Analysis of Scanline and Minimum Entropy Selection Heuristics in Model Synthesis and Wave Function Collapse

Lucas Harvey and Gautam Srivastava
Dept. of Math and Computer Science, Brandon University, Canada
{harveylg42, srivastavag}@brandonu.ca

Abstract* – In this paper, we analyze two versions of a texture synthesis algorithm, study the cases in which they fail to produce a successful result and present modifications that could be made to lessen their rates of failure. This algorithm, Model Synthesis, and its variation Wave Function Collapse are designed to take in a small sample input image, or set of image constraints, and produce a larger pseudorandom output image in which every region of the output image is locally similar to an element of the input image. Both versions of the algorithm accomplish this task by considering their output image as a grid of cells with each cell initially in a superposition of all possibilities for itself and resolving cells one by one until all cells have been resolved from their superposition to a fixed value. One of the key differences between the two versions of the algorithm is the order in which cells are selected to be resolved, one simply selects in a scanline order, while the other resolves first those cells that have the minimum entropy, and thus which we can be most certain of their eventual state. For many inputs, the minimum entropy model reaches a state in which its output is not consistent with the input and thus fails, while the scanline model does not. This paper looks at the cases in which this occurs and concludes that this is often caused by the minimum entropy model creating regions of elevated constraints in its solution. Finally, it presents a possible alteration to the algorithm which allows a minimum entropy model to avoid this manner of failure among a subset of test cases.

Keywords—Model Synthesis, Wave Function Collapse, Texture Synthesis, Procedural Generation, Constraint Solving.

I. Introduction

When approaching the task of procedurally generating images, several algorithms and methods may be used. One such family of algorithm Texture Syntheses [7] of which Model Synthesis [3], created by Paul Merrel, and its derivative algorithm Wave Function Collapse [1] are versions. Model Synthesis was designed to take as an input a set of rules for creating an output image, and then produce that output image to conform to those rules with a level of random variation, to procedurally produce unique but similar results. In doing this, it creates a constraint satisfaction problem from the texture synthesis problem [8]. Maxim Gumin, who created a version of the algorithm titled Wave Function Collapse expanded this to be able to take as input an image and produce as output an image where every region of the output image is similar to a region on the input image [1][9]. These algorithms can fill a key role in certain digital media applications where producing semi-random results that conform to certain established requirements is critical, [2] such as procedural content generation, which has been discussed amongst academics in the past [12][13]. Fig. 1 shows a larger output image created from a smaller input image.

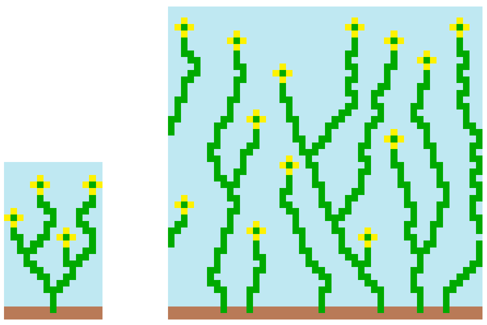
Fig. 1. Wave Function Collapse example

These algorithms function for outputs that can be broken down into a grid of two- or three-dimensional cells which themselves can be selected to be one of many possible outcomes. The algorithm then functions by initially setting all cells to be in a superposition of all possible outcomes and gradually selecting a cell, eliminating all but one remaining possibility for that cell, and then propagating any additional constraints added due to the determining of that cell to other surrounding cells in a cascading series of eliminations. This continues until one of two states occurs, either all cells have resolved themselves to a single possible outcome, or any one cell has eliminated all of its possible outcomes. In the former case, the algorithm is complete, in the latter, the algorithm has reached a conflict and must either be restarted or have its error resolved. These cases of conflict occur when there is a set of rules that collectively eliminate all remaining possibilities for a given cell [4].

The two key differences that differentiate the Model Synthesis and Wave Function Collapse algorithms, the first of which is that Model Synthesis uses a particular method of error handling, modifying by parts, that breaks down any individual problem into a series of sub-problems. Wave Function Collapse, however, typically simply restarts on each failure, though it is not incompatible with some forms of error handling. Each subproblem is much easier to solve, and thus complex inputs with a large chance of failure can be solved for large outputs [4]. However, for this paper, we are centered on the second difference between the two, that Model Synthesis resolves cells in a simple scanline order, while Wave Function Collapse operates in a minimum entropy order. Minimum entropy order will first select those cells that have the most certain outcome, which typically corresponds to those cells that have had the most options eliminated from their original superposition, with ties in entropy resolved randomly.

