Towards Wave Function Collapse using Optimization with Quantum Algorithms

Zirui Zhang
Purdue University
United States of America
zhan4192@purdue.edu

Shannon Cheng
Purdue University
United States of America
cheng652@purdue.edu

Kevin Xiao
Purdue University
United States of America
xiao278@purdue.edu

Priyam Gupta
Purdue University
United States of America
gupta751@purdue.edu

Sean Ruda
Purdue University
United States of America
sruda@purdue.edu

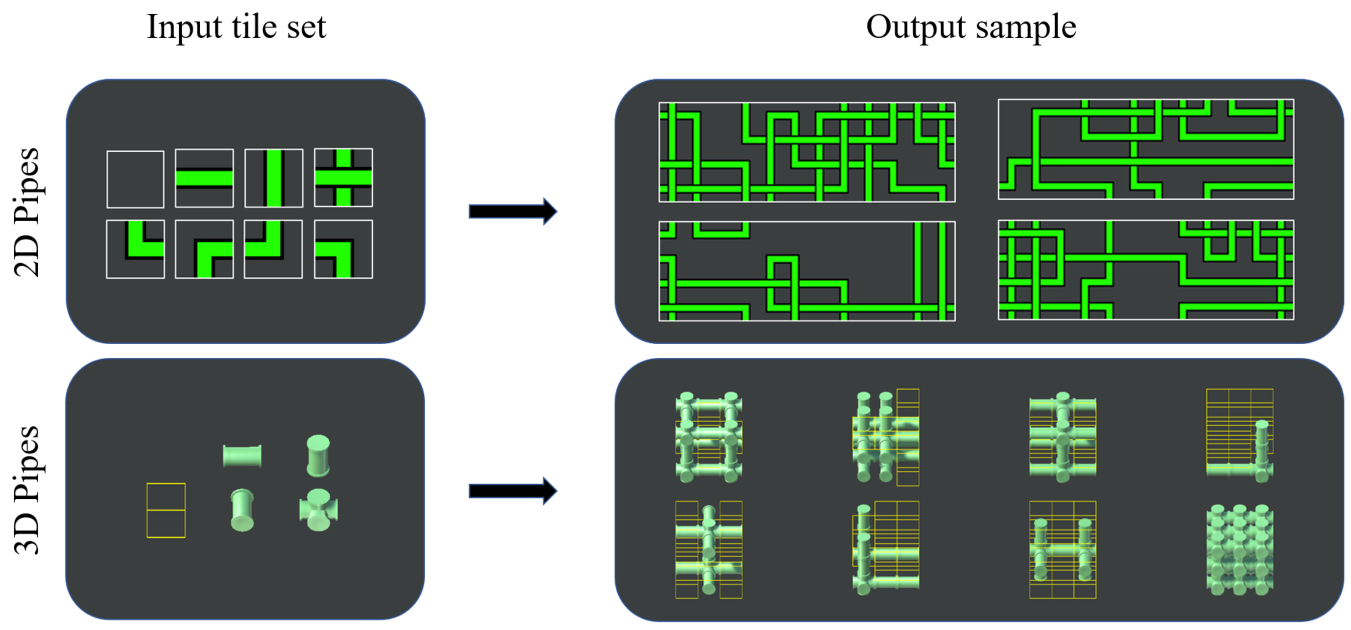

Figure 1: Selection of input tileset with respective generated output samples.

Abstract

Quantum computing has emerged in recent years as a rapidly evolving field, presenting the potential to solve problems that are difficult for classical computers. Our work presents a new implementation of procedural generation using Wave Function Collapse (WFC), solved using Quantum Approximate Optimization Algorithm (QAOA). The problem is formatted as a Constraint Satisfaction Problem (CSP), where the edge rules in classical WFC are encoded as linear constraints. We successfully demonstrate the generation of valid tile configuration in both 2D and 3D, showcasing the potential of quantum computing in hybrid quantum-classical systems. However, our algorithm is constrained by limitations set by current quantum computers, which prevents real-world deployment at this stage.

