# Expectimax 2048

yc3858 Yufan Chen

The proposal proposes a parallel implementation of expectimax algorithm applying to the Game 2048.

## Background

The introduction and the rule of the game 2048 can be found on wikipedia.

Several projects from previous PFP classes used minimax algorithm to develop an AI to automatically play this game. However, actually there is no "opponent" in this game. Specifically, the system will not purposely generate a tile on the place that makes the player having the least winning probability. Instead, after each move of the player, the system will generate a new tile in a random empty position, and it has a probability of 90% for generating a 2, and 10% for 4.

Based on the reason above, we may want to try Expectimax algorithm, which better fits the fact that the "opponent" generates new tiles under a specific probability model.

## Possible Parallel Components

There are several procedures able to be parallelized:

1. Search the game tree in parallel
2. Calculate different terms in the heuristic function in parallel
3. Move and merge tiles in parallel

## Performance Evaluation

We will evaluate the performance of the implementation via two dimensions:

1. The success rate of generating an 2048 tile
2. The time it takes to generate an 2048 tile

We will run the final program 10 times and get the average results of the metrics above.

## Optimization

We can optimize the performance by taking methods below:

1. Check if the workload is evenly distributed to every CPU core. If it isn't, we may want to change the implementation to make the workload more balanced, such as making the tasks more fine-grained.
2. Check if too many sparks are generated so that the overhead is heavy. If it is, we can try to change the implementation to make the tasks more coarse-grained.
3. Play with different search depths to get the best tradeoff between the success rate and the running time.

## Benchmark

To better evaluate what benefit we can get after introducing expectimax and parallel, we can compare the final implementation with:

1. python sequential expectimax implementation
2. haskell sequential expectimax implementation
3. haskell parallel minimax implementation with alpha-beta pruning

# Testing Environment

MacBook Pro Mid 2015

CPU: 2.2 GHz Quad-Core Intel Core i7

Memory: 16 GB 1600 MHz DDR3

# Reference


1. https://en.wikipedia.org/wiki/2048_(video_game)
2. https://www.geeksforgeeks.org/expectimax-algorithm-in-game-theory/
3. https://stackoverflow.com/questions/22342854/what-is-the-optimal-algorithm-for-the-game-2048