1.5em 0pt

Determining Effective Cell Selection Strategies in Wave Function Collapse for Minecraft

Shailyn Ramsamy Moodley
University of the Witwatersrand, Johannesburg
2439416@students.wits.ac.za

***Abstract*—Procedural Content Generation (PCG) has become increasingly vital in modern game development, offering solutions to create vast, diverse, and engaging virtual worlds while minimizing manual content creation effort. As recently as 2016, the WaveFunction Collapse (WFC) algorithm has emerged as a powerful tool for generating coherent, pattern-based content. This research investigates the application of WFC for generating architecturally sound and aesthetically pleasing buildings in Minecraft, with a specific focus on how different cell selection strategies impact the generated structures. The study implements and analyses four distinct cell selection strategies: entropy-based, height priority, center-based, and random walk. Through quantitative analysis of pattern diversity, spatial metrics, and structural coherence, we evaluate how each strategy influences the architectural characteristics of the generated buildings. The findings contribute to the broader understanding of PCG systems by providing empirical evidence of how selection heuristics in WFC algorithms affect output characteristics. This research has particular relevance for sandbox games and virtual world generation, where architectural coherence must be balanced with aesthetic variety. These results suggest that different cell selection strategies can be leveraged to achieve specific architectural goals, from creating more grounded, stable structures to generating more organic, flowing designs. The implementation code is available on GitHub at https://github.com/Shailyn-Ramsamy/Cell-Selection-Strategies-for-Minecraft-WFC**

I. Introduction

Procedural Content Generation (PCG) plays an increasingly vital role in modern game development, offering solutions to create vast and engaging virtual worlds while minimizing manual content creation costs. Within this domain, the WaveFunction Collapse (WFC) algorithm has emerged as a powerful approach for generating coherent, pattern-based content Maxim Gumen [1]. However, while WFC has shown promise in generating detailed structures, the impact of its core selection strategies on the final output remains relatively unexplored. The algorithm's effectiveness in creating functionally valid and aesthetically pleasing buildings can vary significantly based on how it chooses which cells to collapse during the generation process. This selection process is particularly crucial in Minecraft, where buildings must balance structural integrity with aesthetic appeal while conforming to the game's unique voxel-based constraints. This research addresses this gap by conducting a comprehensive analysis of four distinct cell selection strategies within the WFC algorithm: entropy-based, height priority, center-based, and random walk. Through quantitative evaluation of pattern diversity, spatial metrics, and structural coherence, the experiment investigates how different selection heuristics influence the architectural characteristics and functional validity of generated Minecraft buildings. This study contributes to the field of PCG by providing empirical evidence of how selection strategies in WFC algorithms affect output characteristics. Furthermore, the metrics and analysis methods developed in this research provide a framework for evaluating and comparing different approaches to procedural building generation.

II. WaveFunction Collapse

WFC is a family of algorithms that use constraint solving as the core generation algorithm. Gumin occasionally describes the algorithm as a constraint solver [2]. The WaveFunction Collapse (WFC) algorithm employs the principles of quantum mechanics, specifically the concepts of superposition and state collapse. In this context, a "particle" represents a tile or element in the generation process, which can exist in multiple possible states simultaneously, akin to a quantum superposition. The algorithm operates through several sophisticated processes to generate its output. First, tiles are either manually defined or extracted from an example input, with constraints then determined by their valid adjacency relationships. In the case of example-based extraction, the algorithm analyses overlapping patterns within the input, typically using N×N windows (where N is commonly 2 or 3) to capture local structural relationships. These patterns become the fundamental units for generation, preserving both local and global characteristics of the original design. A crucial component of WFC is its use of entropy to guide the generation process. The entropy of a cell is calculated as the weighted sum of the patterns that could still be validly placed at that location, with weights derived from pattern frequencies in the input. Mathematically, the entropy E of a cell is defined as:

$$E = -\sum(p_i log(p_i)) \tag{1}$$

where p_i represents the normalized weight (probability) of pattern i in the cell's current domain. This entropy measure is maximized when all patterns are equally possible (uniform distribution) and minimized when only one pattern remains possible (complete certainty). The algorithm strategically selects cells with minimum non-zero entropy for collapse, as these represent the most constrained choices with the least uncertainty. The core generation process maintains a complex wave function representing the superposition of all possible

states for each cell in the output grid. Each cell initially contains the complete set of possible patterns, and this set is progressively reduced through the algorithm's execution. The generation occurs through two primary operations: observation and propagation. During observation, the algorithm employs the entropy heuristic to identify the cell with the lowest entropy - meaning the cell with the fewest remaining valid patterns in its superposition state. This cell is then "collapsed" by selecting one pattern from its available options, often weighted by the pattern's frequency in the original input to maintain statistical similarity with the source material. The propagation phase then enforces local consistency by updating the constraints of neighboring cells based on the recent collapse. This constraint propagation follows the adjacency rules established during the initial analysis phase, removing invalid patterns from the superpositions of affected cells. This process continues recursively until no further constraints can be propagated. The algorithm alternates between observation and propagation until all cells have been collapsed to a single pattern, resulting in a complete, coherent output that respects both local and global constraints of the original design. We can see impressive results from just the inputted example level to the output in figure 1.

