A Markovian Framing of WaveFunctionCollapse for Procedurally Generating Aesthetically Complex Environments

Franklin Yiu[1], Mohan Lu[1], Nina Li[1], Kevin Joseph[1], Tianxu Zhang[1], Julian Togelius[1], Timothy Merino[1] and Sam Earle[1]

[1]*New York University, United States*

Abstract

Procedural content generation often requires satisfying both designer-specified objectives and adjacency constraints implicitly imposed by the underlying tile set. To address the challenges of jointly optimizing both constraints and objectives, we reformulate WaveFunctionCollapse (WFC) as a Markov Decision Process (MDP), enabling external optimization algorithms to focus exclusively on objective maximization while leveraging WFC's propagation mechanism to enforce constraint satisfaction. We empirically compare optimizing this MDP to traditional evolutionary approaches that jointly optimize global metrics and local tile placement. Across multiple domains with various difficulties, we find that joint optimization not only struggles as task complexity increases, but consistently underperforms relative to optimization over the WFC-MDP, underscoring the advantages of decoupling local constraint satisfaction from global objective optimization.

Keywords

WaveFunctionCollapse, Markov Decision Process, Evolution, PCG

1. Introduction

Procedural Content Generation (PCG) automatically creates game content through algorithmic methods often leveraging search or machine learning-based methods (PCGML). Learning-based methods in particular have enabled the automated creation of game content that optimizes designer-specified high-level objectives [1, 2, 3, 4, 5] that would be difficult to express or satisfy through traditional algorithms such as WaveFunctionCollapse (WFC) [6]. However, many existing learning-based PCG approaches are primarily applied to visually simplistic domains where the tile set imposes limited adjacency constraints [1, 2, 3, 4]. We suspect that even methods which already produce satisfactory results in a domain with more aesthetic adjacency constraints like Mario [7, 8, 9], would achieve higher robustness and efficiency from offloading the learning of the adjacency rules. We hypothesize that optimization-based PCG methods struggle with complex aesthetic adjacency rules due to the combinatorial explosion of valid tile configurations. To test this hypothesis, we compare traditional approaches that must learn these constraints implicitly against our novel formulation that leverages WFC's constraint propagation mechanism.

While WFC is inherently limited in its ability to optimize global functional properties such as playability, it excels at enforcing fine-grained aesthetic coherence by propagating preferences over possible tile placements via constraint solving [10, 11]. By reformulating WFC as a Markov Decision Process (MDP) and embedding an explicit objective function within this framework, we enable a more generalizable approach to guiding content generation toward designer-specified goals while simultaneously maintaining adherence to aesthetic adjacency constraints.

We optimize this MDP by evolving an action sequence using $\mu + \lambda$ evolution. To understand the implications of the MDP's explicit adjacency constraint guarantees, we compare against evolving the

Joint AIIDE Workshop on Experimental Artificial Intelligence in Games and Intelligent Narrative Technologies, November 10-11, 2025, Edmonton, Canada.

fyy2003@nyu.edu (F. Yiu); ml7612@nyu.edu (M. Lu); nl2539@nyu.edu (N. Li); kj2676@nyu.edu (K. Joseph); tz2902@nyu.edu (T. Zhang); julian@togelius.com (J. Togelius); tm3477@nyu.edu (T. Merino); sam.earle@nyu.edu (S. Earle)

tiles of the final artifact directly, via both naive $\mu + \lambda$ evolution and Feasible Infeasible 2-Population (FI-2Pop) [12], where the optimization algorithm must implicitly learn to resolve adjacency violations.

We evaluate these optimization strategies across 3 domains. The binary domain, adapted from Khalifa et al., imposes desired path length constraints as a simplified proxy for functional playability. Notably, varying the target path length allows us to dynamically adjust problem difficulty, enabling a systematic evaluation of how each optimization algorithm scales with increasing task complexity. On the other hand, the biome domains define objective functions that capture global topographic features of distinct biomes, guiding optimization toward semantically coherent and visually pleasing patterns. Finally, we integrate both binary and biome objectives, requiring solutions that jointly optimize metrics pertaining to form and function, thereby reflecting the multi-objective demands typical of real-world game design.

Across all domains and desired path lengths, non-MDP methods (which incorporate WFC-style constraints into objective functions instead of leveraging them via an MDP) perform noticeably worse; evolving the MDP action sequence leads to faster and more consistent convergence. However, on the most difficult objectives, evolving the action sequence converges very inconsistently, hinting at exploration limitations of $\mu + \lambda$ evolution.

Concretely, we offer the following contributions:

- We demonstrate that forcing learning algorithms to learn local adjacency constraints leads to degraded performance in highly constrained domains.
- We present a novel formulation of WFC as an MDP, along with a corresponding `gym` [13] environment to facilitate the evaluation of alternative optimization algorithms.

2. Related Work

2.1. WaveFunctionCollapse Algorithm and Modifications

2.1.1. WaveFunctionCollapse

WFC is an algorithm for procedural content generation, particularly for tile-based environments. It can create coherent and visually consistent outputs based on simple input examples or constraints [14]. The algorithm's core functionality resembles constraint satisfaction with a quantum mechanics-inspired approach: maintaining a superposition of possible states for each tile that gradually "collapses" to definite states through observation and propagation steps [15].

The algorithm operates in two main modes: the simple tiled model, which uses explicit adjacency rules, and the overlapping model, which automatically extracts patterns from example inputs [16]. In the both models, WFC can be seen as a form of self-supervised learning that learns a distribution from minimal examples—often just one—and generates similar content by maintaining the learned adjacency patterns. While WFC excels at maintaining local adjacency constraints, it struggles with global optimization objectives that are critical for gameplay, such as ensuring level solvability or balanced resource distribution [10].

2.1.2. Enhancements to WaveFunctionCollapse

Researchers have developed numerous modifications to improve WFC's capabilities beyond its original formulation. Karth and Smith provide a comprehensive analysis of WFC as a constraint satisfaction problem, establishing its theoretical foundations and positioning it within the broader context of PCG approaches.