The performance, quality of output, and failure rates of the two algorithms are largely similar in many cases, however, there are several notable differences [5]. First, there are certain outputs in which Model Synthesis's output is skewed by the directionality of a Scanline order, as seen in Fig 2 where Model Synthesis and Wave Function Collapse were both used to compute the same tile set. Model Synthesis produced the image on the left, where the placement of cells is directionally influenced to be less dynamic. Secondly, there are many cases in which the failure rate, and thus the performance of Wave Function Collapse is significantly worse. This difference in performance was reported in [5] where Merrel presented a series of data points regarding Wave Function Collapse's greater failure rate yet left exploring the cause of this distinction to future work.

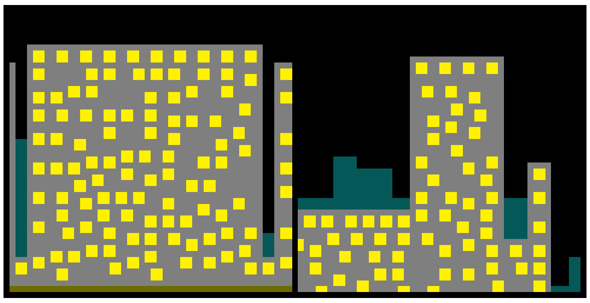
Fig. 2. Directional bias sample

In this work, we explore a cause of Wave Function Collapse's tendency for failure in certain cases when compared to Model synthesis and present a modification to the Wave Function Collapse process which could accommodate for this flaw.

II. PREVIOUS WORK

In 2007 Merrel presented a paper detailing the Model Synthesis algorithm [3], the texture synthesis algorithm which Wave Function Collapse is a variation of. It was originally presented using constraint rules, and then later improved to incorporate Gumin's overlapping image modification, which allows it to take a whole image as input. Merrel's thesis in 2009 further expanded on Model Synthesis [4].

In 2016 Gumin published Wave Function Collapse [1], a version of the Model Synthesis algorithm published by Merrel in 2009, which implemented an overlapping model for input generation which allows an image to be used directly as input, as well as implemented the minimum entropy heuristic as opposed to the scanline heuristic used in Model Synthesis. 'Wave Function Collapse' refers to the visual manner in which the algorithm propagates constraints and is not directly related to the concept in physics.

A 2017 paper by Isaac and Smith [2] analyzed the Model Synthesis and Wave function collapse algorithms to analyze them on several key heuristics, to establish the ideal parameters for the algorithm. One of these studied heuristics was the selection heuristic, the central focus of this paper. The results of the study indicated that both a scanline procedure and a minimum entropy model were similar in performance and failure rate, yet the scanline model used by Model Synthesis would introduce directional biases that may be undesirable. Thus, it advocated for the entropy-based model as it was similar in performance yet lacked certain directional biases for some inputs.

A 2021 paper by Merrel put Wave Function Collapse and Model Synthesis into comparison and found that Wave Function Collapse failed far more often for certain inputs [5]. Merrel attributed this to largely the use of the entropy heuristic, rather than the lack of modifying by parts to handle errors. However, Merrel left to establish why this occurs in future work.

In reviewing Isaac and Smith's 2017 paper, it can be seen that there is some advantage to be pursued in a synthesis technique that does not operate in scanline order, as certain inputs have a directional bias. However, it should be recognized that in many inputs, such as those presented in Merrel's 2021 report, directional bias does not meaningfully affect the results of the output. In reviewing Merrel's 2021 paper, it is clear that though Model Synthesis and Wave Function Collapse typically have similar performance metrics, there are certain cases where Wave Function Collapse performs far worse. Thus, it is necessary to find a manner in which the benefits of the two models can be combined to achieve results unlikely to reach errors with no directional biases in output.

III. BACKGROUND INFORMATION

In both the Model Synthesis and Wave Function Collapse algorithms, when a tile is selected to be resolved the same considerations are made. First, it randomly determines from the set of all non-eliminated possibilities a single option that it shall use and eliminates all others. This determination may give equal priority to each possibility or may use a weighted assignment depending on input parameters. Once all remaining possibilities are eliminated, it then begins the propagation step.