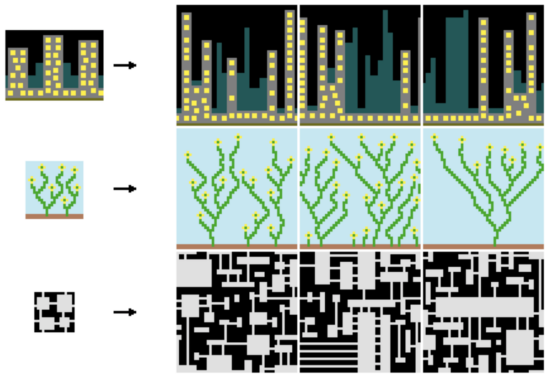

Fig. 1. Input example generating levels using WFC

III. Related Work

A. Evolution of Procedural Content Generation

Early implementations of PCG in video games can be traced back to 1980 with Rogue, which introduced fundamental concepts of algorithmic level generation [3]. According to Short and Adams [3], the level generation in Rogue followed a systematic approach: randomly placing rooms of varying sizes and shapes on a level grid, connecting these rooms by digging out corridors between them, and finally scattering additional features like stairs, monsters, and items throughout the level. While this algorithm was relatively straightforward, it introduced an element of unpredictability and replayability. Throughout the 1980s and 1990s, games like "Pac-Man" (1980) and "Zelda II: The Adventure of Link" (1987) incorporated increasingly complex AI-controlled opponents and path-finding algorithms [4]. The field has seen substantial growth, with research expanding into search-based PCG [5], machine learning-based approaches [6], and constraint solving methods. This evolution has been further accelerated by competitions and conferences dedicated to advancing AI techniques for game-playing, such as the General Video Game AI Competition [7] and the Generative Design in Minecraft (GDMC) settlement generation competition [8].

B. Minecraft Settlement Generation

Christiansen and Scirea [9] introduced a novel approach called "traversal segmentation" for partitioning the three-dimensional Minecraft world into distinct regions. The key idea is to employ multiple rule-based autonomous agents that operate within these regions to generate settlements. The agents use a simulated two-system brain model, inspired by dual-process theory in the book "Thinking, fast and slow" by [10], to make decisions. Similarly, [11] employed autonomous agents for terrain analysis, architectural design, and population modeling. Search-based algorithms, such as evolutionary techniques, are applied to refine the generated designs, ensuring they adhere to constraints like structural integrity and resource availability. Unlike the algorithm that [9] explains, the authors employed optimization strategies like parallelization, spatial partitioning, and level-of-detail rendering to enhance computational performance and scalability.

The use of rule-based pcg in minecraft is used differently in the paper [12]. It explains the use of a combination of constrained growth algorithms and cellular automata to create diverse and functional floor plans for buildings in minecraft. Cellular automata are used to introduce variation and organic patterns in the generated floor plans. While this approach succeeded in generating realistic floor plans, it did not address the broader challenge of generating entire settlements or integrating the generated structures into a coherent environment like [9] and [11].

C. Constraint-Based PCG

Constraint-based approaches to PCG have emerged as a powerful method for generating coherent and controllable content. These methods differ from traditional randomized approaches by enforcing specific rules and relationships between generated elements. Early systems like G. Smith's Tanagra [13] showed how constraint satisfaction problems could be effectively used in mixed-initiative level design tools, allowing designers to input data through graphical interfaces [2]. The field has since evolved to support more accessible approaches to constraint-based generation, with systems like Imaginarium allowing users to specify constraints through natural language interfaces [14]. The WaveFunction Collapse algorithm represents a significant advancement in this domain [2], combining concepts from constraint solving with example-based generation to produce highly structured content while maintaining local coherence. What makes constraint-based PCG particularly valuable is its ability to make strong guarantees about outputs, though this comes with the tradeoff of unpredictable running time [2].

D. *WaveFunction Collapse in Practice*

Since its introduction by Gumin in 2016, WFC has been adopted across various applications in game development. Notable implementations include Bad North's terrain generation and Caves of Qud's village generation systems [2]. More recently, NIELS-NTG's "Vault of Immeasurable Knowledge" (seen in Figure 2) project demonstrates WFC's potential for complex architectural generation by creating a massive procedurally generated underground library, earning 4th place in the 2024 GDMC competition [15]. What makes WFC particularly noteworthy is its ability to learn complex pattern relationships from minimal input data, often requiring only a single small example to generate larger, coherent outputs. This efficiency in learning from limited examples sets it apart from data-hungry machine learning approaches that typically require extensive training datasets.

Fig. 2. Vault of immeasurable knowledge

E. *Pattern-Based Generation Approaches*

WFC builds upon earlier work in pattern-based generation, particularly in the field of texture synthesis. These approaches analyse local patterns in example content to generate new, similar content that maintains the statistical properties of the original. However, WFC innovates by treating this as a constraint satisfaction problem, ensuring strict adherence to the learned patterns while allowing for creative recombination within these constraints.

IV. Methodology

A. *Adjacency Constraint Framework*

Unlike traditional WFC implementations that extract patterns from example inputs, this approach directly encodes structural relationships through an explicit constraint system. The system defines a set of 25 distinct structural elements (each consisting of 9x7x9 blocks) representing different building components (walls, corners, entrances, etc.) as seen in Figure 3 and establishes their valid adjacency rules through a six-directional constraint system (North, South, East, West, Up, Down). Each structural element maintains connection rules across four possible rotational states, with constraints automatically establishing bidirectional relationships. For example, when defining that a wall_top can connect to an inner_corner_top to its east, the system automatically establishes that the inner_corner_top can connect to the wall_top to its west, with appropriate rotational transformations.

