Wavefunction Collapse Algorithm

April 2023
by Phol Wehaprasirtsak(pw483)

Abstract

This article reports the introduction Wave Function Collapse (WFC) algorithm for 2-dimensional inputs and outputs. The algorithm analyses small N x M subsections of an input, assessing how many times each appears, and which appears adjacent to each other. These are used alongside constraints to arrange the subsections inside the output. Following the practice of texture synthesis, the simplest case is a 2D black and white image, whose code was later used as the framework of this report for more complicated cases. The result was presented as an animation showing each tile collapse and how it spreads throughout the whole output image. Then, some modifications on the code were made to support coloured inputs. Finally, conditional collapse was implemented to remove unnatural seams from the flower example case.

1.Introduction

Procedural Content Generation (PCG) is defined as the creation of content using an algorithm. Wave Function Collapse (WFC) is a specific Procedural Content Generation (PCG) algorithm developed by Maxim Gumin [1] that aims to create a novel output image by considering the inherent rules and patterns of a given input image[4]. A great deal of work has been done in this field, but it is still developing with many potentials waiting to be explored. The goal of this report is to explain and analyse how to utilize the Wave Function Collapse algorithm for 2D texture generations from an image input.

2.Theoretical Background

In this section, theory and relevant definitions of image generation with the WFC algorithm are introduced.

Pixel matrix is a two-dimensional array that represents an image. Each element of the array corresponds to a single pixel in the image, and its value determines the color of that pixel. In Python, the most common convention for representing a pixel is a tuple of integer values that correspond to the red, green, and blue color channels of the pixel. This is known as the RGB colour model. The format for an RGB pixel tuple in Python is (R, G, B), where R, G, and B are integers between 0 and 255 that represent the intensity of the red, green, and blue color channels of the pixel, respectively. In addition to the RGB color model, there are other color models. One that will be used here is grayscale for creating images in black and white. The most common convention for representing a grayscale image using the Matplotlib library is a two-dimensional NumPy array, where each element of the array represents the grayscale intensity

of a single pixel in the image. The values in the array are typically integers between 0 and 255, where 0 represents black and 255 represents white. Values in between represent varying shades of gray.

Texture synthesis is the process of algorithmically constructing a large digital image from a small digital sample image by taking advantage of its structural content. Texture synthesis algorithms are intended to create an output image that meets the following requirements: [5] The output should have the size given by the users. The output should be locally similar to the sample, which means the ratios of the same pixel elements appearing in the output image should be the same as those from the input. The output should not have visible artifacts such as seams, blocks and misfitting edges.

The Constraint Solving Algorithm is the key part of the Wave Function Collapse algorithm (WFC). Constraint satisfaction problems (CSPs) are typically defined in terms of 'variables' and 'values'. In the context of WFC-style image generation, there is a variable associated with each location in the output image. In a solution to the problem, each variable takes on a value. Depending on the context, values may come from continuous or discrete domains. For the task addressed by WFC, the values are associated with the discrete set of unique local patterns in the input image. The choice to assign a variable a specific value will often influence the available choices that can be made for other variables. Constraints relate the legal combination of values that a set of variables might take on. For the image generation task, we want to model the idea that the patterns chosen at each location in the output are compatible in terms of exact matches for the pixels in which their associated local windows overlap. The goal of an algorithm for solving CSPs is to find a total assignment (an assignment for every variable) such that no constraints are violated.

Some heuristics aid the selection of a promising variable to select next. Complementary to heuristics, constraint propagation methods do additional bookkeeping by pruning away values from domains that would lead to dead-ends later. Constraint propagation ideally allows a solver to skip past fruitless search without impacting the order in which the space is explored [2].

3. Black and white small size image

3.1 Method

1. **Define input and create a grid of pixels that represents the output image**

As shown in Figure 1, a 4x4 black and white image was selected to use for the first version of the wavefunction collapse code. The small size of this sample will help to simplify and better visualise the whole process. Since RGB colours are not being used, our sample and later patterns can be written as simple array of integers with one value per pixel. The output size was selected to be 50x50 as will be the same throughout this report.

