2048 Multithreaded Solver

Linh Bui

Introduction

Intro

Objective: This Haskell program implements a solver for the 2048 game using the Expectimax algorithm. The solver simulates games and evaluates moves using heuristic functions.

Key Features:

- Intelligent move generation.
- Heuristic evaluation.
- Simulation of multiple games to analyze performance.

Board Representation

Data Structure:

- The game board is represented as a 4x4 grid of integers (`Board = [[Int]]`).
- Empty tiles are represented by `0`.

Weight Matrix:

- A matrix (`weights`) prioritizes high-value tiles in favorable positions for heuristic evaluation.

Game Logic

Initialization:

- `initialBoard`: Creates an empty board and adds two random tiles (either `2` or `4`).
- `addRandomTile`: Places a new random tile at an empty position with probabilities `90%` for `2` and `10%` for `4`.

Move Generation:

- `getMoves`: Simulates all possible moves (`Up`, `Down`, `Left`, `Right`) and returns valid resulting boards.
- `moveLeft`/`moveRight`: Implements the logic for merging and shifting tiles in a row.

Game State Checks:

- `isFull`: Checks if the board is completely filled.
- `hasReachedTarget`: Checks if the `2048` tile is present.

Heuristic Functions

- **Monotonicity (`getMonotonicity`)**:
  - Measures how aligned the tiles are in descending order along predefined paths.
  - Higher monotonicity scores indicate a more organized board.
- **Smoothness (`getSmoothness`)**:
  - Penalizes large differences between neighboring tiles, encouraging smooth transitions.
- **Weighted Sum (`getWeightedSum`)**:
  - Rewards high-value tiles in important positions based on the `weights` matrix.
- **Max Corner (`getMaxCorner`)**:
  - Rewards the board if the largest tile is in the top-left corner, which is a strategic goal in 2048.
- **Combined Heuristic (`heuristic`)**:

```
heuristic = monotonicity - smoothness + weightedSum + maxCorner
```

Expectimax Algorithm

Purpose: Simulates moves and evaluates their outcomes using the heuristic to find the best move.

1. **Expectimax Logic**:
   - **Maximizing Node**: Simulates player moves and chooses the move with the highest expected score.
   - **Chance Node**: Simulates the random addition of tiles (`2` or `4`) and calculates the weighted average of outcomes.
2. **Parallelization**:
   - The evaluations of possible moves (`getMoves`) and random outcomes (`calculateChance`) are parallelized using `parMap rpar` to utilize multiple CPU cores.

Simulation

`simulate10Games`: Runs 10 simulations, calculates success rates, and averages the number of moves.for 1,2,4,6,8,10 cores

Outputs:

- Total games reaching `2048`.
- Average number of moves per game.

Results

- Was not able to get 2048 consistently, only about half the time 2048 was reached.
- Reached 2048 31 out of 60 runs

Core	Moves	Total time	Time per move
10	7917	239.7	0.03
8	8948	293.8	0.032
6	8674	310.7	0.035
4	9000	328.3	0.036
2	7922	330.6	0.041

Thanks!