To improve scalability to large environments, Nie et al. proposed Nested Wave Function Collapse (N-WFC), which decomposes the generation process into nested subproblems. This approach significantly reduces time complexity from exponential to polynomial while maintaining consistency across the generated content. For non-grid environments, researchers have developed graph-based extensions of WFC that enable content generation for arbitrary topologies, enabling applications to 3D worlds and non-uniform structures [10]. LNU introduce additional extensions to WFC including the path

constraint, which enforces global connectivity between specified tiles—addressing one of WFC's key limitations regarding global structure control. This path constraint is distinct from the path-length objective optimized in the present work in that it focuses on paths between designer-specified tiles as opposed to any two most distant tiles on the map, and only ensures the *existence* of such a path, rather than making guarantees about its length. This distinction is significant because optimizing path length requires global reasoning about the entire map structure, which traditional WFC cannot perform. While the path constraint ensures connectivity exists, our work specifically optimizes the longest shortest path between any two points, creating a more challenging optimization problem that must balance local tile placement decisions with their global impact on path topology. [1] Cheng et al. incorporates designer-driven heuristics to steer the generation process toward specific aesthetic or functional outcomes by introducing an automatic rule system along with global, multi-layer, and distance constraints to provide designers with more control over level layouts [6]. Yet the approach hinges on enumerated, non-local constraints—tile caps and anchored cells, hard distance windows among entities, and layer-locking of assets—that must be re-engineered for each domain, limiting generality compared with an objective-based formulation that subsumes these preferences in a unified reward.

2.2. Evolutionary Approaches to PCG

2.2.1. Evolutionary Algorithms

Evolutionary algorithms, widely applied to PCG problems, are a flexible approach to searching the space of possible game content while optimizing for designer-specified objectives [20]. These approaches typically encode content as genomes that evolve through operations such as crossover and mutation, with fitness functions guiding the search toward desired properties.

2.2.2. FI-2Pop for Constrained Optimization

The Feasible-Infeasible Two-Population (FI-2Pop) genetic algorithm [12], is a solution to constrained optimization problems that maintains two separate populations: one of feasible solutions that optimize the objective function, and another of infeasible solutions that minimize constraint violations. This approach has proven particularly effective for PCG applications where content must simultaneously satisfy hard constraints (e.g., playability) while optimizing soft objectives (e.g., challenge or balance).

Sorenson and Pasquier applied FI-2Pop to procedural level generation, creating game maps that satisfy playability constraints while optimizing for designer preferences. Later, Liapis et al. extended FI-2Pop for constrained novelty search in game level design. This approach uses novelty metrics rather than objective functions to drive search, resulting in a broader exploration of the feasible design space.

2.3. Procedural Content Generation via Reinforcement Learning (PCGRL)

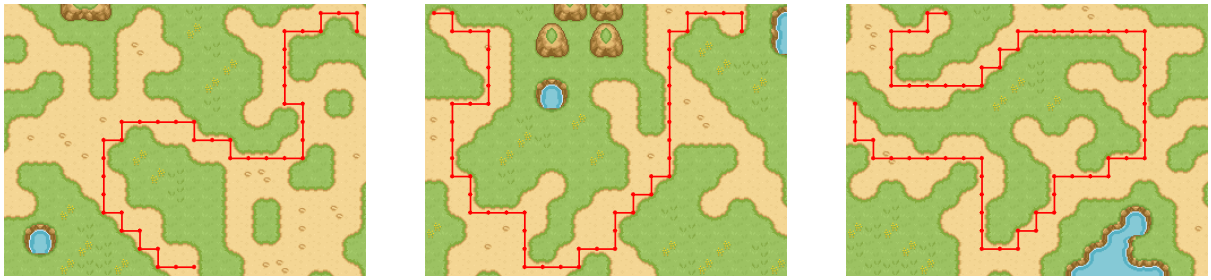
Khalifa et al. introduced PCGRL as a novel approach to procedural content generation that frames level design as a sequential decision-making process optimized through reinforcement learning. By formulating level generation as a Markov Decision Process (MDP), PCGRL enables an agent to learn a policy that maximizes expected level quality through incremental modifications. This approach can operate without human-authored examples and generates content extremely quickly once trained.

3. Problem Domain

All maps and thus objective functions were constructed based on a small subset of *Biome Tileset Pack B - Grassland, Savannah, and Scrubland* [24] which offers combinations of different grass, path, and

[1] We note that future work could attempt to optimize path-lengths between specific tiles with a simple modification to our path-length objective function.

Figure 1: Biome Tileset Pack B - Grassland, Savannah, and Scrubland. Unused tiles are darkened. Path tiles are marked in orange, grass tiles are marked in green, water tiles are marked in blue, and hill tiles are marked in brown. The water center tile is marked in light blue.



(a) Path Length: 40 (b) Path Length: 50 (c) Path Length: 60

Figure 2: Optimizing for target path-lengths in the Binary domain. The red line shows the longest shortest path.

water tiles. We used a subset of the tiles shown in Figure 1. We generate the adjacency rules via manual human labeling, though they could also be extracted from an input image.

3.1. Binary

The binary domain (Figure 2), originally introduced in PCGRL [3], tasks the generator with modifying an existing map composed of solid and empty tiles to increase the longest shortest path between any two empty tiles by at least 20 tiles, while ensuring full connectivity of the empty space. In contrast, our formulation requires generating valid maps entirely from scratch that satisfy prescribed path length constraints, rather than modifying preexisting layouts. We also impose a stricter requirement, penalizing deviations from the target by requiring the generated map to achieve a path length of exactly P. Formally, for a generated map with longest shortest path length p, the objective function awards a

score of $-|p - P|$. While the binary domain in PCGRL comprises a simple "binary" tile-set of wall and empty tiles, we use a more sophisticated tileset in which adjacent tiles—e.g. path and grass tiles—can share straight or rounded edges, creating visually smooth transitions between tile-types.

