Applying Various Search Methods to the Game 2048

Connor Chang*
chan2092@umn.edu

Abstract

Various distinct methods of search rated in the context of the popular game 2048. Minimax search contrasted with Expectimax search in terms of efficiency and quality of play. Several heuristic utility estimations and combinations thereof compared in service of finding the most promising results for non-terminal search of a limited depth.

1 Introduction

The game 2048 was released to positive reception in 2014 by Gabriele Cirulli. Likely due to the game's mathematical yet nontrivial nature, 2048 has since been a target for several researchers and hobbyists in artificial intelligence. This report is preceded by a few notable experimental examples of agents made for playing the game, and continues the research that gave them rise.

As in other applications of artificial intelligence, the central problem in achieving high-quality artificial play of 2048 concerns determining appropriate methodology. A number of choices are available to creators of game-playing artificial intelligence, from search algorithms to the evaluation functions that inform them. Some context-specific aspects of certain games or search environments limit choices in these categories.

In the case of 2048, a new 2 or 4 tile is put on the board at random after each move where there is a space left [1]. The new tile will have the value 2 90% of the time, with the value being 4 in the other 10% of cases. Due to this random behavior, state can be represented as a graph of player choice and random chance nodes, and so the game is referred to as stochastic. Several algorithmic methods of playing stochastic games exist, but these are generally intended for use in two-player games rather than single-player games like 2048. Some small amount of adaptation must be applied in most cases to generalize two-player algorithms to the single-player case.

In form factor, the game is similar to the 15-puzzle, however a primary difference is that multiple pieces on the board move with one player input. This distinction leads to scenarios where evaluation points are gained and lost in a single move. The relation between board manipulation and success is not always intuitive. Another difference is in each game's finished state. Whereas the goal in solving the 15-puzzle is to reach a single correct permutation of positions, the goal of 2048 is to create a single tile with a value of 2048. Many solutions are possible from any starting state, and it's entirely possible for up to four end states to be reached from a single penultimate state [4].

*Generative AI was not used in the creation of this report.

For these reasons and more, this problem's solution is worthy of study.

2 Background Work

A brief study was done by Rodgers and Levine in 2014 which aimed to reveal the most promising strategy for searching in 2048 games [3]. The study examines Monte Carlo Tree Search as well as Averaged Depth-Limited Search for this distinction. The authors noted that, in testing, many game trees for 2048 were symmetrical. Due to this, the strategy of Monte Carlo Tree Search, which typically benefits through pruning asymmetrical trees, was deemed less than optimal for playing 2048. Averaged Depth-Limited Search, however, performed well in terms of scoring and efficiency, outplaying Monte Carlo Tree Search as well as scoring similarly to an Expectimax implementation by Xiao.

The strategy of Expectimax was notably applied to 2048 by Xiao in 2014 [6]. Expectimax, a single-player adaptation on Expectiminimax, was likely chosen for experimentation due to its suitability to the game, being one of the few algorithms well-suited to modeling the nature of the game. Xiao achieved success with this approach, owing in large part to his extensive optimization of the search, allowing many states to be assessed in little time. This Expectimax implementation achieved scores in the range of 124024 to 794076 over 100 test runs, never failing to reach the 2048 tile and reaching the 32768 tile in 36% of runs.

There are many choices when it comes to heuristics by which 2048 state goodness may be evaluated. An exploration of a few of these was done in 2019 by Kohler, et al., who sought to identify heuristics that were simple enough to be used in human gameplay while being effective in either humans or agents attaining high scores [2]. This particular study examined these heuristics individually as well as in composition with one another, evaluating which composition worked best. The conclusion reached was that the greatest outcomes, in order, came from evaluating states on the number of empty squares present on the board, then with greater measured monotonicity of tile values, then with greater number of tiles of the same value, and finally with the official score of the game. This study was done using a unique single-depth best-first search algorithm in order to replicate human behavior, and no other, more general in-depth studies have been done on the topic since.