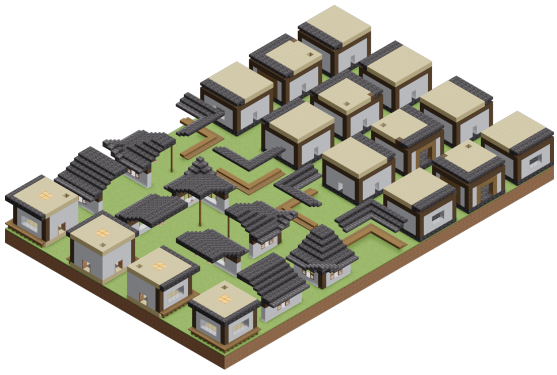

Fig. 3. Examples of created tiles.

This approach allows for architectural validity while maintaining the creative recombination capabilities of WFC. By encoding structural knowledge directly into the constraint system, generated buildings maintain proper support, connectivity, and aesthetic coherence while still allowing for novel combinations of building elements. However, a limitation of this manual constraint definition approach compared to traditional pattern extraction is its reduced ability to capture subtle design patterns and variations that might be present in human-created examples. The system is constrained by the explicitly defined rules rather than learning from the rich variety of patterns that could be extracted from example buildings.

B. *WaveFunction Collapse implementation*

The Wave Function Collapse (WFC) algorithm implementation for Minecraft building generation follows the core principles of superposition and constraint propagation. Each cell in a three-dimensional grid initially contains all possible building modules in various rotational states. The algorithm iteratively collapses cells to single states while maintaining local consistency through constraint propagation. The implementation maintains a grid of cells where each cell contains a set of possible states (module, rotation, variation). During each iteration, a cell is selected for collapse based on its entropy (number of possible states). Upon collapse, the algorithm propagates the constraints to neighboring cells, removing incompatible states based on the pre-defined adjacency rules. This process continues until either all cells are collapsed or a contradiction is reached, requiring backtracking. A key modification from traditional WFC is the handling of variations within modules. Each building module can have multiple weighted variants, allowing for aesthetic diversity while maintaining structural coherence seen in Figure 4. For example, wall segments might have different decorative patterns while maintaining the same connection rules.

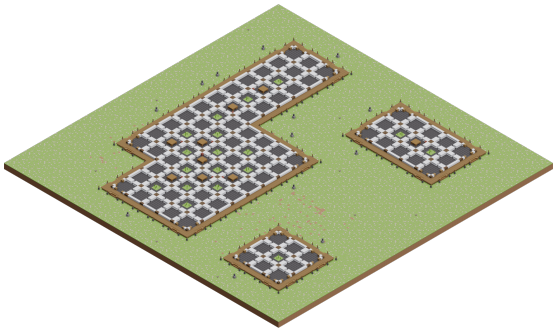

Fig. 4. Weighted internal tile visualisation.

The core WFC algorithm can be summarized as follows:

Algorithm 1 Wave Function Collapse

procedure WFC($Grid, MaxIterations$)
  Initialize $Grid$ with all possible states
  while not $Grid$.isFullyCollapsed() and iterations $<$ MaxIterations **do**
    $cell \leftarrow$ SelectMinEntropyCell($Grid$)
    if $cell$ is None **then**
      break ▷ Generation complete
    end if
    $states \leftarrow$ GetPossibleStates($cell$)
    $chosen \leftarrow$ WeightedRandomChoice($states$)
    SetCellState($cell$, $chosen$)
    PropagateConstraints($Grid$, $cell$)
  end while
  return $Grid$
end procedure

During constraint propagation, each cell's possible states are filtered based on the adjacency rules of its neighbors in all six directions (North, South, East, West, Up, Down). When a cell's state is collapsed, only those neighboring states that have valid connections with the chosen module are retained. This ensures the structural integrity of the generated building while allowing for creative variations in the final output.

C. Cell selection strategies

The effectiveness of the Wave Function Collapse algorithm in generating coherent building structures is significantly influenced by the cell selection strategy. This section presents four distinct strategies that produce meaningfully different architectural outcomes:

1) ***Entropy-Based Selection****:* This baseline strategy, derived from the original WFC implementation, selects cells based on their entropy - the number of possible states remaining. Cells with fewer possible states are prioritized, with small random noise added to break ties. This approach promotes local consistency by resolving the most constrained areas first, leading to balanced, well-distributed structures:

2) ***Height Priority****:* This strategy enforces a bottom-up construction pattern by prioritizing cells at lower y-coordinates.