3.2. Biome/Binary Hybrid

The hybrid domain adds complexity by dictating the biome surrounding the binary path to better simulate the needs of real game maps. We consider 2 biomes which specify different distributions and arrangements of tiles. All percentage calculations are taken from the tiles of the final artifact, not including those used in binary path calculations. For example, if the objective specifies 50% water tiles, this means 50% of the tiles that are not path tiles must be water. We specify the biome objectives below.

- **River** (o_r).
  Let

  r_r the number of connected river regions,
  ℓ the length of the current river path,
  n_c the number of water "center" tiles,
  r_l the number of connected land regions.

  We then define:

$$\begin{aligned} o_r = (1 - r_r) &+ \min(0,\ \ell - 35) \\ &- n_c \\ &+ \min(0,\ 3 - r_l)\,. \end{aligned} \tag{1}$$

  The objective attains its maximum value of 0 exactly when all of the following hold:

  - $r_r = 1$ (exactly one contiguous river region),
  - $\ell \geq 35$ (river path length of at least 35 tiles),
  - $n_c = 0$ (no fully surrounded water tiles),
  - $r_l \leq 3$ (no more than three separate land regions).

  Here, the $(1 - r_r)$ term enforces a single connected channel, the $\min(0, \ell - 35)$ term rewards reaching the target length of 35 tiles, the $-n_c$ penalty encourages a thin, winding river (few interior water tiles), and the cap on r_l discourages excessive fragmentation of the land to prevent it from interrupting the river (Figure 3a).
- **Field** (o_f) .
  Let

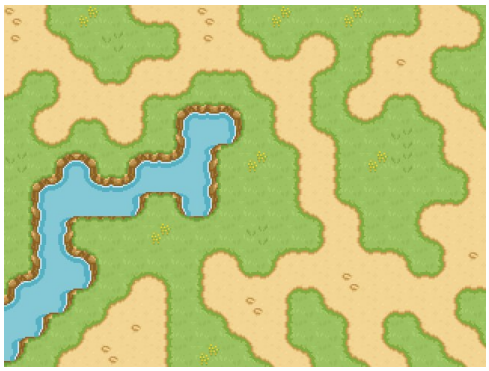
  n_w the number of water tiles,
  n_h the number of hill tiles,
  g the percent of grass tiles,
  f the percent of flower tiles.

  We then define:

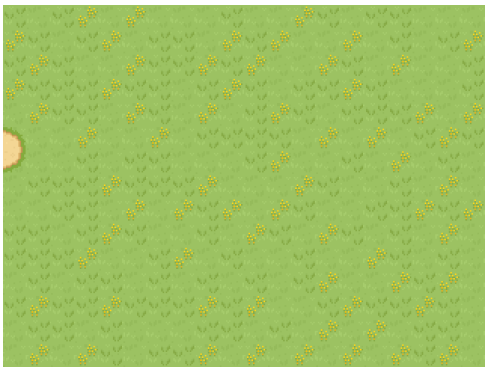
$$\begin{aligned} o_f = -n_w - n_h &+ \min(0,\ g - 20) \\ &+ \min(0,\ f - 20)\,. \end{aligned} \tag{2}$$

  The objective attains its maximum value of 0 exactly when all of the following hold:

  - $n_w = 0$ (no water or shore tiles),
  - $n_h = 0$ (no hill tiles),
  - $g \geq 20$ (at least 20% of tiles are grass),
  - $f \geq 20$ (at least 20% of tiles are flowers).



(a) River Biome



(b) Field Biome

Figure 3: Outputs resulting from the optimization of the Biome objectives

Here, the $-n_w$ and $-n_h$ penalties enforce a clear, unbroken grassy field, while the $\min(0,\ g-20)$ and $\min(0,\ f-20)$ terms ensure minimum coverage of grass and flowers. Together, these components encourage the generation of open grasslands with sufficient floral detail (Figure 3b).

Since all objective functions have a max reward of 0, we can enforce both binary and biome features by optimizing over the sum of the two objective functions. As such, we get the following objective functions:

- hybrid river/binary: $o_{rb} = -|p - P| + o_r$
- hybrid field/binary: $o_{fb} = -|p - P| + o_f$.

4. Methods

All optimization methods have various hyperparameters detailed in Section 9 of the Appendix.

4.1. Direct Map Evolution

These methods operate directly on the final artifact and do not leverage WFC. Instead, the optimization process must learn to satisfy the adjacency rules. For a target map of length ℓ and width w, the genotype is represented as a 2D array of size $\ell \times w$, where each entry contains an integer corresponding to a tile index in the tileset. For example, if position (x, y) contains value z, then the zth tile is placed at coordinate (x, y) in the artifact, irrespective of whether this placement violates adjacency constraints.

Baseline Evolution. The baseline evolutionary algorithm treats each map genotype as an individual and applies standard genetic operators with a penalized fitness that subtracts adjacency violations from raw objective function (Algorithm 1). Given objective score o and v adjacency violations, the individuals will receive a fitness of $o - v$. Over G generations it maintains a population of size N, selecting the top ρN individuals by fitness each round.

FI-2Pop. FI-2Pop [12] attempts to leverage adjacency violations as an exploration medium by maintaining two equal–sized subpopulations, feasible (F) and infeasible (I), and applies tailored selection criteria to each: objective maximization in F and violation minimization in I (Algorithm 2). Since I is not constrained by the objective function, it is free to explore the boundaries of infeasibility where optimality may lie. The offspring are generated separately to replenish each subpopulation to size $N/2$.