One effective performance optimization available to most stochastic games is the use of transposition tables computed

before run time. This technique was used by Xiao's aforementioned Expectimax implementation, allowing search termination when a previously seen state was reached [6]. A more general analysis of this optimization technique was carried out by Veness and Blair in 2007 [5]. This paper examined the existing transposition table in the new context of games with chance nodes. In particular, transposition tables were retrofitted into the Star2 algorithm's pruning logic, allowing the elementary technique of pruning to be done in a stochastic setting. The result of this innovation was a decrease by 37% of depth-13 nodes searched in the game of Dice when compared to unaided Star2. This study also showed a decrease by 86% over Expectiminimax, which is typically a brute-force algorithm.

3 Problem Statement

This work aims to analyze, in a moderate depth, a few previously researched methodologies of solving the game 2048. In doing so, an optimal weighting of heuristic evaluation functions is sought, as well as the search algorithm of more merit between two which have previously been successfully employed in the context.

4 Methods

The following section contains information specific to this work's implementation of a 2048 solver.

4.1 Search Algorithms

The solver can play using two different search algorithms, Minimax and Expectimax.

4.1.1 Minimax. Minimax is a standard algorithm used in non-stochastic, two-player games. When solving using Minimax, the solver assumes, not strictly correctly, that the game's random tile regeneration will act as an adversary, holding that it chooses the positions that most greatly decrease the player's utility going forward. The result of these assumptions is the freedom to cast chance nodes as Min nodes. While this assumption does not hold, it is a useful abstraction which allows the implementation to include the alpha-beta pruning subroutine, greatly increasing the efficiency of this implementation.

4.1.2 Expectimax. Expectimax is an adaptation of the Expectiminimax algorithm which simply removes the Min nodes, leaving only Max and Chance nodes in graphs of state space. Expectimax, as such, is a more correct model for the state in 2048. Instead of holding the assumption of a zero-sum interchange between the game and player, this model allows chance nodes to remain chance nodes, and to have these evaluated by taking the mean of their branches' utilities weighed by each branch's probability of occurrence. The correctness Expectimax reintroduces to solving 2048 is balanced by its inefficiency as a brute-force expansion of every chance node. This implementation, in the service of being able to run at similar depths to Minimax, employs a probability cutoff for chance branches, where branches whose probability of occurrence is less than 5% are not considered further and are instead evaluated by their heuristic value.

4.2 Heuristic Functions

This implementation includes six distinct heuristics gathered from varying preceding studies.

4.2.1 Monotonicity. This heuristic seeks to measure how steadily tile values increase along rows and columns in either direction. A high score in Monotonicity suggests a board displaying gentle gradients of tile values. The code for this heuristic was borrowed directly from pseudocode in Kohler et al. [2].

4.2.2 Free Spaces. This heuristic encodes the fraction of the board's cells which are not occupied by tiles. Previous work has commented on the efficacy of encouraging greater amounts of free space in 2048 states due to its likelihood of allowing greater freedom of tile motion [2, 6].

4.2.3 Uniformity. This heuristic rewards repeated tile values on the board. The presence of multiple tiles of the same value in one state suggests the possibility of merges in upcoming moves, and greater tile values and scores in turn. The evaluation is then weighted by the tile's value. The formula for uniformity is due to Kohler et al. [2].

4.2.4 Score. This heuristic rewards higher scores in the game. In doing so, merges are indirectly encouraged to happen.

4.2.5 Max Tile. This heuristic rewards states based on the value of the greatest-valued tile on the board.

4.2.6 Large Tiles on Edges. This heuristic evaluates states positively if they have large-valued tiles on the edges of the cell grid, and the effect compounds for tiles on two edges. The evaluation is then weighted by the tile's value.

