Parallel Decisions in Strategic Game

Ping-Kuan Kao
Institute of Network Engineering
310552053

Kuang-Hsiang Kuo
Institute of Computer Science and Engineering
311551104

Cheng-Yuan Liu
Department of Computer Science
0816105

KEYWORDS

parallel programming, strategic game

1 INTRODUCTION

There are more and more fun and strategic games recently, and people are trying hard to win the game. This kind of game is very addictive, and 2048 is one of them. In 2048, mostly there will be 4x4 tiles displayed on the screen, initially 2 tiles with a number on it, and the rest is null. All the number displayed will be the power of 2. Player has four decisions, which are up, down, left and right. Once the player chooses one of them, all the tiles with number will move to that direction. Tiles with same numbers will be combined into one tile and the combined tile will be the sum of the original two tiles. For each move, there will be one additional tile with number randomly appeared on the null tile. Player's goal is to get the number **2048**. If the player doesn't get 2048 but all the 4x4 tiles are numbered, then the player loses.

We can see that the player has only four decisions, up, down, left and right, so we can simply move four directions by turns, and compute the best move we can get. However, moving four directions by turns is time-consuming. We can use the concept of parallel programming to speed up the time for making the decision. Using multi-threads to compute the value we can get for each move, and directly get the result. This four moves are independent from each other, so we don't have to worry about the synchronization problem.

The rest of this paper is organized as follows: Section 2 presents the statement if the problem. In Section 3, we introduce our proposed approaches and the algorithm in details. In Section 4, we describe how we choose the programming language. Related work and the expected result are presented in Section 5 and Section 6. Last but not least, our timetable is in Section 7.

2 STATEMENT OF THE PROBLEM

We choose to do the analysis of the game agent for the 2048 game.

2.1 2048 game

In each round, the map will randomly create 1 tile with a value equal to 2 or 4, and the player can choose to move up, down, right, or left in able to combine the tiles with the same value. Once the tiles combine, the value will multiply by 2. If the player can't move in any direction, then the game is over. The final goal of this game is to synthesis a tiles with a value equal to 2048.

2.2 Player Strategy

If you look up on the Internet, you will discover that there are a lot of walkthroughs about this game. Since these strategies are for human beings, most of them make their decision based on some simple rules, such as putting the tile with the largest value in one corner, and other tiles in the same row or column arranged in descending order, etc. However, these simple rules can't handle every situation perfectly, and people can't simulate a large number of possible results without other techniques support, like computers. Hence, we want to simulate as many rounds as possible and predict the next step based on the result by the algorithm.

2.3 Greedy method strategy

Using the greedy method (For example: Minmax, MCTS (Monte Carlo Tree Search)) to help us predict the next step. Before making any step in each round, the algorithm agent will simulate the result of each direction, including which tiles will combine and where the random tile will occur. When all the results are simulated, the agent will make the same simulation on the previous result. After a limited number of layers, the agent will stop simulating, calculate the score of each direction and choose the one with the highest score.

Since these kinds of algorithms will not influence each other in each layer, we want to increase the number of simulations by parallelization.

3 PROPOSED APPROACHES

The block diagram of project is shown at **Figure 1**.

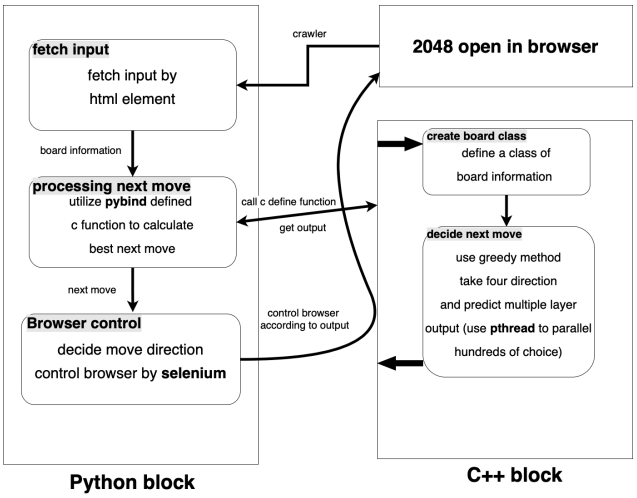

Figure 1: block diagram

The following part explains the function of each block.

Python block

- `fetch input` : Open online 2048 in browser by **selenium** [1]. Then fetch information of board with crawler matching html items.
- `processing next move` : Run c++ defined function with board information. Include C++ defined function in python by **pybind**[2].
- `browser control` : After deciding next move, control browser with **selenium**.

C++ block

- `create board class` : Define a class of board information. When taking the information of board, initiate a new instance of the board.
- `decide next move` : The main block about algorithm. For each possible case, we would utilize **Pthread**[3] to accelerate the process.

4 LANGUAGE SELECTION

We choose **Pthread** to parallel the program. During programming, we might utilize one main thread to manage whole memory. **Pthread** is relatively intuitive to control every thread. For each block which should be paralleled will be assigned with thread we manage. Syntax of creation and killing a thread is not hard to hand, so we might just focus on race condition and deadlock by semaphore or mutex lock.

5 RELATED WORK

Main algorithm to decide next move is **Minmax**. We have found some programs and forums about this method. Due to our reference are written by other languages, we would transform them to C++ with same algorithm.

The following links are forums we reference.

- `stackoverflow`: [4]
- `Medium`: [5, 6]

6 STATEMENT OF EXPECTED RESULTS

The original code evaluates each move by turns, then make the decision. However, after adding the concept of parallel programming, we use threads to evaluate four moves at the same time. However, this game is not guaranteed the player to win. We should evaluate the original code and the parallel code only under the winning condition.

Since the algorithm we use to make the decision is **Minmax**, the depth of the tree will influence the probability for player to win. The deeper the depth, the higher probability the player will win. However, the deeper the tree, the more time it takes. We will compare the depth equal to 1, to the maximum depth, and see the run time between the original code and the parallel code.

We expect the parallel code to run much faster than the original one, since the parallel code can compute faster. From another aspect, the parallel code can take more factors into consideration. We also expect that the parallel version may have the same running time with the original one, but the depth of the tree will be much deeper than the original one!

7 TIMETABLE

Week	TODO
2-3 (9/19 - 9/29)	Group registration
4-6 (10/3 - 10/21)	Proposal discussion
7 (10/24 - 10/27)	Writing the proposal
8-9 (10/31 - 11/11)	Convert the Python code into C
11 (11/14 - 11/18)	Speed up the C code using Pthread
12 (11/21 - 11/25)	Presentation preparation
13- (11/28 -)	Writing the final project report

REFERENCES

[1] https://selenium-python.readthedocs.io
[2] https://pybind11.readthedocs.io/en/stable/
[3] https://www.cs.cmu.edu/afs/cs/academic/class/15492-f07/www/pthreads.html
[4] https://stackoverflow.com/questions/22342854/what-is-the-optimal-algorithm-for-the-game-2048
[5] https://medium.com/@bartoszzadrony/beginners-guide-to-ai-and-writing-your-own-bot-for-the-2048-game-4b8083faaf53
[6] https://towardsdatascience.com/playing-2048-with-minimax-algorithm-1-d214b136bffb