Algorithm 2 Entropy-Based Selection

function CALCULATEENTROPY(x, y, z)
  $cell \leftarrow grid[z][y][x]$
  if $|cell| \leq 1$ **then**
    return ∞
  end if
  return $|cell| + random() \times 0.1$
end function

By ensuring lower levels are constructed first, it mimics real-world construction processes and produces more structurally sensible buildings. This approach is particularly effective for multi-story structures where foundation stability is important:

Algorithm 3 Height Priority Selection

return $\min(candidates, key = \lambda pos : pos[1])$

3) ***Center-Based Growth****:* Creates structures that expand outward from a central point by prioritizing cells based on their distance from the grid's center. This approach tends to produce more symmetrical and balanced buildings, with a clear architectural focus point. The strategy calculates the Euclidean distance from each candidate cell to the center of the grid:

Algorithm 4 Center-Based Selection

function DISTANCETOCENTER(pos)
  $dx \leftarrow pos.x - center.x$
  $dy \leftarrow pos.y - center.y$
  $dz \leftarrow pos.z - center.z$
  return $\sqrt{dx^2 + dy^2 + dz^2}$
end function

4) ***Random Walk****:* This strategy simulates an organic growth pattern by selecting cells through a constrained random walk. Starting from a previous position, it limits the selection to cells within a maximum step distance (default 2 units). If no valid cells are found within this range, it makes a random choice from all candidates. This creates more flowing, organic structures that still maintain local coherence:

Algorithm 5 Random Walk Selection

$maxStep \leftarrow 2$
if $previousPos = null$ **then**
  return RandomChoice($candidates$)
end if
$close \leftarrow$ FilterCandidatesWithinDistance($candidates$, $previousPos$, $maxStep$)
if $close$ is not empty **then**
  return RandomChoice($close$)
end if
return RandomChoice($candidates$)

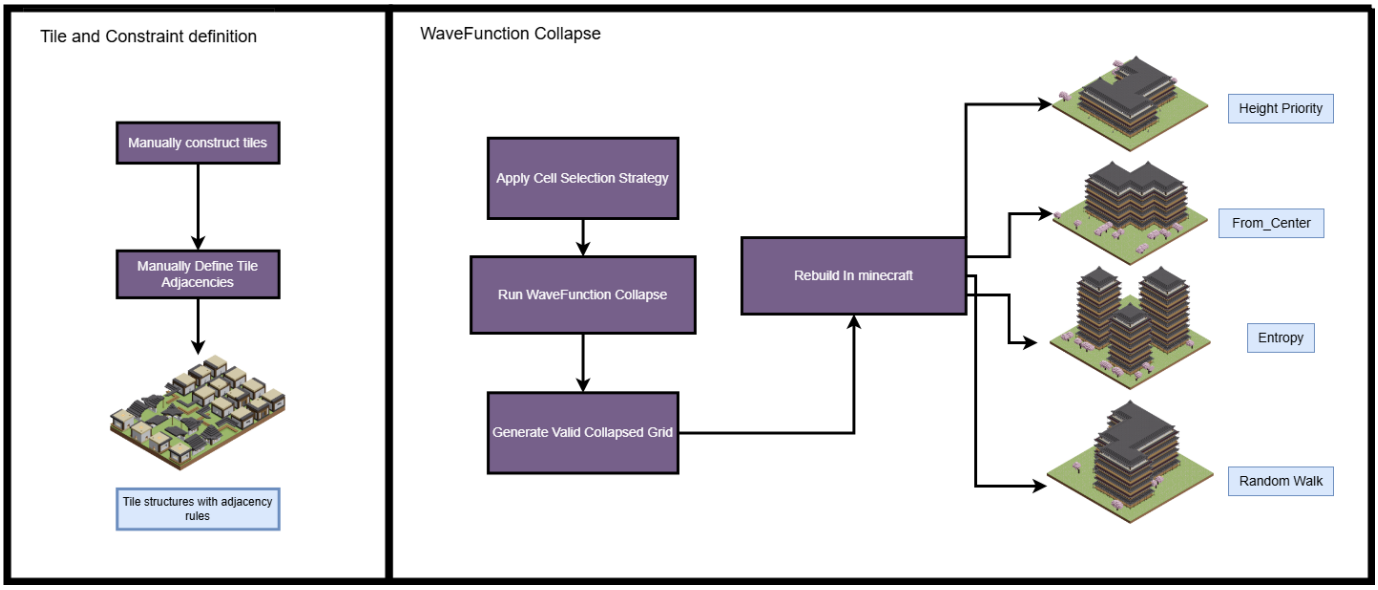

Fig. 5. Overview of the WaveFunction Collapse pipeline for Minecraft building generation. Left panel: Initial setup phase for manual tile creation and adjacency rules. Right panel: Core WaveFunction Collapse algorithm execution flow, showing the transition from cell selection through final building reconstruction with four distinct strategies $(HeightPriority, From\_Center, Entropy, RandomWalk)$ producing varied architectural outcomes.

V. Experiments

A. Experimental Design and Setup

The experiment evaluated four distinct cell selection strategies for the Wave Function Collapse algorithm in the context of Minecraft building generation. Testing was conducted across varying grid sizes, beginning with 8×8×8 structures and incrementing dimensions every 40 runs up to 13x13x13. Each strategy underwent 240 total runs, providing robust statistical sampling for analysis.

The four strategies under evaluation were:

- Entropy-based selection (traditional WFC approach)
- Height priority selection (bottom-up construction)
- Center-based growth (radial expansion)
- Random walk selection (organic growth patterns)

The experiment utilized a parallel processing pipeline to efficiently manage the large number of generation attempts. Each run generated a complete structure and calculated a comprehensive set of metrics, with special attention paid to four key measurements that capture the essential characteristics of the generated structures.

