Composition of Basic Heuristics in Solvers of the Game 2048

Iris Kohler, Foaad Khosmood, and Theresa Migler
California Polytechnic State University, Department of Computer Science and Software Engineering

2048

2048 is an incredibly popular game created by Gabrielle Cirulli and released on March 9, 2014. It and various clones were downloaded tens of millions of times soon after release. It is a tile-based game with random elements, where all tiles have some power of 2 on them. Players match tiles with the same value to get a tile with the next power of 2. The goal is to reach a tile with the value of 2048, though the game can be played after reaching the value.

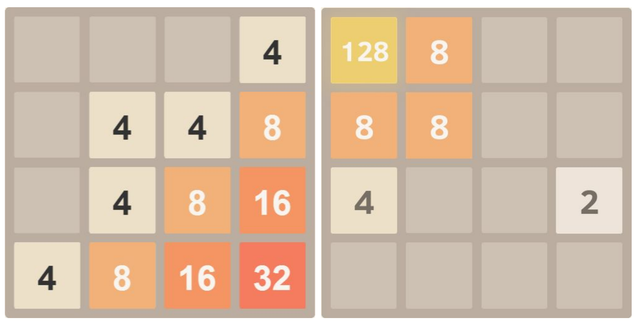

Figure 1: On the left, a board with a high monotonicity score. On the right, a board with high uniformity and empty scores.

Heuristics

Our research focuses on solving 2048 using heuristic functions that are simple enough to be replicated by a human player. Most non-random heuristics often have multiple moves leading to the same heuristic value, so we also explore the use of a secondary heuristic to break ties. This may lead to another tie, requiring a third heuristic, and so on. We call these *non-terminal heuristics.*

The *greedy* heuristic prefers moves that produce the highest scores.

The *empty* heuristic prefers moves that create the most empty spaces.

The *uniformity* heuristic prefers moves that produce the most tiles with the same value as other tiles on the board.

The *monotonicity* heuristic prefers moves that push higher value tiles into the corners, with values decreasing down rows and columns.

In the case that all heuristics produce ties, a random choice can be used to break the tie. We define each sequence of basic heuristic functions as a “composition.”

Input: G - a game
Output: The game's monotonicity score
1: **procedure** ScoreMonotonicity(G)
2: $best \leftarrow -1$
3: **for** $i \leftarrow 1, 4$ **do**
4: $current \leftarrow 0$
5: **for** $row \leftarrow 0, 3$ **do**
6: **for** $col \leftarrow 0, 2$ **do**
7: **if** $G[row][col] \geq G[row][col+1]$ **then**
8: $current \leftarrow current + 1$
9: **end if**
10: **end for**
11: **end for**
12: **for** $col \leftarrow 0, 3$ **do**
13: **for** $row \leftarrow 0, 2$ **do**
14: **if** $G[row][col] \geq G[row+1][col]$ **then**
15: $current \leftarrow current + 1$
16: **end if**
17: **end for**
18: **end for**
19: **if** $current > best$ **then**
20: $best \leftarrow current$
21: **end if**
22: Rotate the board 90 degrees clockwise
23: **end for**
24: **return** $best$
25: **end procedure**

Figure 2: The algorithm for determining monotonicity

Strategies

We define a strategy to be a composition of heuristics such that the following properties hold:

1. A strategy is a composition of zero or more non-terminal heuristics followed by a random choice.
2. No heuristics in the strategy are repeated.
3. The random choice is always the last heuristic in the strategy.

We attempt to find the best-performing strategy.

Input: G - a game, M - a set of moves
Output: The best moves according to the specific evaluator
1: **procedure** EvaluateGame(G, M)
2: $bestScore \leftarrow -1$
3: $bestMoves \leftarrow \emptyset$
4: **for all** $move \in M$ **do**
5: $G' \leftarrow G$
6: apply $move$ to G'
7: $curScore \leftarrow$ the score given by the evaluator's specific criteria
8: **if** $curScore > bestScore$ **then**
9: $bestScore \leftarrow curScore$
10: $bestMoves \leftarrow \{move\}$
11: **else if** $curScore = bestScore$ **then**
12: $bestMoves \leftarrow bestMoves \cup \{move\}$
13: **end if**
14: **end for**
15: **return** $bestMoves$
16: **end procedure**

Figure 3: The algorithm for choosing a move based on a strategy

Experiments

We built a driver to take a strategy and run it on a simulated 2048 board. We ran each strategy 1,000 times, taking their mean, standard deviation, median, and max scores. The results are shown in Figure 4 and Table 2.

Table 1: A key for all results and the performance of individual heuristics alone

Symbol	Evaluator	Mean	Std Dev	Median	Max
r	Random	1093.876	539.7451701	1058	4348
g	Greedy	1090.644	530.2134224	1044	3088
m	Monotonicity	3177.172	1196.254349	3218	7636
u	Uniformity	1079.356	516.6318469	1038	2996
e	Empty	1948.132	878.6316626	1652	6336

Conclusions

Strategies that select first for high numbers of empty spaces with high monotonicity as a secondary tiebreaker perform the best over 1,000 games. Any heuristics that come after do not seem to make much difference, as long as empty and monotonicity are the first and second heuristics in the composition respectively.