SA Technical Communications '24, December 03–06, 2024, Tokyo, Japan
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1140-4/24/12
https://doi.org/10.1145/3681758.3698015

CCS Concepts

• **Computing methodologies** → *Graphics systems and interfaces.*

Keywords

Procedural generation, Quantum computing, Optimization

ACM Reference Format:
Zirui Zhang, Shannon Cheng, Kevin Xiao, Priyam Gupta, and Sean Ruda. 2024. Towards Wave Function Collapse using Optimization with Quantum Algorithms. In *SIGGRAPH Asia 2024 Technical Communications (SA Technical Communications '24), December 03–06, 2024, Tokyo, Japan.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3681758.3698015

1 Introduction

Quantum computing has emerged as a field providing potential acceleration to classical computing, demonstrating promising speedups with capabilities to handle probabilistic computation. Unlike classical computers, quantum computers utilize quantum bits (qubits) that can exist simultaneously in multiple states, offering speedups, demonstrated by algorithms such as Grover's for unstructured search [Grover 1996] and Shor's [Shor 1997] for factoring prime numbers.

One potential application of quantum computing lies in procedural generation algorithms. One such algorithm, Wave Function Collapse (WFC), is utilized for pattern generation by dividing up an input tileset into small chunks, which is then rearranged for content generation [Gumin 2016]. It functions by collapsing a superposition of possible configurations to produce an output based on local constraints, often setting the generation ruleset in the form of tile compatibility. Traditionally, WFC has been employed in game development and computer graphics to generate infinite patterns while adhering to predefined rules.

However, the classical implementation does not guarantee a solution, as demonstrated in Fig. 2. Since the algorithm only considers one tile at a time during generation, without backtracking, it has the potential to generate a configuration with incomplete solutions as the algorithm does not revisit old tile placements [Karth and Smith 2017].

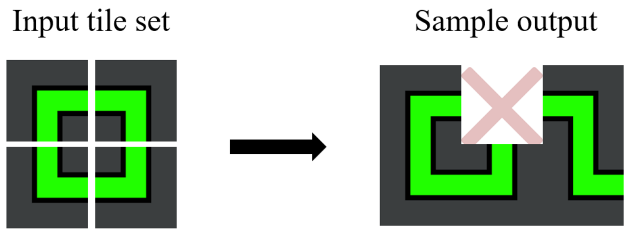

Figure 2: Classical wave function collapse with incomplete tileset, assuming no tile rotation, producing incomplete solution.

On the other hand, quantum computing can present a new hybrid approach for the WFC algorithm with its probabilistic nature and ability to explore multiple potential solutions simultaneously. Coincidentally, the concept of a wave function and its ability to collapse from superposition also arises from quantum mechanics, leading to the natural consideration of solving it with quantum computers. While potential implementations have been explored in the past, results often indicate that encoding adjacency rules as circuits only provides theoretical feasibility and cannot currently be realized on actual quantum hardware [Heese 2024]. Our work presents a potential implementation that successfully generated content using an optimization solver on current quantum hardware.

Our work explores a hybrid implementation of WFC using Quadratic Unconstrained Binary Optimization (QUBO), often used to tackle combinatorial optimization problems [Glover et al. 2019], formulated as an optimization problem and optimized via Quantum Approximate Optimization Algorithm (QAOA) [Farhi et al. 2014]. By addressing the constraints of WFC with a quantum algorithm, we aim to propose a novel approach to procedural content generation while offering potential solutions to said constraints, demonstrating a new use case for quantum computing in computer graphics.

2 Method and Implementation

First, the problem must be formatted as a constraint-satisfaction problem (CSP). This approach has been implemented previously as a graph-based WFC, demonstrating the generation of valid configuration by considering all constraints collectively rather than in isolation [Kim et al. 2019]. Similarly, our approach formats the WFC algorithm as a CSP, with adjustments to variables and constraints. Quantum computers can be especially effective at certain optimization problems, demonstrating numerical speedup in certain cases[Golden et al. 2023], as they evaluate all possible configurations in superposition concurrently. Additionally, the probabilistic nature of quantum measurement allows for the generation of multiple solutions.

2.1 Variable and Objective Function

Reworked for quantum computers, each cell in the grid represents a variable corresponding to potential tile placement. Each variable is represented by a qubit in the system, where a measurement of 1 on a qubit indicates the presence of a specific tile in the cell, and 0 indicates its absence. Let i and j be the row and column indices of the grid, respectively, and t be the tile type. The variable $x_{i,j,t}$ takes the value 1 if tile type t is placed at position (i, j) on the grid and 0 otherwise. Similarly, in the context of a 3D grid, the variables can be expressed as $x_{i,j,k,t}$ where i, j, k are the coordinates of a cell in the 3D grid. The objective function is designed to minimize the sum of all variables, formulated as:

$$\text{Minimize} \quad Z = \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{t=1}^{T} x_{i,j,t} \tag{1}$$

where n and m are the dimensions of the grid, and T is the number of different tile types.