In the propagation step, the additional constraints that stem from this determination are propagated out to their neighbors, and from them to their neighbors, and so on. The purpose of this step is to ensure that the valid options for future cells remain consistent with those currently established constraints, to prevent the algorithm from removing all legal constraints from a cell and reaching a failure state. However, as noted in Merrel's thesis, maintaining perfect consistency is an NP-hard problem, and thus an approximation to this perfect consistency is used [4].

To achieve this imperfect approximation, the AC3 or AC4 algorithm is used [5][10][11], to propagate the constraints in a manner that approximates the perfect set of consistent options. This process operates by pathing, that is a certain value for a cell will not be eliminated provided there is a legal path of cells from the current cell to the intended cell that would permit that cell to hold the value in question. This concession to the performance of the algorithm is what allows failures to occur, and for certain inputs is far more prevalent with a scanline heuristic.

IV. METHOD

I used the source code provided publicly by Gumin, which has an inbuilt scanline and minimum entropy functionality. I then modified the code to save a copy of the image currently generated at each step, and to display not-yet selected cells as colored by their number of remaining options available to visualize the manner in which the constraints are propagated across results. I then ran the inputs that Merrel presented had a high difference in success rates between the two versions of the algorithm.

After this, I then reviewed large numbers of failure cases for the two algorithms to review the differences and possible causes of their failure. After reaching a hypothesis, I then implemented a modification to the minimum entropy algorithm to attempt to correct this perceived failing and finally attempted the test set again to monitor for the efficacy of the modification

V. Analysis of Data

The attached findings are the results achieved from running the inputs presented by Merrel in 2021 with the scanline and minimum entropy selection heuristics, without the use of modifying by parts. As addressed by Merrel, the data affirms that the cause of the failure rate for Wave Function Collapse is largely caused by the minimum entropy heuristic, rather than the lack of modifying by parts. Of note is that, for some inputs, the data shows that the scanline heuristic can't achieve a failed result even without modifying by parts, as evidenced by the data in Fig 3.

Tileset	Errors (Max 100 Trials)	
	Scanline	Min Entropy
Summer 100x100	0	100
Summer 200x200	0	100
Castle 100x100	0	100
Castle 200x200	0	100
Knots Dense 100x100	0	1
Knots Dense 200x200	0	100
Knots TE 100x100	0	100
Knots TE 200x200	0	100
Knots T 100x100	0	4
Knots T 200x200	0	100
Knots CE 100x100	0	23
Knots CE 200x200	0	100
Rooms 100x100	0	0
Rooms 200x200	0	32
RedDot200x200	0	100
Skew1 200x00	4	100
Cat 400x400	0	6

Fig. 3. Table of performance differences between MS and WFC

This data indicates that for certain inputs, the scanline heuristic never allows for a local region in the output to be formed in which the constraints are strict enough to eliminate all legal options for a cell. For instance, the input subset 'Knots TE' is a simple problem set in which all cells start in a superposition between five possible options, which are an empty space and a T intersection shape in each of four possible rotations, shown in Fig 4. The rules for placement forces empty spaces, such as those from empty cells and the missing direction of the T intersection to align with other empty spaces, and for the lines of T intersections to align with other T intersection lines.

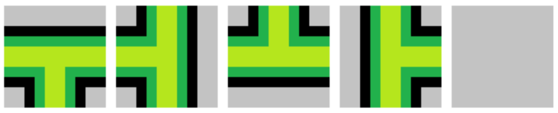
Fig. 4. Five possible cell evaluations for Knots TE

For a scanline selection heuristic, the Knots TE input set is a trivially easy problem. In Scanline procedures, it is typical that most cells will be resolved with two immediate adjacencies, the adjacency from the current cell, and the adjacency from the cell of the last completed row. This is shown in fig 5, where the next cell to evaluate in a left-to-right order always have two adjacent cells. The only instances in which scanline order will evaluate a node with more than two adjacencies is when the previous row propagated enough constraints onto a future row to cause it to resolve down to 1 remaining option before being selected by the scanline heuristic.

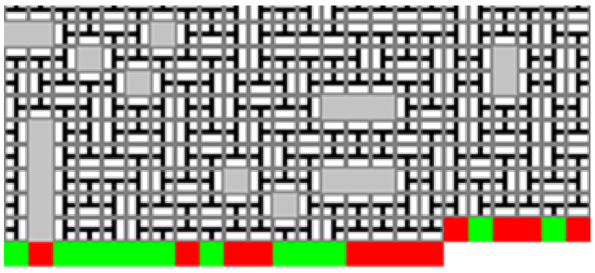
Fig. 5. Knots TE midway through execution.