Table 2. The performances of all strategies sorted in decreasing order by mean

Evals	Mean	Std Dev	Median	Max
emr	3947.828	1883.284192	3442	12488
gemr	3946.132	1857.778333	3444	12248
emugr	3944.404	1861.70881	3480	12688
gemur	3930.656	1824.018777	3510	12872
geumr	3925.008	1821.796057	3458	12264
eumr	3924.292	1820.369262	3516	13248
uegmr	3904.228	1836.624947	3462	12292
guemr	3895.64	1833.612905	3444	12220
emgur	3883.58	1820.294095	3448	12376
eugmr	3879.128	1812.957992	3474	12356
emgr	3874.148	1810.141301	3436	12136
emur	3873.476	1828.501309	3444	12404
egumr	3868.248	1754.354847	3452	12476
uemr	3830.04	1784.919673	3442	11684
eumgr	3821.28	1805.720349	3408	12128
uemgr	3817.144	1843.801752	3422	15444
ugemr	3815.972	1846.260129	3382	13688
egmr	3769.508	1762.662472	3398	11968
egmur	3764.664	1728.744888	3418	12352
megr	3687.952	1468.508783	3494	11448
ugmer	3682.44	1513.566088	3448	10504
umger	3678.352	1535.196896	3472	10648
umegr	3676.16	1431.604897	3484	11100
umer	3672.664	1447.026382	3462	13396
mger	3665.756	1458.024221	3470	8864
gmer	3662.668	1502.474834	3490	10712
mer	3637.388	1528.651613	3404	10840
megur	3625.4	1471.250393	3400	11364
meur	3623.864	1429.232739	3400	11788
muger	3623.132	1490.673072	3444	12324
gmeur	3620.264	1429.395361	3408	9336
gmuer	3618.04	1464.73884	3416	11528
mgeur	3617.696	1464.641238	3422	10504
muegr	3612.708	1381.247365	3464	7776
mguer	3604.02	1493.090684	3396	12172
gumer	3596.568	1475.615754	3400	8752
muer	3582.2	1435.47073	3380	9084
meugr	3579.928	1520.070186	3406	13100
umr	3229.72	1229.608101	3220	11212
mgr	3209.704	1278.374484	3200	8496
gmr	3209.088	1246.911346	3180	9004
gmur	3197.608	1227.234174	3198	8656
umgr	3196.72	1212.559037	3184	7500
mgur	3189.768	1277.70139	3168	13708
gumr	3184.816	1265.142221	3194	7776
mur	3149.556	1196.947034	3172	8148
mugr	3144.456	1176.338475	3190	7800
ugmr	3125.172	1217.673711	3108	8384
ger	1967.7	909.2398704	1662	6416
eugr	1951.296	914.2897661	1610	5572
egr	1943.604	911.8011292	1628	6280
uger	1941.028	900.1908149	1626	6092
guer	1934.952	900.8464052	1616	6392
uegr	1927.008	885.6564977	1586	5508
geur	1910.176	900.5592224	1586	5972
eur	1889.224	874.0607426	1578	5392
egur	1872.296	866.901208	1548	6160
uer	1835.372	860.0558898	1510	5584
ugr	1100.708	554.9087373	1040	3160
gur	1083.056	540.6129899	1040	4308

