Playing 2048 with Expectimax

Project Plan

Contents

[Abstract 2](#_Toc115899361)

[Introduction 2](#_Toc115899362)

[Timeline 3](#_Toc115899363)

[Term 1 3](#_Toc115899364)

[Term 2 3](#_Toc115899365)

[Risk Assessment 5](#_Toc115899366)

[Early Deliverables 4](#_Toc115899367)

[Final Deliverables 4](#_Toc115899368)

[References 6](#_Toc115899369)

# Abstract

In this project, I plan to make a program capable of playing the game 2048 using the Expectimax AI Search Algorithm. My motivation for this project is my prior interest in the game 2048. I have managed to reach a tile of 8192 with a loosely defined strategy in the past and have long wondered what would be possible with a better-defined strategy.

There are two ways I intend to extend this project beyond previous attempts at playing 2048. Firstly, I intend to make the size of the grid variable. The algorithm should be able to solve smaller, larger, and rectangular 2048 games. For larger puzzles either performance or effectivity will be sacrificed, unless the algorithm can be optimised.

# Introduction

I intend to do this project in Java using JavaFX for the interface. This is because Java is currently my most familiar Object Orientated Language and I have experience with JavaFX.

2048 is a puzzle game featuring a 4x4 grid and sliding tiles. All the tiles can be moved, at the same time, in any of the four directions (up, down, left, and right). Each tile has a power of 2 on and if two collide they merge add their values are added together creating the next power of two. After the grid tiles slide a new tile is placed in a random free cell, containing a 2 or a 4. The goal of this game is to get a 2048 tile however the game can be continued past this point. The game ends when there are no free cells remaining (Cirulli, 2014).

Expectimax is an algorithm that can be used to solve a game of 2048 (Yun, Wenqi, & Yicheng, 2016). It is an adapted version of the min-max algorithm. The typical min-max algorithm is designed for playing two-player games, where both players are rational. One player aims to maximize the score, the second player tries to minimise the score (Russell & Norvig, 1995, pp. 161-167). In Expectimax the second player makes random decisions the first player must then maximize the expected score (Russell & Norvig, 1995, pp. 177-180). Expectimax can then be applied to 2048 by having the first player maximise the expected score while the second player places the tiles in the grid randomly.

One of the things I intend to do is make the game scalable. This will mean that I need to adapt the heuristics to work on different-sized grids. Adapting the heuristics for rectangular grids will be partially difficult. In a square game, it will be likely that the same strategy can be scaled up or down, but more thought will be required in a rectangular grid.

Another major thing to consider is optimisation. Typically, in the min-max algorithm, a technique called alpha-beta pruning is used to reduce the size of the tree, by cutting away unnecessary branches (Russell & Norvig, 1995, pp. 168-169). This allows you to eliminate some decisions that will never be made without looking at them. Pruning in the Expectimax algorithm is typically considered to be impossible without additional information (Simic, 2021). I would like to figure out if any form of pruning is possible on an Expectimax tree in 2048. based on knowledge about the game 2048.

# Timeline

## Term 1

* Week 1: Reading about Expectimax and 2048
  + Solving 2048 with Expectimax
    - (Bilal, Wahab, & Javed, 2018)
    - (Yun, Wenqi, & Yicheng, 2016)
  + General 2048
    - (Cirulli, 2014)
    - (2048 (Wikipedia))
  + Minmax Algorithm and Expectimax
    - (Russell & Norvig, 1995)
    - (Simic, 2021)
* Week 2 - 3: Implement a decision tree data structure capable of running the Expectimax algorithm.
  + Nodes do not need to be specific for 2048.
  + Should be able to get a score from leaf nodes and calculate a score for other nodes.
  + Weighted edges for edges between
* Week 4 – 5: Implement a resizable 2048 game with a simple user interface.
  + Allow user to input size of grid and start a new game.
  + Random initial setup for game.
  + Keep track of score
* Week 6 - 8: Generate a depth-limited decision tree from any 2048 game.
  + Generate list of possible moves from a 2048 game
  + Generate list of possible states after a 2048 move
  + Recursively repeated this until the depth limit has been reduced.
  + (Yun, Wenqi, & Yicheng, 2016)
* Week 9: Implement working heuristics for traditional 2048 game
  + Generate heuristic score representing how
  + (Yun, Wenqi, & Yicheng, 2016)
* Week 10 – 11: Finish off and prepare for the Interim report and presentation.
  + (Example Report)

## Term 2

* Week 1 - 2: Create rules for generating any square heuristic.
  + I intend to try to find a pattern between the heuristics and create a general rule for creating one.
  + It is possible I will have to make rules for generating heuristics if I cannot complete generalise them.
* Week 3: Create rectangular heuristics of various sizes
  + To help understand potential rules for working out rectangular games.
  + I will try at least sizes:
    - 4x5 (slightly bigger)
    - 4x3 (slightly smaller)
    - 9x3 (extreme ratio)
* Week 4 – 5: Try and find and implement general rules to generate any size of heuristic.
* Week 6 - 8: Investigate the possibility of pruning the tree.
  + I would like to know if I can eliminate any possible moves in advance of computing them.
  + It is possible that enough knowledge of the tree can be calculated from the state the game is in.
  + (Simic, 2021)
* Week 8-9: Collect Statistics on how the project is performing across a variety of problems and configurations.
* Week 10 – 11 Finalise final report.