2.2 Constraints

Within the QUBO, constraints are formulated either as linear or quadratic terms due to the nature of binary quadratic forms. This limitation is intrinsic to the representation capabilities of QUBO, which is structured to handle expressions of the form:

$$\text{Minimize} \quad f(x) = \sum_{i=1}^{n} a_i x_i + \sum_{i=1}^{n} \sum_{j=1}^{n} b_{ij} x_i x_j \tag{2}$$

where x_i are binary variables ($x_i \in \{0, 1\}$) and a_i and b_{ij} are coefficients defining the linear and quadratic terms, respectively [Borle et al. 2021].

For our procedural generation model, we ensure each cell in the grid must contain exactly one tile. This can be expressed as a series of linear constraints to ensure that the sum of all tile variables in each cell equals one:

$$\sum_{t=1}^{T} x_{i,j,t} = 1 \quad \forall i, j \tag{3}$$

The adjacency rules between tiles are enforced through a set of constraints, ensuring that specific tiles can only be placed next to compatible tiles within the given rule set. Let $x_{i,j,t}$ denote the binary variable where t indicates the tile type at grid position (i, j). The general form of these constraints for an adjacency pair, where cell (i, j) has tile type t and an adjacent cell (i', j') can host compatible tiles t', is given by:

$$x_{i,j,t} - \left(\sum_{t' \in T_{adj}(t)} x_{i',j',t'} \right) + \left(\sum_{t'' \in T_{comp}(t)} x_{i,j,t''} \right) = 0 \tag{4}$$

Where $x_{i,j,t}$ is the binary variable for tile type t at cell (i, j), $T_{adj}(t)$ is the set of all tile types that can be placed adjacently to tile type t at cell (i', j'), and $T_{comp}(t)$ represents the set of other valid tile types that can exist in the context of t' at cell (i', j') based on adjacency rules.

In addition to adjacency constraints, similar to Gumin's original implementation, it's also possible to fix a specific tile at a certain position within the grid. This can be accomplished through a simple equality constraint:

$$x_{i,j,t} = 1 \tag{5}$$

which ensures that tile type t is placed at position (i, j), enforcing fixed tile placements at this location. This constraint is used prior to optimizing the problem, often as a method to create output with desired configurations.

2.3 QAOA Solver

The Quantum Approximate Optimization Algorithm (QAOA) is a hybrid quantum-classical algorithm designed to solve combinatorial optimization problems. In this section, we give a brief overview of the usage of this algorithm. This algorithm encodes the problem as a quantum Hamiltonian (H_P) reflecting the QUBO's constraints and objective function. It begins with a uniformly distributed quantum state and applies a sequence of transformations and measurements.

Algorithm 1 Quantum Approximate Optimization Algorithm for QUBO

```
1:  Inputs: QUBO coefficients, number of qubits, depth $p$
2:  Output: Optimal or near-optimal binary string solution probability distribution
3:  Encode the QUBO into a problem Hamiltonian $H_P$
4:  Initialize parameters $\beta$ and $\gamma$ randomly
5:  Prepare an initial state $|\psi_0\rangle$ (uniform superposition of all states)
6:  for iteration = 1 to MaxIterations do
7:      Apply unitary transformations $U(H_P, \gamma)$ and $U(H_M, \beta)$ to $|\psi_0\rangle$
          - $H_M$ is the mixing Hamiltonian
          - Each unitary is controlled by the parameters $\beta$ and $\gamma$
8:      Measure the quantum state to obtain a binary string
9:      Evaluate the quality of the binary string using the QUBO
10:     Use a classical optimizer to adjust $\beta$ and $\gamma$ based on outcomes
11: end for
12: return the binary string sample with the probability distribution
```

Each iteration refines the control parameters (β and γ) using classical feedback to progressively improve the solution quality [Koretsky et al. 2021]. In the context of solving a QUBO, the QAOA can be formulated as shown in Algorithm 1.

QAOA is designed to approximate the solution to a given CSP by adjusting the parameters β and γ to minimize the expected value of the problem Hamiltonian over the quantum states. Ideally, with sufficient iteration, the quantum state converges to one where the measurement outcome clusters around the optimal solution.

Upon measurement of the quantum states, the QAOA produces a distribution of solutions where each solution's probability correlates with its ability to satisfy the objective function and respective constraints. At this point, all quantum states have been collapsed into measurement distributions. With the given probability distribution of solutions, post-processing is applied, selecting the first x configurations with the highest probability while validating the solution with the original QUBO problem, ensuring it fits all constraints.