Algorithm 1 Baseline Evolution

Input: generations G, population size N, survival rate ρ
Output: Best genome found

Initialize population P with N random genomes
Each genome x has an objective score o_x, adjacency-constraint violation penalty v_x, and fitness f_x.
for $g = 1$ **to** G **do**
  for all genome $x \in P$ **do**
    $(o_x, v_x) \leftarrow \text{Evaluate}(x)$
    $f_x \leftarrow o_x - v_x$
  end for
  elites $\leftarrow$ top $\lceil \rho N \rceil$ genomes in P by f_x
  offspring $\leftarrow$ Reproduce(elites, $N - |\text{elites}|$)
  $P \leftarrow$ elites $\cup$ offspring
end for
return best feasible genome in P

Algorithm 2 FI-2Pop Evolution

Input: generations G, population size N, survival rate ρ
Output: Best genome found

Initialize population P with N random genomes
Initialize empty sets F and I
for $g = 1$ **to** G **do**
  for all genome $x \in P$ **do**
    $(o_x, v_x) \leftarrow \text{Evaluate}(x)$
    if $v_x = 0$ **then**
      $F \leftarrow F \cup \{x\}$
    else
      $I \leftarrow I \cup \{x\}$
    end if
  end for
  $F_s \leftarrow$ top $\lceil \rho |F| \rceil$ genomes in F by objective score (o_x)
  $I_s \leftarrow$ top $\lceil \rho |I| \rceil$ genomes in I by lowest violation (v_x)
  $O_F \leftarrow \text{Reproduce}(F_s, \frac{N}{2} - |F_s|)$
  $O_I \leftarrow \text{Reproduce}(I_s, \frac{N}{2} - |I_s|)$
  $P \leftarrow O_F \cup O_I$
end for
return best genome in F

4.2. MDP Representation.

By formalizing WFC as a Markov Decision Process (MDP), we leverage its guarantees to offload the burden of learning adjacency constraints from the optimizer. This reformulation transforms the generation problem into a sequential decision process where every action results in a valid intermediate configuration.

4.2.1. State Representation

We define each state s_t as the current configuration of the WFC grid at timestep t. For a target map of size $\ell \times w$ with n_t tile types, s_t is represented by an $\ell \times w \times n_t$ binary tensor G_t, where each depth slice encodes the feasibility of a tile at a cell:

- $G_t[x, y, i] = 1$ indicates that tile type i is currently *feasible* at cell (x, y).

Algorithm 3 WFC-MDP: One Environment Step

Input: belief grid $G \in \{0,1\}^{\ell \times w \times n_t}$, adjacency rules A, agent action (tile logits) a
Output: updated grid G', reward r

$(x^\star, y^\star) \leftarrow$ FindLowestEntropyCell(G)
$m \leftarrow G[y^\star, x^\star, :]$ *feasibility mask at selected cell*
if $(x^\star, y^\star) = \emptyset$ or $\sum_{t=1}^{T} m_t = 0$ **then**
  $r \leftarrow -1000$
end if
$p \leftarrow \text{softmax}(a)$
for all t $\in \{1, \ldots, n_t\}$ **do**
  if $m[t] = 0$ **then**
    $p[t] \leftarrow 0$
  end if
end for
$t^\star \leftarrow \arg\max p$
$G \leftarrow$ Collapse($G,\ (x^\star, y^\star) \leftarrow t^\star$)
$(G,\ \text{ok}) \leftarrow$ PropagateConstraints($G,\ A,\ (x^\star, y^\star)$)
if not ok **then**
  $r \leftarrow -1000$
else if IsFullyCollapsed(G) **then**
  $r \leftarrow$ Objective(G)
end if
return $(G,\ r)$

- $G_t[x, y, i] = 0$ indicates that tile type i is *infeasible* at cell (x, y).

A cell is *collapsed* if and only if it has a one-hot feasibility vector ($\sum_{i=1}^{n_t} G_t[x, y, i] = 1$); otherwise it is *uncollapsed* ($\sum_{i=1}^{n_t} G_t[x, y, i] > 1$). The MDP terminates when every cell is collapsed (equivalently, when all $\ell \times w$ cells have one-hot feasibility vectors), yielding the final artifact.

4.2.2. Action Representation

At each timestep t, the action a_t specifies which tile to collapse in a single uncollapsed cell (Algorithm 3). It is parameterized as an n_t-dimensional logit vector, with each entry corresponding to a possible tile; these logits lead to corresponding per tile probability after softmax. To ensure compliance with adjacency constraints, we mask out invalid tiles by setting their probabilities to zero. The selected action is then arg max over the probabilities, ensuring that each collapse step is constraint-respecting. Since exactly one tile is collapsed per action, a complete map requires a sequence of $\ell \times w$ actions.

4.2.3. Objective Structure

We adopt a sparse objective model: intermediate states contribute no score, and only the terminal state is evaluated using the objectives defined in Section 3. If WFC enters a contradiction where a cell has no valid tiles remaining, the process truncates immediately and incurs a large negative objective of -1000, thus discouraging invalid map constructions.

4.3. Evolving an Action Sequence

We use a standard $\mu + \lambda$ evolutionary algorithm to optimize the full sequence of WFC collapse actions (Algorithm 4). Each individual in the population encodes a fixed-length sequence of collapse decisions, represented as logits over the tile set at each of the $\ell \times w$ positions. Two genotype shapes are supported:

- **1D representation:** a flat sequence of length $\ell \times w$ such that element at position x is the action applied at time step x.
- **2D representation:** a grid-aligned sequence of dimensions $\ell \times w$, where each action's position is inferred from WFC's next-collapse coordinates. Action at genotype coordinate (x, y) corresponds to collapsing the tile at (x, y).

Algorithm 4 Evolving an Action Sequence

Input: generations G, population size N, survival rate ρ_s
Output: Best genome found

```
Initialize population P with N action sequences
for g = 1 to G do
  for all genome x ∈ P do
    s ← s_0; O_x ← 0; v_x ← 0
    for t = 1 to ℓ × w do
      a_t ← x[t]
      (s', r_t) ← env.step(a_t)
      O_x ← O_x + r_t
      s ← s'
    end for
  end for
  elites ← top ⌈ρ_s N⌉ genomes in P by f_x
  offspring ← Reproduce(elites, N − |elites|)
  Initialize population R with (N − |elites|) action sequences
  P ← elites ∪ offspring
end for
return best feasible genome in P
```

5. Results

We evaluate each optimization method across desired path lengths 10-100 in intervals of 10 for both binary and hybrid domains. Convergence robustness is defined as the proportion of runs that achieve the maximal reward of 0, while sample efficiency is measured by the number of generations it takes to evolve at least one population member with reward of 0. All methods were run with a fixed sample budget (population size = 48) to enable fair comparisons.