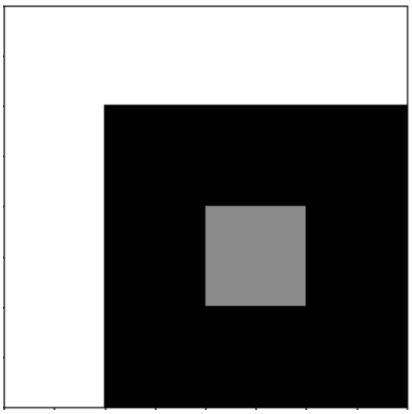

Figure 1: Input image 4x4 sample used as source image

2. Define the size of sub patterns to be of size 2x2 and extract all patterns and their rotations

Some code was written to extract all possible unique patterns of size 2x2 pixels (12 in total for the input). Their 90° rotations are included so the image can expand in all directions. “Weight” in Figure 2 tells how many times the given pattern occurs in the sample. “Prob” stands for probability of the pattern occurring in the input image which is equal to that in the output image (local similarity).

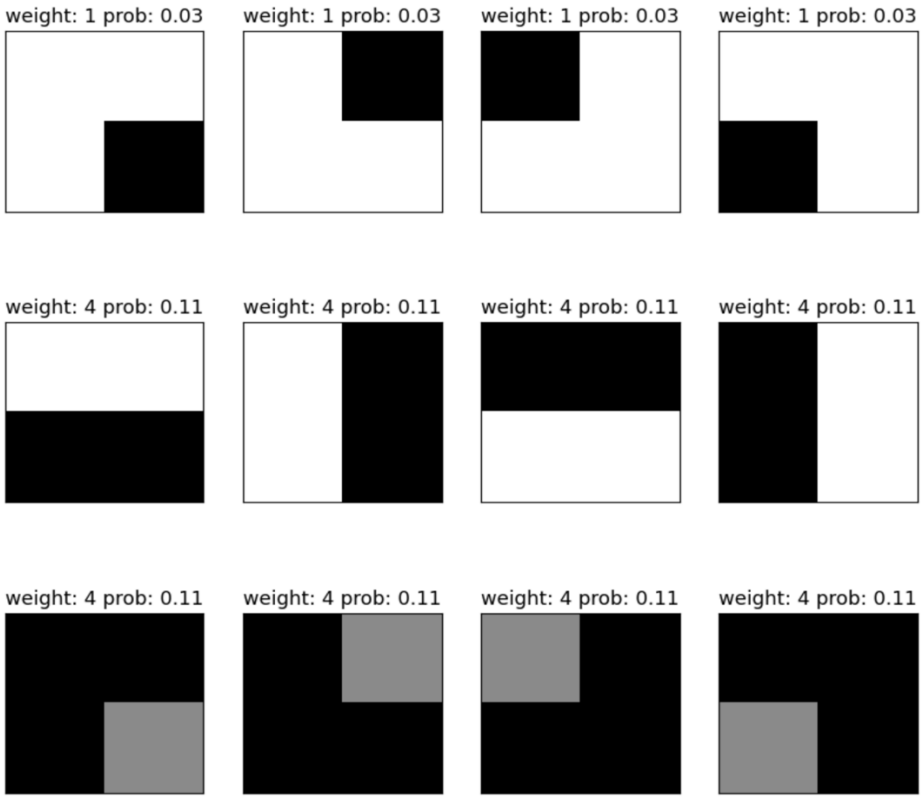

Figure2: 2x2 patterns and their rotations extracted from the sample. All possible 2x2 patterns are shown with the associated probability signifying how likely the pattern to be found on the sample.

3. Initialise each pixel of the output image with a set of possible values based on the input sample.

In the beginning, all 12 patterns are possible at all positions. So, every pixel of the output matrix was initialized with a list of all patterns.

The 9 neighbouring directions were defined as a tuple, where the elements are the positional indices in the output array as shown in Figure 3(a).