B. Core Metric Definitions and Implementation

*1) **Pattern Diversity Analysis**:* Pattern diversity is quantified using the Shannon-Wiener diversity index (H'), a fundamental measure in information theory and ecology [16] adapted for architectural pattern analysis. The formula for this metric is:

$$H' = -\sum(p_i ln(p_i)) \tag{2}$$

where p_i represents the proportion of each pattern type in the generated structure. This metric provides a sophisticated way to measure how effectively each strategy utilizes the available pattern space. Higher values indicate more diverse pattern usage, while lower values suggest more repetitive structures.

The implementation handles edge cases where certain patterns might be absent:

Algorithm 6 Pattern Diversity Calculation

```
function CalcPatternDiversity(patternCounts)
    total ← ∑ patternCounts.values
    proportions ← ∅
    for count in patternCounts.values do
        proportions.append(count/total)
    end for
    diversity ← 0
    for p in proportions do
        if p > 0 then
            diversity ← diversity + (p × ln(p))
        end if
    end for
    return −diversity
end function
```

*2) **Structural Connectivity Assessment**:* The connectivity ratio metric builds upon graph theory principles established by Hendrikx et al. [17] for evaluating procedural content coherence. This measure examines the ratio of actual connections between building elements to the maximum possible connections in a fully connected structure:

$$CR = C/Cmax \tag{3}$$

This metric proves particularly valuable in identifying strategies that produce more cohesive and architecturally sound structures. The implementation considers both horizontal and vertical connectivity:

Algorithm 7 Structural Connectivity Calculation

```
procedure CalcConnectivityRatio(grid)
    actualConnections ← CountValidConnections(grid)
    maxConnections ← CalculateMaxPossibleConnections(grid.shape)
    return actualConnections/maxConnections
end procedure
```

*3) **Vertical Coherence Measurement**:* The assessment of vertical construction quality employs a modified version of the Gini-Simpson index that specifically examines layer-to-layer transitions. The vertical stratification metric is defined as:

$$VS = 1 - \sum (L_i - L_{i-1})^2 / n \tag{4}$$

where L_i represents the density of each vertical layer. This metric is particularly important given Minecraft's gravity-bound construction constraints. Higher values indicate smoother vertical transitions, while lower values suggest more abrupt changes between layers.

*4) **Pattern Distribution Regularity**:* The Kullback-Leibler (KL) divergence measures how pattern distributions in generated structures differ from expected distributions. While Lucas and Volz [18] demonstrated its effectiveness for 2D tile-based content, this implementation extends the concept to 3D voxel-based structures by analysing layer-wise pattern distributions:

$$DKL(P||Q) = \sum P(x) log(P(x)/Q(x)) \tag{5}$$

Where:

- P represents the pattern distribution in a given layer
- Q represents the distribution in adjacent layers
- x represents tile patterns (module, rotation, variation tuples)

The average KL divergence across all generated structures provides insight into each strategy's consistency and predictability. Lower values indicate more uniform pattern distribution, while higher values suggest more variable or chaotic generation patterns.

VI. Results and Discussion

A. Overview of Experimental Results

This study's comprehensive evaluation of four distinct cell selection strategies reveals intricate patterns in both their distributional characteristics and temporal evolution. The combined analysis of violin plots and longitudinal metrics seen in Figures 9 and 7 provides deep insights into how each strategy shapes the generated structures across multiple dimensions of architectural quality.

B. Pattern Diversity Analysis

The pattern diversity analysis reveals that all four strategies perform remarkably similarly in terms of their growth patterns and absolute values. Looking at the pattern diversity line graph, all strategies follow an almost identical logarithmic growth curve, starting around $2{\times}10^3$ at 500 blocks and reaching approximately $7{\times}10^3$ at maximum volume. The confidence intervals show considerable overlap throughout the entire volume range, suggesting no strategy has a clear statistical advantage over the others in terms of pattern diversity. We see some minor differences regarding entropy-based and from_center strategies showing slightly higher values above 1750 blocks - these differences are minimal, with all strategies falling within a very narrow band throughout their evolution. The height priority strategy shows marginally lower values after 1500 blocks, plateauing slightly below the others at around $6.5{\times}10^3$, but this difference is subtle and may not be practically significant. The consistency across strategies suggests that pattern diversity may be more influenced by the underlying WFC algorithm and available pattern set than by the specific cell selection strategy employed.

C. Structural Coherence Assessment

The structural coherence analysis reveals intriguing patterns across both connectivity ratios and KL divergence metrics. Connectivity ratios show remarkable consistency across all strategies up to about 1500 blocks, clustering around a median of 0.10, though the height priority strategy occasionally achieves higher peaks up to 0.20. Beyond 1500 blocks, height priority shows noticeably wider confidence bands, suggesting less predictable connectivity in larger structures. Entropy-based and from_center strategies maintain the most consistent variance patterns throughout the volume range, with their violin plots showing compact interquartile ranges. This consistency suggests that all strategies maintain basic structural integrity effectively, though with varying degrees of predictability. The KL divergence analysis paints a more varied picture. The height priority strategy exhibits the highest median divergence around 18 with substantial variance, displaying notably wide confidence intervals and an extended violin plot distribution, indicating more abrupt transitions between layers. In contrast, entropy-based and from_center strategies maintain lower, more consistent divergence values around 14, with tighter confidence intervals suggesting more predictable pattern transitions. The random walk strategy occupies a middle ground, balancing between consistency and variation, though demonstrating increasing variance at larger volumes.