Table 1 reports performance across increasing path length targets in the binary domain. As expected, longer path lengths correspond to increased difficulty as all methods show reduced convergence rates and increased generations to convergence with rising desired path length P.

However, even at low difficulty ($P \leq 40$), non-MDP baselines occasionally fail to converge. In contrast, both MDP-based evolution strategies exhibit perfect or near-perfect convergence and require fewer generations. This early divergence suggests that directly modeling constraint satisfaction via WFC confers immediate robustness advantages, even in seemingly simple settings.

As P increases, the performance gap widens. At $P = 60$, non-MDP methods converge in less than 20% of runs and exhibit inefficiency when they do, with baseline requiring over 170 generations on average. Evolution 1D and 2D remain below 60 generations and achieve convergence in 84% and 72% of runs respectively, underscoring their superior scalability.

For $P \geq 70$, all non-MDP methods fail almost entirely. MDP methods, while less consistent, still achieve convergence in up to 35% of runs (1D) and 15% (2D). Interestingly, the baseline was able to converge once, in relatively few generations. This outlier suggests a rare, favorable initialization or fortuitous stochasticity early on in the evolutionary process rather than systematic optimization success.

In the hybrid river/binary domain (Table 2), MDP-based methods again demonstrate clear robustness advantages. For moderate difficulty ($20 \leq P \leq 40$), 1D and 2D evolution achieve convergence in at least 36% and 45% of runs respectively, requiring roughly 50–100 generations on average. In contrast, baseline and fi2pop converge in at most 14% and 4% of runs, and when they do succeed, require over 80–120 generations. At $P = 50$, both MDP methods maintain non-trivial success rates (≥19%) with convergence typically within 80–100 generations, whereas baseline converges only 4% of the time and fi2pop fails entirely. Beyond this threshold, only MDP methods achieve any convergence at all: both 1D and 2D occasionally succeed for $P \geq 60$, though with growing variance in required generations, while non-MDP methods collapse to zero convergence. The performance gap is more stark in Table 3 and the hybrid field/binary domain. Non-MPD methods completely fail to converge across almost all target path lengths.

Method	Metric	10	20	30	40	50	60	70	80
evo 1d	Generations	**5.9±0.6**	8.4±0.6	6.7±0.5	9.4±0.6	**21.9±1.4**	54.8±3.2	**114.7±10.4**	228.0±73.0
	Converged%	**1.00**	**1.00**	**1.00**	**1.00**	**0.97**	**0.84**	**0.35**	**0.02**
evo 2d	Generations	7.4±0.6	**6.2±0.5**	**5.8±0.5**	**9.1±0.8**	23.6±1.5	**41.9±2.8**	74.9±13.4	**136.0±74.0**
	Converged%	**1.00**	**1.00**	**1.00**	0.98	0.95	0.72	0.15	**0.02**
baseline	Generations	14.8±2.5	11.0±1.8	8.8±0.9	22.8±3.2	70.2±12.2	171.8±48.9	35	—
	Converged%	0.85	0.95	1.00	0.95	0.65	0.16	0.01	—
fi2pop	Generations	9.7±1.3	12.9±1.5	8.4±1.3	17.8±2.2	30.3±4.3	60.2±14.2	—	—
	Converged%	0.96	0.91	0.92	0.84	0.49	0.08	—	—

Table 1
Convergence performance across increasing binary path-length objectives (P). Each method is evaluated on convergence robustness (proportion of runs achieving optimal reward) and sample efficiency (mean generations to convergence with standard error). Results highlight the superior scalability of MDP-based approaches. Bold values indicate the highest convergence proportion and the lowest statistically significant mean generations per target path length. A visual representation can be found in the Appendix (Figure 5a).

5.1. Implications and Cross-Domain Insights

Two key patterns emerge:

1. **MDP Encapsulation of Constraints is Crucial**: Across all desired path lengths, methods that offload constraint enforcement to WFC consistently outperform those that must learn it implicitly. This discrepancy is especially pronounced given more difficult objectives (i.e. higher target path lengths, hybrid biome/binary domains), where the feasibility space is severely constrained.
2. **Feasible Region Shrinkage Limits Optimization**: At high path lengths, even MDP methods fail to converge reliably. This likely stems from the exponentially shrinking volume of the feasible space and limited tendency toward exploration in vanilla $\mu + \lambda$ evolution—as compared to e.g. Quality Diversity [25]. Despite valid intermediate states, the reward landscape remains highly sparse and multi-modal.

These findings highlight a fundamental insight: procedural generation under complex constraints benefits most when constraint satisfaction is externalized and search is guided through structurally aligned representations. The clear failure of joint optimization approaches, particularly in aesthetically constrained domains, emphasizes the importance of architectural modularity in generative design systems.

Method	Metric	10	20	30	40	50	60	70
evo 1d	Generations	62.0±6.5	59.7±6.0	77.3±5.7	98.5±10.4	78.7±9.5	**78.8±26.9**	—
	Converged%	**0.42**	**0.45**	**0.59**	**0.46**	**0.28**	**0.05**	—
evo 2d	Generations	**38.3±9.8**	49.7±6.0	**52.2±4.7**	**51.2±6.3**	**48.9±8.0**	92.0±37.0	**36**
	Converged%	0.11	**0.45**	0.42	0.36	0.19	0.03	**0.01**
baseline	Generations	98.5±11.5	89.6±18.7	114.6±25.9	111.7±22.3	70.7±13.0	—	—
	Converged%	0.03	0.09	0.12	0.14	0.04	—	—
fi2pop	Generations	40.5±39.5	48	136	82.3±40.8	—	—	—
	Converged%	0.03	0.01	0.01	0.04	—	—	—

Table 2
Convergence performance across increasing binary path-length objectives under the hybrid river/binary domain (P). Each method is evaluated on convergence robustness (proportion of runs achieving optimal reward) and sample efficiency (mean generations to convergence with standard error). Almost all experiments converging on less than 50% of runs, MDP base methods maintain a noticeable lead in both metrics. Bold values indicate the highest convergence proportion and the lowest statistically significant mean generations per target path length. A visual representation can be found in the Appendix (Figure 5b).