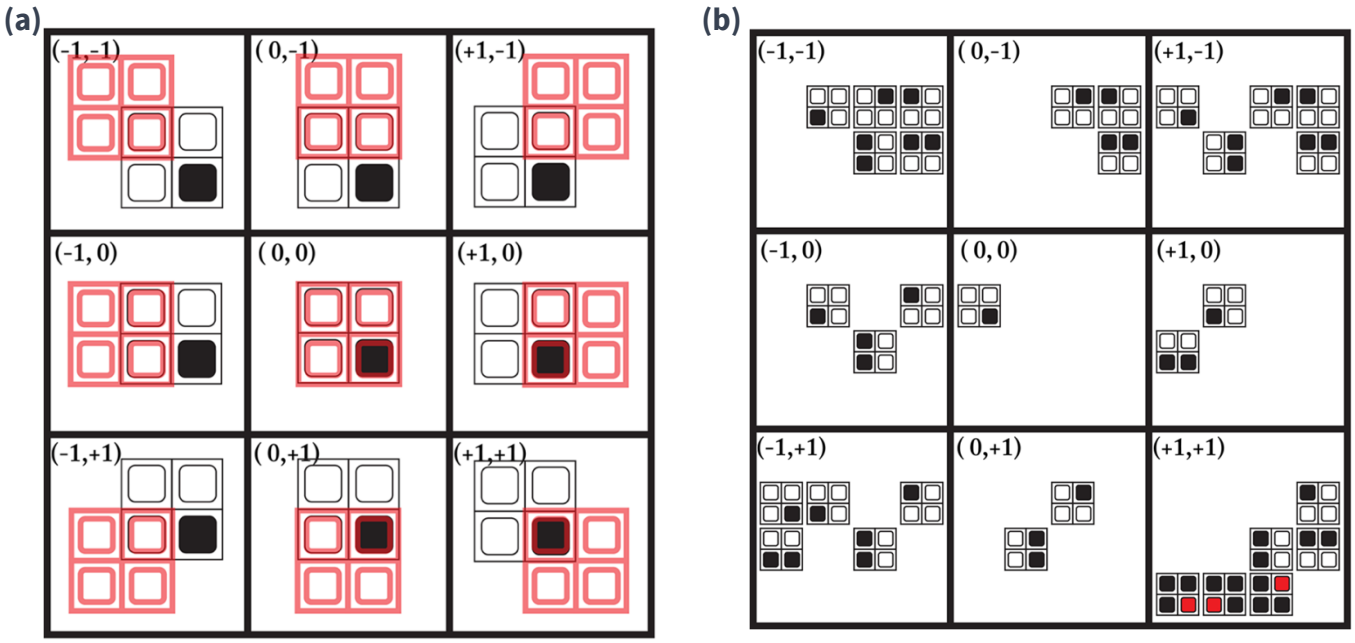

Figure3: Setting up to determine the allowed adjacent patterns. The top left pattern from Figure1 represents the pattern at the centre[2].

(a) The nine ways that two 2 × 2 patterns can overlap. For each pattern, the adjacent indices hold the precalculated list of which other patterns are valid.

(b) Example of deriving compatible patterns that can overlap with the central pattern in each direction. Note that the grey colour from the input (Figure1) has been replaced by red in this diagram.

A class of possible adjacent patterns for each central pattern was written in preparation for the constraint propagation step later. A schematic diagram of compatible patterns from each direction is shown in Figure 3(b).

4. **The algorithm then applies a set of rules to the current output pixel values to determine which values are possible based on the neighboring pixels and the input sample.**

The minimum remaining value heuristic was implemented, where Shannon entropy (**Appendix A**) is used as the indicator. The entropy values for each of the output indices were continuously calculated and compared. The process either terminates when all the pixels have zero entropy, or an error occurs. Otherwise, the wavefunction at the pixel with minimum entropy will collapse to a pattern based on the associated probability of each possible patterns.

5. **Propagate and repeat iteratively**.

The lists of possible patterns in the output matrix are successively updated due to the additional constraint from the recently collapsed position.

Gumin's algorithm does not implement local backtracking and instead globally restarts in the rare case a conflict is reached. The algorithm must be run another time in the event that conflict occurs. For this investigation, the probability of conflict being reached is thought to be negligible. Therefore, it is unlikely to impede runtime significantly.

6. **Post-processing and animation.**

Lastly, post-processing techniques are applied to the output image. In this example, the top left tile of each pattern on each pixel is chosen to represent the pixel.

The workflow diagram has been drawn to summarise all the steps in on in Figure 4 shown on the next page.