D. Vertical Organization

Vertical stratification emerges as perhaps the most distinctive differentiator between strategies. The entropy-based approach achieves the highest median stratification of approximately 0.15, though with notable variance in its distribution, showing moderate but consistent variance throughout with a

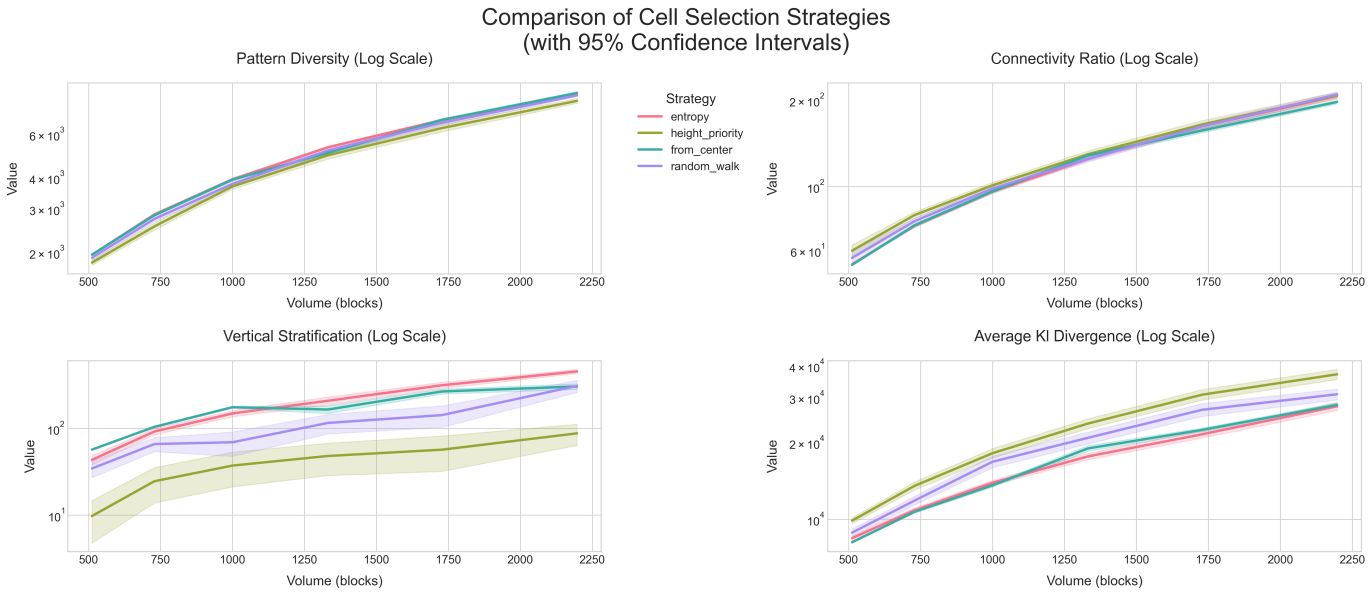

Fig. 6. Performance comparison of cell selection strategies across key metrics: pattern diversity (top left), connectivity ratio (top right), vertical stratification (bottom left), and average KL divergence (bottom right).

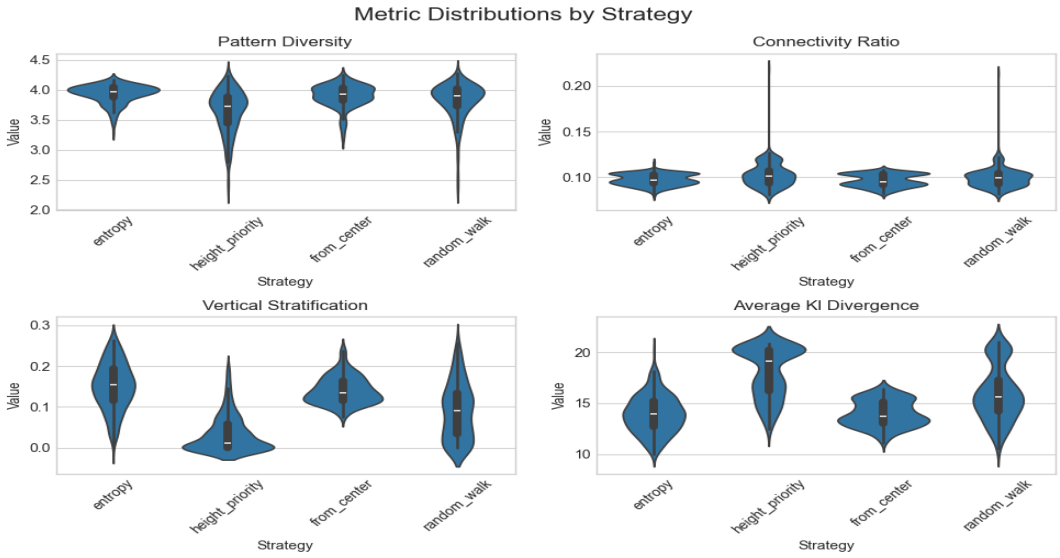

Fig. 7. Distribution of key performance metrics (Pattern Diversity, Connectivity Ratio, Vertical Stratification, and Average KL Divergence) across different WFC cell selection strategies. White dots indicate medians, black bars show interquartile ranges, and violin width represents data density at each value. Data collected from 240 generation runs per strategy.