# Early Deliverables

1. Proof of concept: decision tree.
2. Proof of concept: program: Simple Expectimax example.
3. Proof of concept: 2048 game.
4. Proof of concept: solving a 2x2 2048 with Expectimax.
5. Proof of concept playing 2x2 2048 with a simple heuristic
6. Report: Design Patterns for AI and Search.
7. Report: Techniques used by human solvers and previous automated solvers.
8. Report: User interface design for solver
9. Report: Complexity, NP hardness and big O notation.

# Final Deliverables

1. The final program will be fully object-oriented with a full implementation life cycle using modern software engineering principles, written in java.
2. The final program will theoretically be able to play any 2048 game, though eventually, performance issues will limit the feasibility.
3. The final program will have a user interface capable of keeping track of statistics about the algorithms, an easy way to create new puzzles (of a specified size) as well as a solve button.
4. The report will describe the practicality and (if possible) the effectiveness of the pruning.
5. The report will describe how the heuristics have been generalised to support more 2048 games.
6. The report will describe the software engineering process involved in developing the program.
7. The report will describe interesting algorithms and programming techniques used (including Expectimax) on the project.
8. The report will discuss the implementation and performance of the decision tree.

# Risk Assessment

* I may fail to get the Expectimax algorithm working in time, as I have never written a similar tree-based algorithm. To mitigate this risk, I will be creating a proof-of-concept program early on, if I am unable to complete this I will try a simpler algorithm which only models one move.

This risk is unlikely as there are plenty of resources available to help with the implementation of this algorithm.

* There is a high risk that I will not be able to prune the Expectimax tree, as I am not sure if it is even possible. In this event I can’t do this optimisation I will sacrifice the effectivity of the algorithm, partially on bigger projects to make it practical to use. I will also investigate the possibility of other optimisations.
* There is a risk that I will have some difficulty adapting the heuristics to different sizes of 2048 games. If these heuristics cannot be adapted, I will create heuristics designed for a more limited range of grids that are only used when appropriate. This risk is practically high for grids with extreme rations e.g., 2x9 where I fear they may be no effective strategies.
* There is a risk that I will fall behind by spending too long on the user interface. To reduce the risk of this I will be using JavaFX and Gluon Scene Builder. JavaFX is a library for creating user interfaces that I have some experience with, and Scene Builder is a graphical tool that can be used to quickly create user interfaces. This will reduce the workload in creating the interface.

# References

2048 (Wikipedia). (n.d.). Retrieved 31 9, 2022, from https://en.wikipedia.org/wiki/2048\_(video\_game)

Provided some basic information about the game 2048 and a link to the original source code. Also has a short section on human strategy.

Bilal, M., Wahab, A., & Javed, H. (2018). 2048 Game Using Expectimax. Retrieved 09 20, 2022, from https://github.com/Wahab16/2048-Game-Using-Expectimax

Example of a 2048 game that is solved with Expectimax. The key interesting feature of this project is that it is capable of regularly reaching 8192 with a short delay between moves that are to its use of a dynamic depth in the decision tree.

Cirulli, G. (2014). 2048 (Github). *2048*. Retrieved 8 24, 2022, from https://github.com/gabrielecirulli/2048

The source of 2048. Contains a lot of information about the game in the code and running example. This will be particularly useful when it comes to understanding specific details such as probabilities.

Russell, S., & Norvig, P. (1995). Artifical Intelligence A Modern Approch. In *Artifical Intelligence A Modern Approch (Third Addition).* Prentice Hall.

A Book which contains a lot of information on adversarial searches such as min-max and Expectimax.

Simic, M. (2021). Expectimax Search Algorithm. Retrieved 10 1, 2021, from https://www.baeldung.com/cs/expectimax-search

An article that simply but clearly explains the Expectimax algorithm and the feasibility of pruning it.

Yun, N., Wenqi, H., & Yicheng, A. (2016). *AI Plays 2048.* Retrieved 26/09/2022, from http://cs229.stanford.edu/proj2016/report/NieHouAn-AIPlays2048-report.pdf

A report describing solving 2048 with the Expectimax algorithm. This was chosen for its clear documentation rather than the optimisations.