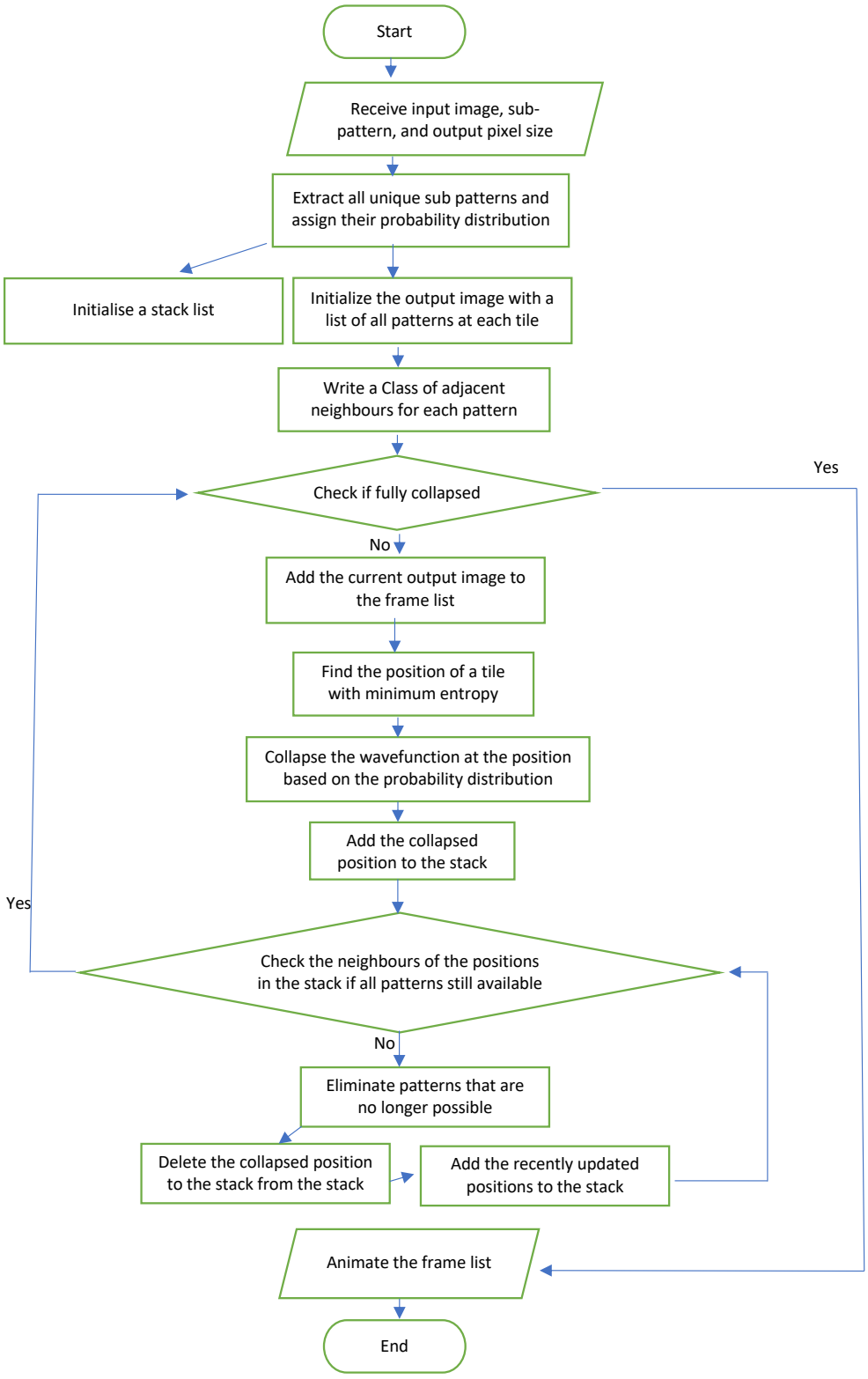

Figure 4: Workflow of the entire process.

3.2 Results and Discussion

Figure 5 shows that the animation works as intended. Figure 5(a) is when the first location is collapsed at random. The propagation starts from the collapsed tile spreading through its neighbours in Figure 5(b) and (c) until completion in Figure 5(d). Partially resolved cells are rendered as the average of their potential outputs; hence, appear blurry.