In the scanline heuristic, with two immediate adjacencies the Knots TE model considers four possible scenarios, that both the adjacent cells are empty, that both have an incoming line, and that one does and the other does not for both incoming cells. For Knots TE, there is always a legal option for each possible case, and that option never cascades sufficiently to cause adjacent cells to be forcibly resolved. Thus, it is clear how the Knots TE input does not fail for the Scanline model, as it requires more than two adjacencies to fail, and Knots TE will only ever consider cells with 2 or fewer adjacencies. Note that this is untrue for the Knots TE input if the output image is configured to wrap such that the constraints of the first line act upon the last line, as this causes some cells to resolve with more than two adjacencies, which permits for failure.

In the minimum entropy model, however, it is possible to produce a circumstance in which a cell is resolved with more than two immediate adjacencies. Consider the case below, in which multiple adjacencies have occurred to cause the Knots TE input to fail, as minimum entropy simply selects a cell with the minimum current entropy and does not optimize for reducing the overall constraints of the output. This allows a series of minimum entropy satisfying choices to be made which causes a region of the output to begin to close in on itself, dramatically increasing the rate at which certain inputs fail. This is seen in Fig 6, which chose a region beginning to close in on itself, dramatically increasing the chance of failure.

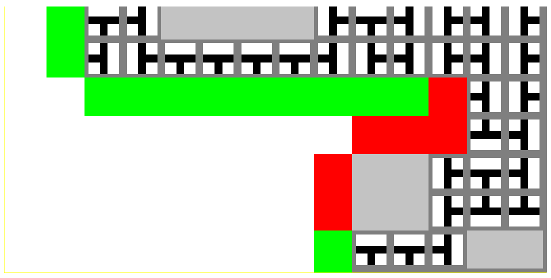

Fig. 6. An example of an elevated region of constraint.

VI. Proposed Solution

As an examination of many of the cases of failure has presented that failures often occur from regions of elevated constraints caused by the algorithm 'folding in' on itself and reaching states with many overlapping constraints, I shall now present an alteration that could be made to the algorithm to accommodate for this occurring, and accompanying data regarding the effect the alteration has on the failure rate of test cases.

The modified portion of the algorithm occurs during the cell selection process, where each cell is sequentially checked for its entropy, and the cell with the minimum entropy including a fixed random factor to resolve ties is selected to be resolved. For each cell whose entropy plus a random factor is the smallest yet found, an additional requirement must be satisfied before being accepted as the next cell to be resolved. First, the four cardinally adjacent cells are checked to see if they have resolved into a single cell. For each adjacent cell that has not resolved to a single value, the algorithm continues to check all the cells in that direction from the origin cell to see if a future cell is reached.

If all directions from the origin cell either have a resolved cell immediately adjacent to the cell in question or do not have any resolved cells from the origin cell to the edge of the output, then the cell is a valid placement. If this is not the case, the algorithm then evaluates whether there are any cells in the opposite direction of the direction checked such that resolving the origin would be resolving a cell between two cells. If this is the case, the cell is considered both valid and a priority cell, as it is part of a gap between two cells and must be resolved as quickly as possible, before the gap can be worsened. If not, then the cell is not currently a valid cell to resolve, as those cells between the origin cell and the resolved cell along its directional path must be resolved first.

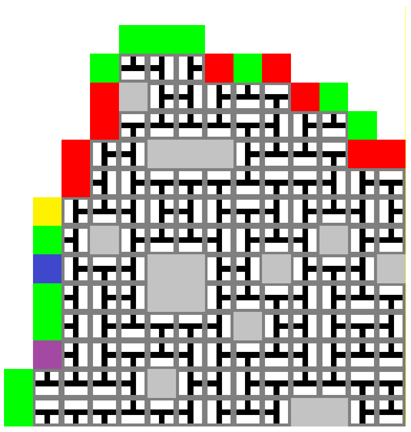

Fig. 7. Modified algorithm running example.

For example, in Fig 7 the purple cell would be a legal option to resolve next, as south and east are already resolved, and north and west have no cells anywhere along their paths. The blue and yellow cells are illegal options, as there are empty cells to their south, with a resolved cell along their path. However, if the yellow cell were already resolved, the blue cell would be legal and have priority to resolve this discrepancy.