symmetric violin plot distribution. This variance suggests flexibility in vertical development while maintaining overall coherence. The height priority strategy, true to its design philosophy, shows lower stratification values around 0.05 with remarkably low variance, demonstrated by tight confidence intervals and a compact violin plot distribution, reflecting its systematic bottom-up approach to construction. The temporal analysis reveals a critical threshold around 1500 blocks where strategies begin to diverge significantly in their vertical organization. The entropy-based strategy maintains consistent growth in stratification, while the height priority approach shows limited vertical variation even at larger scales. Particularly interesting is the random walk strategy's behaviour in larger structures, where it demonstrates increasing vertical stratification and variance, particularly evident in the diverging confidence bands after 1750 blocks, suggesting emergent complexity at scale. The from_center strategy exhibits the most stable variance pattern in larger structures, maintaining consistent confidence intervals even at maximum volume.

E. *Implications for Minecraft Building Generation*

Each strategy reveals distinct characteristics that make it suitable for specific architectural objectives. The entropy-based strategy emerges as the most versatile, offering an excellent balance of pattern diversity and structural coherence while maintaining consistent performance across scales. This makes it particularly suitable for general-purpose building generation where adaptability and variety are prioritized.

The height priority strategy, while more constrained in its pattern diversity and vertical variation, excels in creating structurally sound and predictable buildings. Its consistent connectivity ratios and systematic approach make it ideal for traditional architectural styles where stability and regularity are paramount.

The from_center strategy achieves an admirable balance across all metrics, with particularly strong performance in pattern diversity and local coherence. This balance makes it especially suitable for generating centrally-organized structures that require both symmetry and variety.

The random walk strategy's intermediate performance across metrics, combined with its unique scaling behaviour, positions it as an excellent choice for generating more naturalistic structures. Its increasing vertical stratification in larger structures suggests an ability to create more organic, emergent architectural forms.

These findings provide valuable guidance for procedural generation systems, suggesting that optimal results might be achieved through strategic selection of generation approaches based on specific architectural goals and project requirements. The analysis also reveals the potential value of hybrid approaches that could leverage different strategies at different scales or for different aspects of the same structure.

VII. Limitations and Future Work

While this study provides valuable insights into cell selection strategies for WFC-based Minecraft building generation, several limitations warrant attention and suggest directions for future research. The current implementation's restriction to 13x13x13 block structures due to computational constraints limits our understanding of how these strategies perform at larger scales; future work should explore parallel processing techniques and hierarchical generation approaches to enable the analysis of more expansive structures. The manually defined adjacency rules, while functional, represent a rigid interpretation of architectural validity that cannot adapt to diverse building styles; this could be addressed through the development of machine learning systems capable of extracting and validating building patterns from player-created structures. Our evaluation of buildings in isolation, without considering environmental context, represents another significant limitation; future research should extend the algorithm to consider terrain features, neighboring structures, and settlement-level coherence. The current metrics, while quantitatively robust, may not fully capture subjective aspects of architectural quality; this gap could be addressed through comprehensive user studies and the development of new metrics that better reflect aesthetic and functional success from a player's perspective. The fixed set of 25 structural elements limits the architectural possibilities; future work should explore methods for automatically expanding and validating the pattern set. Additionally, the static nature of each selection strategy suggests the potential for developing hybrid approaches that dynamically switch between strategies based on local context and building requirements. These improvements, combined with the integration of deep learning techniques for style transfer and pattern prediction, could significantly advance the capabilities of procedural building generation in Minecraft, leading to more diverse, contextually appropriate, and aesthetically pleasing structures.

VIII. Conclusion

This research evaluated four distinct cell selection strategies within the Wave Function Collapse algorithm for Minecraft building generation, revealing how different approaches influence architectural outcomes. Our comparative analysis shows that each strategy offers unique advantages: entropy-based selection provides the best balance of pattern diversity and coherence, height priority excels at structural stability, center-based generation creates well-organized symmetric buildings, and random walk produces more organic structures. The quantitative metrics developed demonstrate that selection strategies significantly impact both statistical properties and architectural characteristics of generated buildings, with pattern diversity reaching 7×10^3 at maximum volume for the most effective strategies. These findings provide practical guidance for procedural generation systems, suggesting that optimal results may be achieved by selecting strategies based on specific architectural goals. The framework developed here, combining architectural principles with algorithmic generation, offers a foundation for creating more sophisticated procedural content generation systems that can adapt to varying requirements while maintaining structural coherence. As games continue to demand larger and more diverse virtual worlds, these insights contribute to the development of more effective procedural generation techniques.