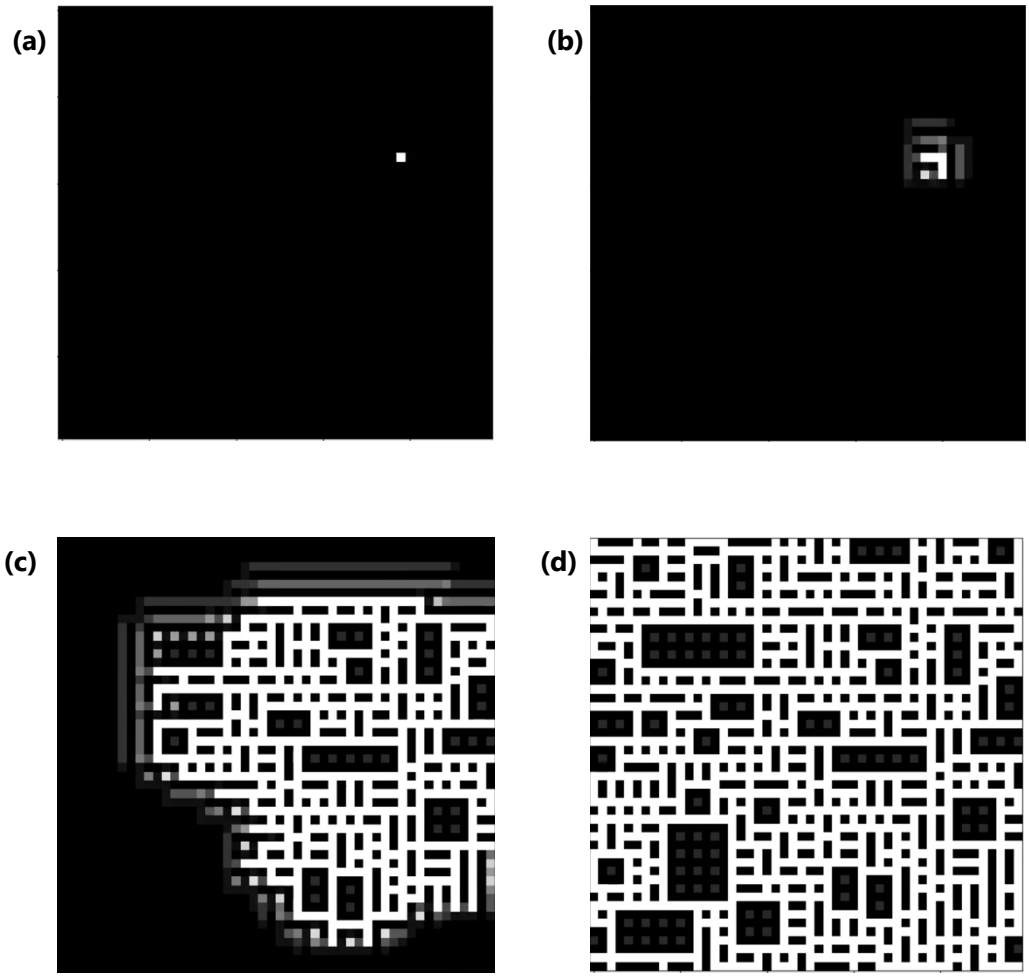

Figure 5: 4 frames of the animation illustrate how the output is presented as animation form. The frame order is from Figure 5(a) to (b) to (c), and Figure 5(d) is the fully collapsed output image.

In the attempt to achieve local similarity, the rule for the collapsing tile was modified. Results from two modifications are close to local similarity but deviate from it from two opposite ends.(Figure6)

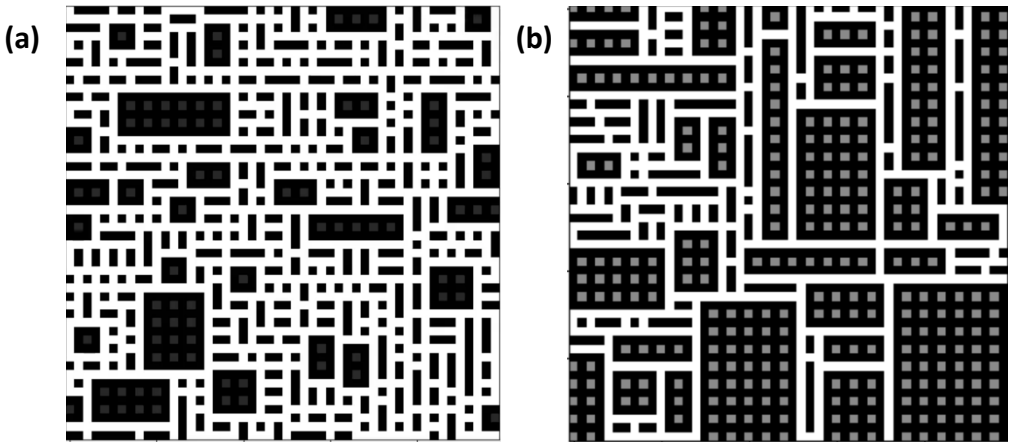

Figure 6: Possible outcomes from the two modifications.
(a) Maximum probability modification; (b) Probability distribution modification.