3 Experiment and Results

In this section, we provide a brief analysis of the input sample used and the results generated. For our experiment, we used Aer simulator, Rigetti Ankaa 2, and IBM Osaka, with IBM Qiskit as our primary programming library [Javadi-Abhari et al. 2024]. With the IBM Osaka quantum computer containing the most qubits, 127 total, we were able to handle up to 127 variables in the QUBO.

With a 2D tileset and grid, the QUBO is formulated following the objective function and constraints outlined in the previous section. For the 2D pipe tileset, it contains 8 tiles, requiring 8 qubits per cell on the grid. With one iteration of QAOA and post-processing, the algorithm can produce a 3 by 4 grid.

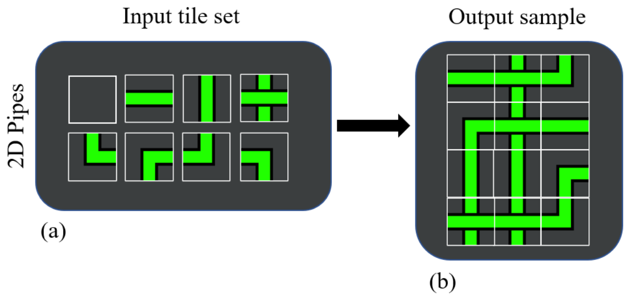

Figure 3: Pipe 2D tileset (a) with an output sample (b) being generated by Algorithm 1.

For larger grids, due to the qubit constraint on the quantum computers used, previously generated tiles can be used as additional constraints where one row or column can be fixed with the simple equality constraint defined in equation 5, demonstrated in Fig 4.

By using partial information from previous output, the algorithm is no longer constrained to the maximum allowed qubit in one iteration, mimicking the idea of enforcing local constraints in the original WFC algorithm. It should be noted that with each fixed constraint given by the previous run of QAOA, enough qubits should be reserved such that new rows or columns can generated with the fixed constraints.

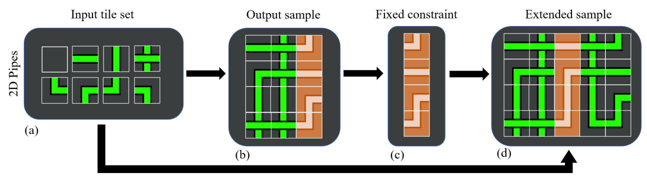

Figure 4: Previous sample's last column, highlighted in orange (c), used as constraints to generate an additional set of 3 by 4 grid (d), extending the original output sample (b) with the same input tileset (a).

Similarly, the algorithm can be extended to 3 dimensions as it only requires additional variables and constraints. Whereas each cell in a 2D grid only needs to check its four neighbors, in 3D, each cell must check against its six neighbors. While each constraint adds additional gate depth to the quantum circuit for the QAOA, it does not require additional qubits. Outputs can also be used as fixed constraints for additional row, column, or layer generation as long as they stay within the total qubit constraint per QAOA iteration.

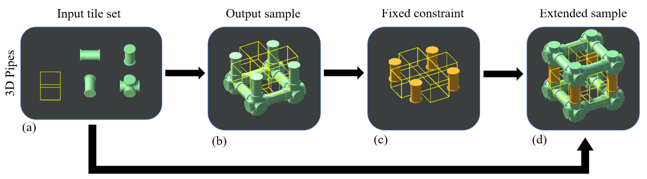

Figure 5: Given a 3D tileset (a), a 3 by 2 by 3 grid is generated (b). The second layer, highlighted in orange (c), is used to generate an extended version of the sample (d), added to the original output (b).

By feeding the algorithm with a set of 3D tilesets and constraints for each of its faces, creating 5 variables per cell, the algorithm successfully generates a sample of valid configuration. Due to qubit count constraints, only 18 cells were processed at a single time, requiring 90 qubits.

4 Conclusion and Future Outlook

We have successfully demonstrated the generation of coherent tile configuration in both 2D and 3D, reformulating the WFC algorithm as a QUBO optimization problem, solved through QAOA on both simulated quantum computers and real quantum computing hardware.

It should be noted that while our algorithm demonstrates an alternative approach to solving the WFC problem through quantum optimization, its practical application on current quantum hardware remains challenging. As the title of our work suggests, our implementation is not yet ready for real-world usage. All of the samples and results we provided in the previous section remained relatively small compared to the content WFC is currently being used to generate.

Current quantum computers are limited significantly by various factors such as quantum noise[Resch and Karpuzcu 2021] and decoherence time with increased gate-depth and circuit complexity [Saki et al. 2019]. These issues inherently limit the reliability and scalability of current quantum algorithms, making it difficult to incorporate them into various hybrid systems. Consequently, despite our algorithm producing the correct results, it is unlikely to be implemented into any classical systems in the near future.