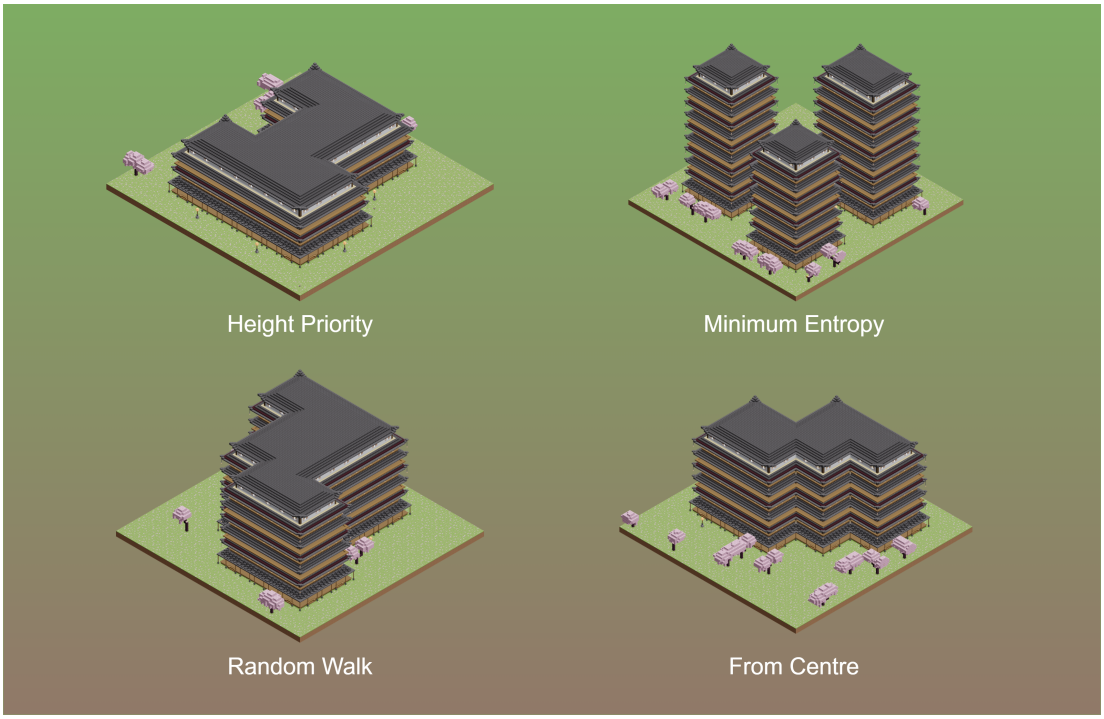

Fig. 8. A closer look at each cell selection strategy at work

Fig. 9. An in-game look at the WFC code in action

REFERENCES

[1] M. Gumin. (2016, Sep.) Wave function collapse algorithm. Version 1.0. [Online]. Available: https://github.com/mxgmn/WaveFunctionCollapse

[2] I. Karth and A. M. Smith, "Wavefunctioncollapse: Content generation via constraint solving and machine learning," *IEEE Transactions on Games*, vol. 14, no. 3, pp. 364–376, 2021.

[3] T. Short and T. Adams, *Procedural generation in game design*. CRC Press, 2017.

[4] S. Rabin, *AI Game Programming Wisdom 3 (Game Development Series)*. Charles River Media, Inc., 2006.

[5] J. Togelius, N. Shaker, and M. J. Nelson, "The search-based approach," *Computational Synthesis and Creative Systems, Cham, Springer*, pp. 17–30, 2016.

[6] A. Summerville, S. Snodgrass, M. Guzdial, C. Holmgård, A. K. Hoover, A. Isaksen, A. Nealen, and J. Togelius, "Procedural content generation via machine learning (pcgml)," *IEEE Transactions on Games*, vol. 10, no. 3, pp. 257–270, 2018.

[7] D. Perez-Liebana, S. Samothrakis, J. Togelius, T. Schaul, and S. Lucas, "General video game ai: Competition, challenges and opportunities," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016.

[8] C. Salge, M. C. Green, R. Canaan, and J. Togelius, "Generative design in minecraft (gdmc) settlement generation competition," in *Proceedings of the 13th International Conference on the Foundations of Digital Games*, 2018, pp. 1–10.

[9] S. S. Christiansen and M. Scirea, "Space segmentation and multiple autonomous agents: a minecraft settlement generator," in *2022 IEEE Conference on Games (CoG)*. IEEE, 2022, pp. 135–142.

[10] K. Daniel, *Thinking, fast and slow*, 2017.

[11] A. Iramanesh and M. Kreminski, "Agentcraft: An agent-based minecraft settlement generator," 2021.

[12] M. C. Green, C. Salge, and J. Togelius, "Organic building generation in minecraft," in *Proceedings of the 14th International Conference on the Foundations of Digital Games*, 2019, pp. 1–7.

[13] G. Smith, J. Whitehead, and M. Mateas, "Tanagra: Reactive planning and

constraint solving for mixed-initiative level design," *IEEE Transactions on computational intelligence and AI in games*, vol. 3, no. 3, pp. 201–215, 2011.

[14] I. Horswill, "Imaginarium: A tool for casual constraint-based pcg," in *Proceedings of the AIIDE Workshop on Experimental AI and Games (EXAG)*, 2019.

[15] N. Poldervaart, "GDMC 2024 - Vault of Immeasurable Knowledge." [Online]. Available: https://github.com/Niels-NTG/GDMC2024

[16] C. E. Shannon, "A mathematical theory of communication," *The Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.

[17] M. Hendrikx, S. Meijer, J. Van Der Velden, and A. Iosup, "Procedural content generation for games: A survey," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 9, no. 1, pp. 1–22, 2013.

[18] S. M. Lucas and V. Volz, "Tile pattern kl-divergence for analysing and evolving game levels," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2019, pp. 170–178.