In Figure 6(a), a possible outcome using the "Maximum probability modification" to the collapsing rule is shown. This rule means that if there is more than one possible pattern at the location with minimum entropy, the pattern with the biggest associated probability will be chosen. In contrast, the "Probability distribution modification" (Figure6(b)) collapsing rule means that the patterns were chosen with a random sample weighted by the frequencies that patterns appear in the input image.

From Figure1, the input has 1 grey pixel out of 16, which accounts for 6.25 %.
From Figure6, the Maximum probability case has grey content of 297/2500 = 11.88 %.
,and the Probability distribution case has grey content of 82/2500 = 3.28 %.

It is notable that wavefunction collapse algorithm is highly probabilistic and the outcome from each run will be different. Nevertheless, there is still a trend that can be observed. The grey content of the ideal case is positioned between the results from the two modifications. Therefore, it is suspected that the algorithm that would achieve "local similarity" would lie somewhere in between the two cases.

Backtracking search method [2] is not being used here as the technique has been proved to be unnecessary especially not many constraints have been introduced. Backtracking would significantly require longer runtime, while the likelihood of encountering error with the current method is so slim that it is not worth slowing down the runtime of the algorithm. From this, it is reasonable to globally restart the greedy search if a conflict is encountered instead.

4. Black and white imported image without constraint

Another step forward is to import real image and implement the code from previously.

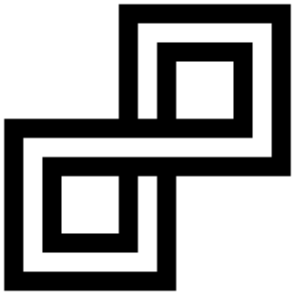

Figure 7: Image of 'knot' picture as an example of imported black and white input

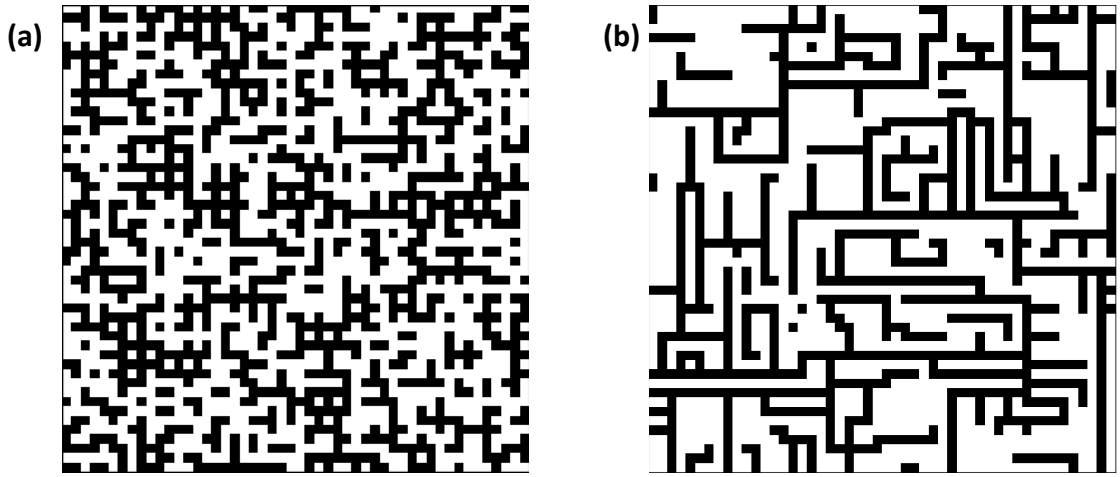

Figure 8: Possible outcomes from the two modifications.
(a) Maximum probability modification; (b) Probability distribution modification.

The code from the previous part works as intended. The only difference is the pixel matrix of the input image (Figure7) has to be imported using matplotlib.pyplot.imread, and the results can be observed in Figure 8. Figure8(b) is very similar to the examples displayed on Gumin's Github repository[1], which suggests that the "Probability distribution modification" might be preferred for the input.