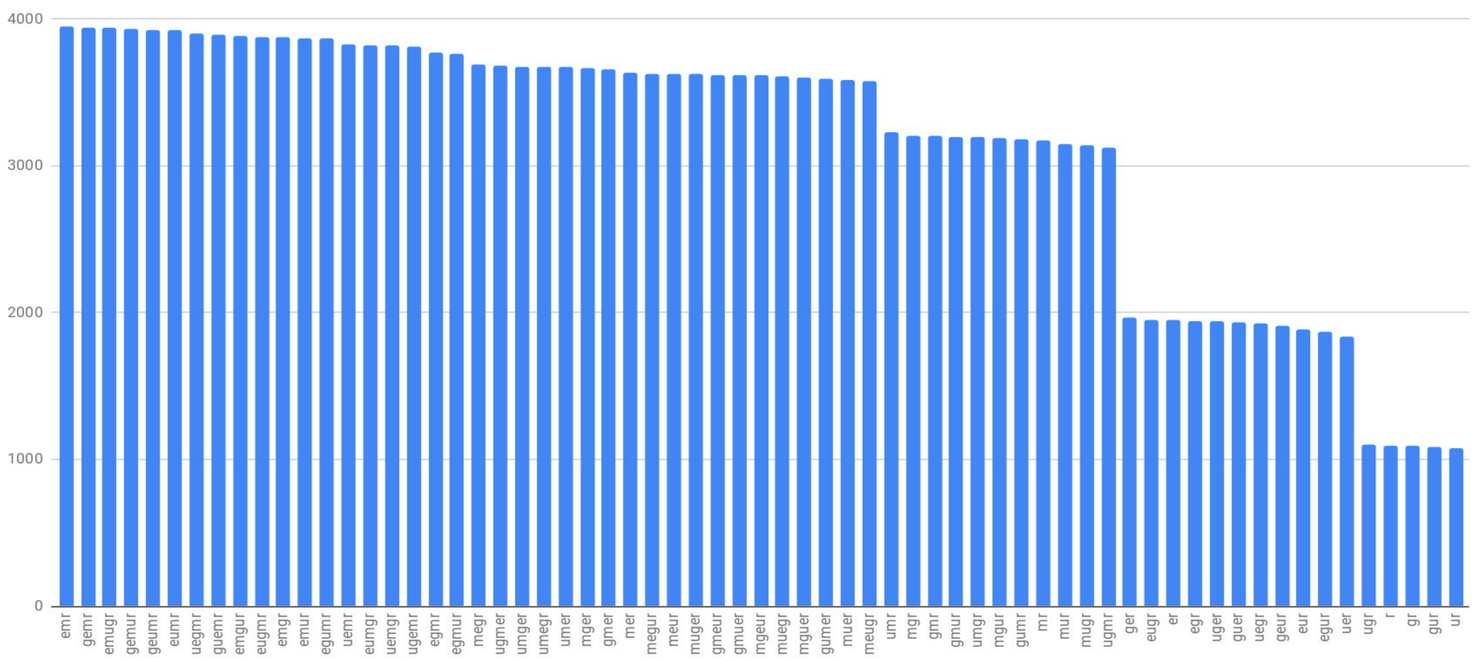

Figure 4: The graph of each strategy’s mean performance over 1,000 games sorted in decreasing order

References

- Abdelkader, A., Acharya, A., & Dasler, P. (2016). 2048 without new tiles is still hard. In 8th international conference on fun with algorithms (FUN 2016). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.
- Madhuparna Das and Goutam Paul. 2018. Analysis of the Game "2048" and its Generalization in Higher Dimensions. CoRR abs/1804.07393 (2018). arXiv:1804.07393 http://arxiv.org/abs/1804.0739
- David Eppstein. 2018. Making Change in 2048. CoRR abs/1804.07396 (2018).
- Wojciech Jaśkowski. 2016. Mastering 2048 with Delayed Temporal Coherence Learning, Multi-State Weight Promotion, Redundant Encoding and Carousel Shaping. CoRR abs/1604.05085 (2016). arXiv:1604.05085 http://arxiv.org/abs/1604.05085
- Stefan Langerman and Yushi Uno. 2018. Threes!, Fives, 1024!, and 2048 are hard. Theoretical Computer Science 748 (2018), 17 – 27. https://doi.org/10.1016/j.tcs. 2018.03.018 FUN with Algorithms
- K. Matsuzaki. 2016. Systematic selection of N-tuple networks with consideration of interinfluence for game 2048. In 2016 Conference on Technologies and Applications of Artificial Intelligence (TAAI). 186–193. https://doi.org/10.1109/TAAI.2016.7880154
- Rahul Mehta. 2014. 2048 is (PSPACE) Hard, but Sometimes Easy. Electronic Colloquium on Computational Complexity (ECCC) 21 (2014), 116.
- Kazuto Oka and Kiminori Matsuzaki. 2016. Systematic Selection of N-Tuple Networks for 2048. In Computers and Games, Aske Plaat, Walter Kosters, and Jaap van den Herik (Eds.). Springer International Publishing, Cham, 81–92
- P. Rodgers and J. Levine. 2014. An investigation into 2048 AI strategies. In 2014 IEEE Conference on Computational Intelligence and Games. 1–2. https://doi.org/10.1109/CIG.2014.6932920
- M. Szubert and W. Jaśkowski. 2014. Temporal difference learning of N-tuple networks for the game 2048. In 2014 IEEE Conference on Computational Intelligence and Games. 1–8. https://doi.org/10.1109/CIG.2014.6932907
- I-Chen Wu, Kun-Hao Yeh, Chao-Chin Liang, Chia-Chuan Chang, and Han Chiang. 2014. Multi-Stage Temporal Difference Learning for 2048. In Technologies and Applications of Artificial Intelligence, Shin-Ming Cheng and Min-Yuh Day (Eds.). Springer International Publishing, Cham, 366–378.
- Robert Xiao. 2014. What is the optimal algorithm for the game 2048? Retrieved September 14, 2018 from https://stackoverflow.com/a/22498940
- Robert Xiao, Wouter Vermaelen, and Petr MorÃąvek. 2018. 2048-AI. Retrieved
- September 14, 2018 from https://github.com/nneonneo/2048-ai

Acknowledgements

The 2048 simulator we used was partially based off of John Patterson’s work for his senior project. This project was part of Cal Poly College of Engineering’s Summer Undergraduate Research Program in 2018. 2048 was created by Gabrielle Cirulli.