Method	Metric	10	20	30	40	50	60
evo 1d	Generations	66.8±13.9	**80.2±11.7**	155.1±24.4	344.4±37.6	551.7±104.2	472.0±214.0
	Converged%	**0.47**	**0.80**	**0.90**	**0.62**	**0.15**	**0.05**
evo 2d	Generations	59.5±9.1	94.9±15.0	**148.1±19.9**	**189.6±46.2**	**260**	**160.0±159.0**
	Converged%	0.35	0.39	0.30	0.15	0.01	0.03
baseline	Generations	—	—	—	—	—	—
	Converged%	—	—	—	—	—	—
fi2pop	Generations	27.0±10.5	—	—	—	—	—
	Converged%	0.04	—	—	—	—	—

Table 3
Convergence performance across increasing binary path-length objectives under hybrid the field/binary domain (P). Each method is evaluated on convergence robustness (proportion of runs achieving optimal reward) and sample efficiency (mean generations to convergence with standard error). Results highlight the superior scalability of MDP-based approaches, especially at higher difficulty levels. Bold values indicate the highest convergence proportion and the lowest statistically significant mean generations per target path length. A visual representation can be found in the Appendix (Figure 5c).

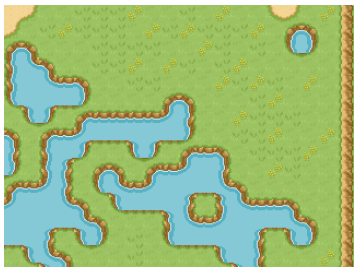
6. Discussion

6.1. Other Biomes

WFC-MDP is compatible with various domains without necessitating the bespoke engineering required by more traditional extensions to WFC. By changing the objective function, WFC-MDP is able to optimize for other gameplay artifacts like pond and hill biomes (Figure 4). These biomes were excluded from the main results as they were not adapted to work in conjunction with binary either because they proved excessively difficult to optimize or were not structured to allow long continuous paths; instead they illustrate the flexibility of WFC-MDP. We leave the optimization of these more challenging hybrid objectives to future work that builds on our WFC-MDP formulation.

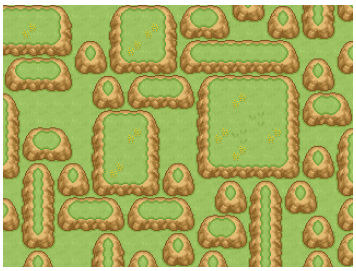
6.2. Representation Matters

We evaluated two encoding schemes for collapse sequences: a 2D representation that maps actions to fixed spatial locations, and a 1D representation that encodes a strict sequential order. The 2D encoding reduces the risk of early mutations cascading across the map by localizing genetic variation, resulting in smoother and more stable optimization dynamics. Conversely, the 1D encoding allows mutations in earlier steps to significantly influence subsequent tile collapses, which introduces instability but



(a) Pond Biome

(b) Pond and Hill Biome



(c) Hill Biome

Figure 4: Outputs resulting from the optimization of other Biome objectives

encourages broader exploration. Empirically, we observe that 1D encodings tend to converge more frequently possibly due to this exploratory behavior. In contrast, 2D encodings exhibit better sample efficiency, aligning with their more localized impact. These trade-offs suggest promising directions for future research, such as hybrid encoding schemes that balance exploration and stability.

6.3. Toward Scalable and Interactive Generation

Our method operates on consumer hardware, making it viable for map generation during the game development process. However, as problems get more complex, live evolution becomes time consuming. Further research can explore learning generalized policies that are capable of generating multiple artifacts from single training instance. For example, one could train a Reinforcement Learning policy to output/edit the tile-type logits in our 1D or 2D evolutionary genomes given varying target path-lengths as in [26], with reward equal to the score of a map collapsed via WFC over these logits. Alternatively, one could apply Imitation Learning to the data generated by the evolutionary processes in this paper, similar to [27], and/or use such Imitation Learning to jump-start the RL process outlined above. Future work could explore additional heuristics beyond path length optimization to test the generality of the WFC-MDP approach. Additionally, RL extensions could leverage the `gym` environment to test alternative optimization methods on the WFC-MDP formulation.

7. Limitations

7.1. Optimization Parameters

We used a fixed population size of 48 across all algorithms due to hardware constraints and to maintain a consistent measure of sample efficiency. This choice may not reflect the optimal configuration for each method. In particular, FI-2Pop's dual-population architecture may be disproportionately affected by smaller population sizes.

7.2. Observation Utilization

In an MDP formulation, the agent's observation can inform optimal action selection. Agents can make more optimal tile selections if given the partially collapsed map and the next collapse position. However, our evolution-based implementations operate over the full action sequence in advance and do not leverage intermediate observations. By instead evolving e.g. a neural network controller to output tile logits at each step of the WFC-MDP, these observations could be leveraged to improve performance.

7.3. Optimization Limitations

Although MDP-based methods demonstrate improved consistency and sample efficiency on challenging tasks, the standard $\mu + \lambda$ evolutionary strategy often exhibits inadequate exploration, leading to poor convergence on the hardest problems. We hypothesize that the large PCG state space and highly

multimodal fitness landscape demand algorithms with stronger exploration capabilities, such as Quality Diversity evolutionary algorithms [25] or Novelty Search [28].

8. Conclusion

This work recasts WaveFunctionCollapse (WFC) as a Markov Decision Process (MDP), enabling optimizers to sidestep the combinatorial burden of learning tile adjacency rules by offloading constraint enforcement to WFC's propagation mechanism. By evaluating this formulation across various scalable and progressively constrained domains, we uncover a central insight: explicit algorithmic decoupling of constraint satisfaction from objective optimization dramatically improves both convergence reliability and sample efficiency.