5. RGB coloured imported image with constraint

RGB coloured images have lists of length 3 representing colours for a pixel as opposed to an integer in the black and white case. Thus, the lists were encrypted as an integer in a dictionary called 'colour dictionary' in order to accommodate the code from the black and white input case. The encrypted input would undergo through the same process as previously and get decrypted at the end for the matplotlib library to be able to show the matrix as images/animations output. Similarly, if the input images are in RGBA format (list of length 4 representing a pixel), the same idea can be used to cover the problem.

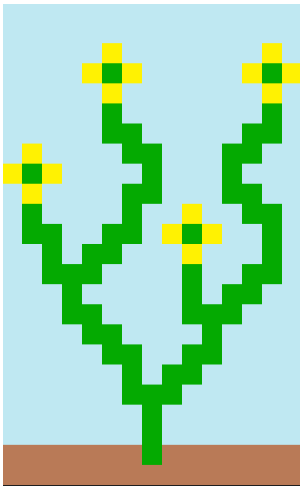

Figure 9: Image of 'flowers' picture as an example of imported coloured input.

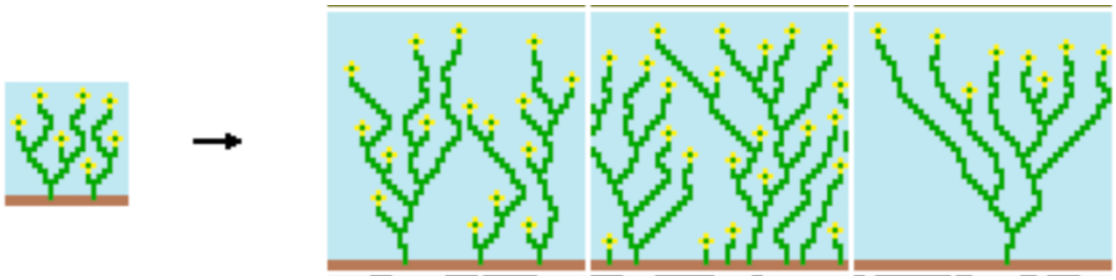

Figure 10: Image of 'flowers' and some of the expected outcomes as presented on [1].

The code from the previous part did not work as intended at all. There were several problems as listed here: the flower can propagate in only certain directions, the green pixels inside the flowers were indistinguishable and the stems do not start from the ground.

The first problem was addressed by eliminating all the rotations. The second one was slightly more involved. By adding another key to the colour dictionary, the colour green enclosed by the flower petals(yellow) can be distinguished from other green pixels(stems)

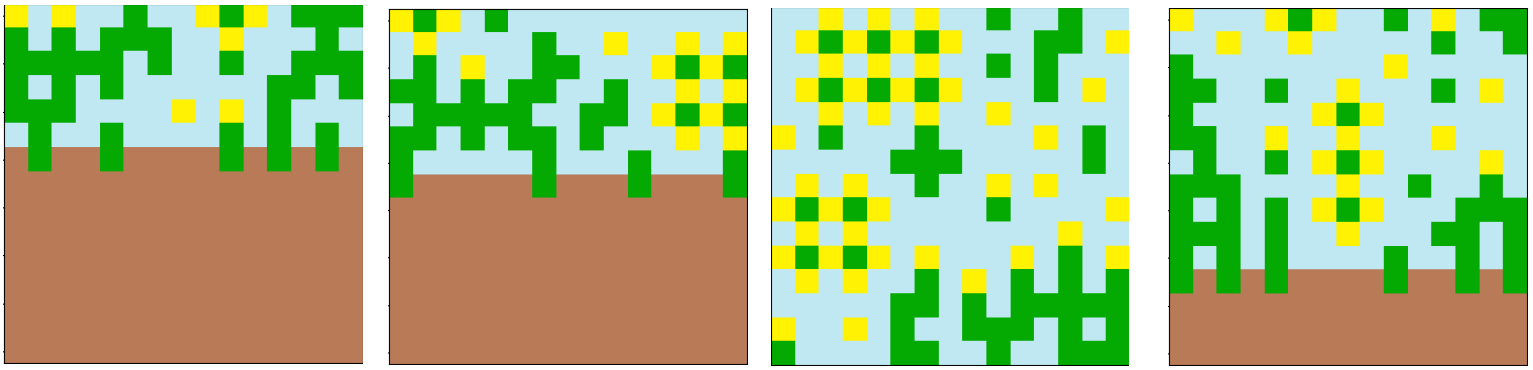

Figure 11: Examples of the outcomes from the Maximum probability modification.