5 Experiment

5.1 Procedure

The general solver built for this research will be run with varying search algorithms and combinations of heuristics. Each specific combination being experimented with will be run 10 times with a depth of 2, and these results will be compiled and analyzed.

5.2 Results

The results of each algorithm with every individual heuristic function are displayed in the following two tables.

Minimax Max.	Heuristic Max. Tile	Avg.	Std. Dev.	Median
Score	13787.2	2596.58	14550	15896
1024				
Free Spaces	9918.9	3939.94	10046	16088
1024				
Large Edges	7477.6	3844.63	6452	16128
1024				
Monotonicity	2100.4	913.88	1950	3500
256				
Max. Tile	6986.8	3983.14	6438	14328
1024				
Uniformity	8538.4	2891.94	7278	14384
1024				

	Expectimax				
Heuristic	Avg.	Std. Dev.	Median	Max.	Max. Tile
Score	21374.8	6832.46	19712	32236	2048
Free Spaces	13299.6	2826.78	14128	16256	1024
Large Edges	6748.4	3791.73	5370	14644	1024
Monotonicity	3047.2	1661.13	2970	5460	512
Max. Tile	6384	1290.35	6692	7916	512
Uniformity	3770	2329.14	3018	8236	512

6 Analysis

The following section contains analysis based upon the previously appended figures.

6.1 Search Algorithms

The collected experimental data suggests that both Minimax and Expectimax achieve similar results across all of the heuristics they were tested with. The variance is not extreme, and would likely disappear with greater number of test runs, but this would bear out to further testing.

6.2 Heuristic Functions

The data suggests that most heuristics are not self-sufficient in commonly reaching the 2048 tile, as only the the Score heuristic was capable of winning in this way. The likeliest answer is that more than one heuristic may be combined to create a stronger evaluation function.

7 Conclusion

The research carried out for this report showed no clear answer as to which search algorithm performs better in the context of searching 2048 state space. Both Minimax and Expectimax performed the same within significant tolerance.

A clearer answer was shown in which heuristic functions are more useful in evaluating non-terminal 2048 states. The results of testing suggest that evaluating based on the game's scoring protocol is the most accurate singleton heuristic evaluation, followed by the number of free spaces on the board.

8 Future Work

In future work, it would be worthwhile to examine in more detail the differences between the heuristics used here. This would include a different research approach, likely involving many more data points.

Another interesting point of future study may be the possible combinations of heuristic evaluators, which might possibly lead to better solving.

References

[1] Madhuparna Das and Goutam Paul. 2019. Analysis of the game '2048' and its generalization in higher dimensions. (2019). https://doi.org/10.48550/arXiv.1804.07393

[2] Iris Kohler, Theresa Migler, and Foaad Khosmood. 2019. Composition of basic heuristics for the game 2048. In *Proceedings of the 14th International Conference on the Foundations of Digital Games* (San Luis Obispo, California, USA) *(FDG '19)*. Association for Computing Machinery, New York, NY, USA, Article 52, 5 pages. https://doi.org/10.1145/3337722.3341838

[3] Philip Rodgers and John Levine. 2014. An investigation into 2048 AI strategies. In *IEEE Conference on Computational Intelligence and Games*. IEEE, Dortmund, Germany, 1–2. https://doi.org/10.1109/CIG.2014.6932920

[4] John Sijitsma. 2020. *Creating a Formal Model of the Game 2048*. Master's thesis. https://www.cs.ru.nl/bachelors-theses/2020/Johan_Sijtsma__4793676___Creating_a_Formal_Model_of_the_Game_2048.pdf s4793676.

[5] Joel Veness and Alan Blair. 2007. Effective Use of Transposition Tables in Stochastic Game Tree Search. In *2007 IEEE Symposium on Computational Intelligence and Games*. 112–116. https://doi.org/10.1109/CIG.2007.368086

[6] Robert Xiao. 2014. Writing a 2048 AI. https://www.robertxiao.ca/hacking/2048-ai/