Our WFC-MDP framework not only succeeds where traditional methods collapse, but also reveals how reframing generation as a sequence of valid state transitions exposes a richer interface for guidance and learning. The domain's scalability allowed us to observe how even strong inductive biases (like constraint propagation) falter without sufficient exploration capacity, suggesting fertile ground for hybrid learning approaches.

Looking ahead, our formulation opens the door to more controllable and expressive uses of WFC. Because objective functions are more generalizable than hard-coded global constraints, they provide a natural bridge to machine learning methods, particularly those that rely on flexible reward signals or policy learning. By enabling WFC to support objective driven control over layout generation, this work establishes a blueprint for scalable, constraint-aware, and ML-compatible PCG systems.

9. Appendix

9.1. Final Hyperparameter Settings

Table 4 records our tuned hyperparameters.

		Optimization method			
Domain	**Parameter**	**Baseline**	**FI–2Pop**	**Action Seq (1D)**	**Action Seq (2D)**
Binary	number_of_actions_mutated_mean	89	162	97	44
	number_of_actions_mutated_standard_deviation	157.2498	196.1993	120.0876	28.2708
	action_noise_standard_deviation	0.0810	0.0418	0.1296	0.1409
	survival_rate	0.5211	0.3552	0.4151	0.2328
	cross_over_method	1 (ONE_POINT)	1 (ONE_POINT)	1 (ONE_POINT)	0 (UNIFORM)
	cross_or_mutate	0.8324	0.9871	0.7453	0.9557
Hybrid River/Binary	number_of_actions_mutated_mean	1	142	79	48
	number_of_actions_mutated_standard_deviation	56.7544	69.3442	111.7231	14.0969
	action_noise_standard_deviation	0.1452	0.0413	0.1087	0.0647
	survival_rate	0.7988	0.4726	0.4133	0.3077
	cross_over_method	1 (ONE_POINT)	1 (ONE_POINT)	1 (ONE_POINT)	0 (UNIFORM)
	cross_or_mutate	0.9633	0.9130	0.9930	0.8902
Hybrid Field/Binary	number_of_actions_mutated_mean	86	178	23	132
	number_of_actions_mutated_standard_deviation	146.9724	68.7875	0.5458	56.0667
	action_noise_standard_deviation	0.3916	0.0125	0.4937	0.0407
	survival_rate	0.7513	0.7720	0.1861	0.3245
	cross_over_method	0 (UNIFORM)	0 (UNIFORM)	1 (ONE_POINT)	1 (ONE_POINT)
	cross_or_mutate	0.6876	0.7570	0.8241	0.8415

Table 4
We employed Optuna [29] to automatically tune evolutionary hyperparameters for each of our three methods and across all experiments. For each method/experiment we ran 20 trials, optimizing the cumulative best reward after 20 training attempts with 100 generations each.

9.1.1. Hyperparameter Definitions

`number_of_actions_mutated_mean` (int) Expected number of genome actions to mutate each generation.

`number_of_actions_mutated_standard_deviation` (float) Standard deviation for truncated-normal sampling of the mutation count.

`action_noise_standard_deviation` (float) Standard deviation of Gaussian noise added to each mutated action's value.

`survival_rate` (float) Fraction of the population preserved into the next generation.

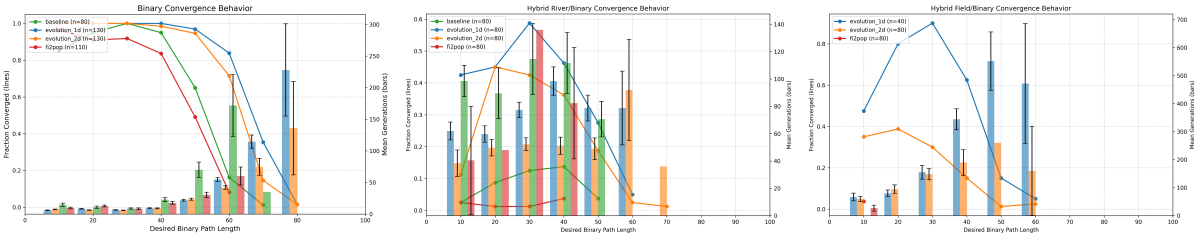
`cross_over_method` (enum) Crossover strategy:

- **0 = UNIFORM** Gene-wise mixing: each child gene is chosen from one parent with 50% probability.
- **1 = ONE_POINT** Single cut-point crossover: parents swap tails at one random index.

`cross_or_mutate_proportion` (float) Fraction of offspring produced via crossover.

9.2. Convergence Behavior Plots

Plots correlated with the tables in Section 5 (Figure 5).



(a) Plot of Table 1 (b) Plot of Table 2 (c) Plot of Table 3

Figure 5: Plot which serve as a visual representation of the convergence behavior expressed in a corresponding tables. The lines correlate with the fraction of converged training samples (left y axis — higher is better) and the bars correlate to the mean generations to successfully converge (right y axis — lower is better).

10. Declaration on Generative AI

There was no use of generative AI in this paper.

References

[1] G. Todd, S. Earle, M. U. Nasir, M. C. Green, J. Togelius, Level generation through large language models, in: Proceedings of the 18th International Conference on the Foundations of Digital Games, 2023, pp. 1–8.

[2] Y. Nie, M. Middleton, T. Merino, N. Kanagaraja, A. Kumar, Z. Zhuang, J. Togelius, Moonshine: Distilling game content generators into steerable generative models, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 39, 2025, pp. 14344–14351.

[3] A. Khalifa, P. Bontrager, S. Earle, J. Togelius, Pcgrl: Procedural content generation via reinforcement learning, in: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, volume 16, 2020, pp. 95–101.