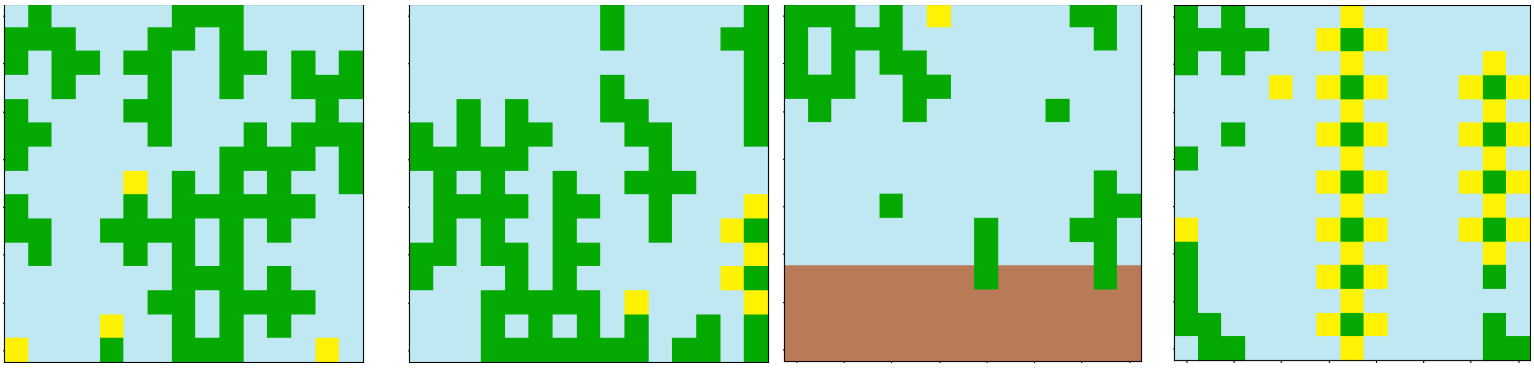

Figure 12: Examples of the outcomes from the Probability distribution modification.

From Figure 11 and 12, the results are still unnatural especially the pictures from Figure 12. So, for this input, the "Maximum probability modification" seems to give more natural results with fewer levitating stems.

5.Conclusion

We have shown that Wavefunction Collapse is a significant application of constraint solving for PCG with multiple uses. The report started from dealing with 2 dimensional small black and white inputs. Then, it expands the code to cover imported inputs, coloured inputs and eventually includes additional constraints. Several attempts to modify the collapsing rules were made, and the most sensible modifications are the "Maximum probability modification" and the "Probability distribution modification". Both lead to satisfying yet deviating results from the ideal case. I believe that it is only a matter of preference for each task to decide which definition is better. Through experiments with the ASP surrogate implementation [2], it can be shown that WFC's choice of heuristic and decision to only apply global restarts of search are reasonable choices for the original discrete image generation task. WFC also works with

abstract chunks of content rather than literal, blendable color values. It has many exciting applications such as poetry and constrained level generation[1].

6.References

[1] Maxim Gumin. 2016. Bitmap & tilemap generation from a single example by collapsing a wave function https://github.com/mxgmn/WaveFunctionCollapse. (30 Sep 2016). Retrieved May 20, 2017 from https://twitter.com/ExUtumno/ status/781834584136814593

[2] Isaac Karth, Adam M. Smith, 2017. *WaveFunctionCollapse is Constraint Solving in the Wild* , Department of Computational Media, University of California Santa Cruz

[3] Quentin Morris. 2021. Modifying Wave Function Collapse for more Complex Use in Game Generation and Design, Department of Computer Science, University of Trinity.

[4] Texture Synthesis, Department of Computer Science, University of Berkeley.

Appendix A:

Shannon entropy

In information theory, the entropy of a random variable is the average level of "information", "surprise", or "uncertainty" inherent to the variable's possible outcomes.

$$\boldsymbol{H(X) = \sum_{i}^{n} -P_i \, log_2(P_i)}$$

For this report, the formula expresses the mathematical expectation of uncertainty. The bigger number of possible patterns for a tile, the bigger the "uncertainty" (entropy) is. Tile where only one pattern is possible has entropy equal to 0