With our work, we aim to not only present a quantum alternative to the WFC algorithm but also to inspire future research in the field of quantum computing usage for computer graphics. By demonstrating theoretical advantages over the classical approach, we hope to encourage the advancement of novel technology with practicality.

Acknowledgments

This work was supported by the National Defense Education Program (NDEP) under Grant No. HQ0034-21-1-0014. We would also like to thank Professor Mahdi Hosseini for supporting our the Quantum Game Club, along with Lauren Britten for providing the idea of implicit adjacency rule.

References

Ajinkya Borle, Vincent E. Elfving, and Samuel J. Lomonaco. 2021. Quantum approximate optimization for hard problems in linear algebra. *SciPost Phys. Core* 4 (2021), 031. https://doi.org/10.21468/SciPostPhysCore.4.4.031
Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. 2014. A Quantum Approximate Optimization Algorithm. arXiv:1411.4028 [quant-ph] https://arxiv.org/abs/1411.4028
Fred Glover, Gary Kochenberger, and Yu Du. 2019. A Tutorial on Formulating and Using QUBO Models. arXiv:1811.11538 [cs.DS] https://arxiv.org/abs/1811.11538
John Golden, Andreas Bärtschi, Daniel O'Malley, and Stephan Eidenbenz. 2023. Numerical Evidence for Exponential Speed-Up of QAOA over Unstructured Search for Approximate Constrained Optimization. 496–505. https://doi.org/10.1109/QCE57702.2023.00063
Lov K. Grover. 1996. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing* (Philadelphia, Pennsylvania, USA) *(STOC '96)*. Association for Computing Machinery, New York, NY, USA, 212–219. https://doi.org/10.1145/237814.237866
Maxim Gumin. 2016. *Wave Function Collapse Algorithm.* https://github.com/mxgmn/WaveFunctionCollapse
Raoul Heese. 2024. Quantum Wave Function Collapse for Procedural Content Generation. *IEEE Computer Graphics and Applications* (2024), 1–13. https://doi.org/10.1109/mcg.2024.3447775
Ali Javadi-Abhari, Matthew Treinish, Kevin Krsulich, Christopher J. Wood, Jake Lishman, Julien Gacon, Simon Martiel, Paul D. Nation, Lev S. Bishop, Andrew W. Cross, Blake R. Johnson, and Jay M. Gambetta. 2024. Quantum computing with Qiskit. https://doi.org/10.48550/arXiv.2405.08810 arXiv:2405.08810 [quant-ph]
Isaac Karth and Adam M. Smith. 2017. WaveFunctionCollapse is constraint solving in the wild. In *Proceedings of the 12th International Conference on the Foundations of Digital Games* (Hyannis, Massachusetts) *(FDG '17)*. Association for Computing Machinery, New York, NY, USA, Article 68, 10 pages. https://doi.org/10.1145/3102071.3110566
Hwanhee Kim, Seongtaek Lee, Hyundong Lee, Teasung Hahn, and Shinjin Kang. 2019. Automatic Generation of Game Content using a Graph-based Wave Function Collapse Algorithm. In *2019 IEEE Conference on Games (CoG)*. 1–4. https://doi.org/10.1109/CIG.2019.8848019
Samantha Koretsky, Pranav Gokhale, Jonathan M. Baker, Joshua Viszlai, Honghao Zheng, Niroj Gurung, Ryan Burg, Esa Aleksi Paaso, Amin Khodaei, Rozhin Eskandarpour, and Frederic T. Chong. 2021. Adapting Quantum Approximation Optimization Algorithm (QAOA) for Unit Commitment. In *2021 IEEE International Conference on Quantum Computing and Engineering (QCE)*. 181–187. https://doi.org/10.1109/QCE52317.2021.00035
Salonik Resch and Ulya R. Karpuzcu. 2021. Benchmarking Quantum Computers and the Impact of Quantum Noise. *ACM Comput. Surv.* 54, 7, Article 142 (jul 2021), 35 pages. https://doi.org/10.1145/3464420
Abdullah Ash Saki, Mahabubul Alam, and Swaroop Ghosh. 2019. Study of Decoherence in Quantum Computers: A Circuit-Design Perspective. arXiv:1904.04323 [cs.ET] https://arxiv.org/abs/1904.04323
Peter W. Shor. 1997. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM J. Comput.* 26, 5 (1997), 1484–1509. https://doi.org/10.1137/S0097539795293172