[4] S. Earle, F. Kokkinos, Y. Nie, J. Togelius, R. Raileanu, Dreamcraft: Text-guided generation of functional 3d environments in minecraft, in: Proceedings of the 19th International Conference on the Foundations of Digital Games, 2024, pp. 1–15.
[5] A. Summerville, S. Snodgrass, M. Guzdial, C. Holmgård, A. K. Hoover, A. Isaksen, A. Nealen, J. Togelius, Procedural content generation via machine learning (pcgml), IEEE Transactions on Games 10 (2018) 257–270.
[6] D. Cheng, H. Han, G. Fei, Automatic generation of game levels based on controllable wave function collapse algorithm, in: Entertainment Computing–ICEC 2020: 19th IFIP TC 14 International Conference, ICEC 2020, Xi'an, China, November 10–13, 2020, Proceedings 19, Springer, 2020, pp. 37–50.
[7] H. J. Lee, E. Simo-Serra, Using unconditional diffusion models in level generation for super mario bros, in: 2023 18th International Conference on Machine Vision and Applications (MVA), IEEE, 2023, pp. 1–5.
[8] Z. Wang, J. Liu, G. N. Yannakakis, The fun facets of mario: Multifaceted experience-driven pcg via reinforcement learning, in: Proceedings of the 17th International Conference on the Foundations of Digital Games, 2022, pp. 1–8.
[9] T. Shu, J. Liu, G. N. Yannakakis, Experience-driven pcg via reinforcement learning: A super mario bros study, in: 2021 IEEE Conference on Games (CoG), IEEE, 2021, pp. 1–9.
[10] H. Kim, S. Lee, H. Lee, T. Hahn, S. Kang, Automatic generation of game content using a graph-based wave function collapse algorithm, in: 2019 IEEE conference on games (CoG), IEEE, 2019, pp. 1–4.
[11] I. Karth, A. M. Smith, Wavefunctioncollapse: Content generation via constraint solving and machine learning, IEEE Transactions on Games 14 (2021) 364–376.
[12] S. O. Kimbrough, G. J. Koehler, M. Lu, D. H. Wood, On a feasible–infeasible two-population (fi-2pop) genetic algorithm for constrained optimization: Distance tracing and no free lunch, European Journal of Operational Research 190 (2008) 310–327. URL: https://www.sciencedirect.com/science/article/pii/S0377221707005668. doi:https://doi.org/10.1016/j.ejor.2007.06.028.
[13] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, W. Zaremba, Openai gym, arXiv preprint arXiv:1606.01540 (2016).
[14] M. Gumin, Wave Function Collapse Algorithm, 2016. URL: https://github.com/mxgmn/WaveFunctionCollapse.
[15] I. Karth, A. M. Smith, Wavefunctioncollapse is constraint solving in the wild, in: Proceedings of the 12th International Conference on the Foundations of Digital Games, FDG '17, Association for Computing Machinery, New York, NY, USA, 2017. URL: https://doi.org/10.1145/3102071.3110566. doi:10.1145/3102071.3110566.
[16] R. Heaton, The wavefunction collapse algorithm explained very clearly, 2018. URL: https://robertheaton.com/2018/12/17/wavefunction-collapse-algorithm/.
[17] I. Karth, A. M. Smith, Wavefunctioncollapse: Content generation via constraint solving and machine learning, IEEE Transactions on Games 14 (2022) 364–376. doi:10.1109/TG.2021.3076368.
[18] Y. Nie, S. Zheng, Z. Zhan, X. Song, Extend wave function collapse algorithm to large-scale content generation, 2023 IEEE Conference on Games (CoG) (2023) 1–8. URL: https://api.semanticscholar.org/CorpusID:260886968.
[19] B. LNU, Wave function collapse tips and tricks, 2020. URL: https://www.boristhebrave.com/2020/02/08/wave-function-collapse-tips-and-tricks/.
[20] J. Togelius, G. N. Yannakakis, K. O. Stanley, C. Browne, Search-based procedural content generation: A taxonomy and survey, IEEE Transactions on Computational Intelligence and AI in Games 3 (2011) 172–186. doi:10.1109/TCIAIG.2011.2148116.
[21] N. Sorenson, P. Pasquier, Towards a generic framework for automated video game level creation, in: C. Di Chio, S. Cagnoni, C. Cotta, M. Ebner, A. Ekárt, A. I. Esparcia-Alcazar, C.-K. Goh, J. J. Merelo, F. Neri, M. Preuß, J. Togelius, G. N. Yannakakis (Eds.), Applications of Evolutionary Computation, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 131–140.
[22] A. Liapis, G. N. Yannakakis, J. Togelius, Constrained novelty search: A study on game content generation, Evolutionary Computation 23 (2015) 101–129. URL: https://doi.org/10.1162/

EVCO_a_00123. doi:10.1162/EVCO_a_00123. arXiv:https://direct.mit.edu/evco/article-pdf/23/1/101/1514786/evco_a_00123.pdf.

[23] A. Khalifa, P. Bontrager, S. Earle, J. Togelius, Pcgrl: procedural content generation via reinforcement learning, in: Proceedings of the Sixteenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, AIIDE'20, AAAI Press, 2020.

[24] VectoRaith, Biome tileset pack b - grassland, savannah, and scrubland, https://vectoraith.itch.io/biome-tileset-pack-b, 2025.

[25] J. K. Pugh, L. B. Soros, K. O. Stanley, Quality diversity: A new frontier for evolutionary computation, Frontiers in Robotics and AI 3 (2016) 40.

[26] S. Earle, M. Edwards, A. Khalifa, P. Bontrager, J. Togelius, Learning controllable content generators, in: 2021 IEEE Conference on Games (CoG), IEEE, 2021, pp. 1–9.

[27] A. Khalifa, J. Togelius, M. C. Green, Mutation models: Learning to generate levels by imitating evolution, in: Proceedings of the 17th International Conference on the Foundations of Digital Games, 2022, pp. 1–9.

[28] J. Lehman, K. O. Stanley, Evolving a diversity of virtual creatures through novelty search and local competition, in: Proceedings of the 13th annual conference on Genetic and evolutionary computation, 2011, pp. 211–218.

[29] T. Akiba, S. Sano, T. Yanase, T. Ohta, M. Koyama, Optuna: A next-generation hyperparameter optimization framework, in: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining, 2019, pp. 2623–2631.