, the sequential alpha-beta pruning Kalah AI, and the parallel min-max Kalah AI on various board states and compare their performance using ThreadScope. We will also experiment using different numbers of cores.
5. If we needed to impose a depth-limit in min-max or alpha-beta pruning, then the move returned by the algorithm is not guaranteed to be optimal (assuming the opponent plays optimally) [3]. Hence, if a depth-limit was necessary, we will also compare the optimality of the AIs by making them play against each other and analyzing the win-loss statistics.
6. If time permits, we also plan to investigate how our parallel min-max Kalah AI performs on different Mancala boards (e.g. with 20 houses instead of 12).

References

[1] Gonai, H. (2022, November 26). *kalah-min-max*. GitHub. Retrieved November 28, 2022, from https://github.com/harukigonai/kalah-min-max

[2] Wikimedia Foundation. (2022, November 6). *Kalah*. Wikipedia. Retrieved November 28, 2022, from https://en.wikipedia.org/wiki/Kalah

[3] Wikimedia Foundation. (2022, November 23). *Minimax*. Wikipedia. Retrieved November 28, 2022, from https://en.wikipedia.org/wiki/Minimax