Thus, the algorithm can avoid many cases where an elevated region of constraint would be created due to many adjacencies, and when such a region begins to form due to propagated constraints, the algorithm prioritizes resolving this discrepancy first before the flaw can be exacerbated. When executing this modified algorithm on the input problems, the failure rates presented in Fig 8 are given. Note, however, that runtimes are worse in this model, thus 400x400 is impractical to compute. Thus, this flaw and alteration resolves many of the failure cases of a minimum entropy heuristic model yet does not provide a complete resolution of the disparity between scanline and minimum entropy performance. It should further be noted that this process is certainly more computationally expensive than simply proceeding in scanline order, which provides these benefits in a less computationally complex manner.

Tile set	Errors (Max 100 Trials)		
	Scanline	Min Entropy	Min Entropy with Modification
Summer 100x100	0	100	100
Summer 200x200	0	100	100
Castle 100x100	0	100	100
Castle 200x200	0	100	100
Knots Dense 100x100	0	1	0
Knots Dense 200x200	0	100	0
Knots TE 100x100	0	100	0
Knots TE 200x200	0	100	0
Knots T 100x100	0	4	0
Knots T 200x200	0	100	0
Knots CE 100x100	0	23	0
Knots CE 200x200	0	100	0
Rooms 100x100	0	0	0
Rooms 200x200	0	32	0
RedDot200x200	0	100	0
Skew1 200x00	4	100	0
Cat 400x400	0	6	N/A

Fig. 8. Failure rates for modified algorithm

VII. Future Work

In resolving this issue, more work must be done to identify the nature of the minimum entropy heuristic's failures in those cases not resolved by the presented modification to the algorithm, as well as to identify more efficient manners of cleanly solving for the discrepancy of minimum entropy. Alternatively, if the scanline heuristic's directional biasing were resolved, it may become preferable to discontinue the minimum entropy heuristic altogether. Further, promising work has been done by Merrel in adapting wholly different manners of solving the task which does not depend on a grid-based solution [6].

VIII. Conclusion

In conclusion, after analysis of the cases of failure of the Wave Function Collapse algorithm for certain input cases, a key cause of failure stems from elevated regions of constraints caused by the folding-in effect among resolved cells. For certain inputs, this failure can be resolved by modifying the algorithm to avoid creating case scenarios where such a failure may occur.

References

[1] M. Gumin, “Wave function collapse”, 2016.

[2] I. Karth, & A. Smith. “WaveFunctionCollapse: Content Generation via Constraint Solving and Machine Learning”. In *IEEE Transactions on Games*, vol. 14, no. 3, pp. 364-376, 2022.

[3] P. Merrel, “Example-based model synthesis,” in I3D ’07: Symposium on Interactive 3D graphics and games, ACM Press, 105–112, 2007.

[4] P. Merrel, “Model Synthesis”. PhD thesis, University of North Carolina at Chapel Hill, 2009.

[5] P. Merrel, Comparing Model Synthesis and Wave Function Collapse, 2021.

[6] Merrell. Example-Based Procedural Modeling Using Graph Grammars. *ACM Transactions on Graphics,* 2023.

[7] A. Efros and T. Leung. “Texture Synthesis by Non-parametric Sampling”. IEEE International Conference on Computer Vision, 1999.

[8] A. Mackworth. “Consistency in Networks of Relations.” Artificial Intelligence, 1977.

[9] P. Harrison. “Image Texture Tools.” Clayton School of Information Technology, 2005.

[10] R. Wallace. “Why AC-3 is Almost Always Better than AC-4 for Establishing Arc Consistency in CSPs” Proceedings of the 13th International Joint Conference on Artificial Intelligence, 1993.

[11] R. Mohr and R Henderson. “Arc and path consistency revisited.” Artificial Intelligence, 1986.

[12] A. Smith and M. Mateas. “Answer set programming for procedural content generation: a design space approach.” IEEE Transactions on Computer Intelligence and AI in Games, 2011.

[13] A. Summerville, S. Snodgrass, M. Guzdial, C. Holmgard, A. Hoover, A. Isaksen, A. Nealen, and J. Togelius, “Procedural content generation via machine learning (pcgml).” IEEE Transactions on Computer Intelligence and AI in Games, 2018.