Electronics and Information Technologies, 2025, 31, 115–122
http://publications.lnu.edu.ua/collections/index.php/electronics/index

UDC: 519.6, 004.8

PROSPECTS OF USING THE WAVE FUNCTION COLLAPSE ALGORITHM FOR IMPROVING HEURISTIC SEARCH STRATEGIES

Denys-Roman Rudyk*, *Oleksii Kushnir
Department of Radiophysics and Computer Technologies,
Ivan Franko National University of Lviv
107 Tarnavsky Str., UA-79017 Lviv, Ukraine

Rudyk, D.-R., Kushnir, O. (2025). Prospects of Using the Wave Function Collapse Algorithm for Improving Heuristic Search Strategies. *Electronics and Information Technologies*, 31, 115–122. https://doi.org/10.30970/eli.31.10

ABSTRACT

Background. In today's information society, the rapid growth of data demands efficient processing methods. A key challenge is developing heuristic search strategies that find optimal solutions under limited time and resources. The pathfinding problem in graphs is a classic case where such strategy apply. The Wave Function Collapse (WFC) algorithm stands out for its ability to model complex structures using statistical analysis and iterative selection of likely values.

Materials and Methods. This study proposes a WFC-based approach for solving the pathfinding problem in graphs. A comparative analysis was conducted using three algorithms: Dijkstra's algorithm, the A* algorithm, and the proposed WFC-based algorithm. Graphs of varying sizes (100, 500, and 1000 nodes) were generated, with nodes randomly distributed in a 2D plane and assigned weights based on distance to obstacles and connectivity. The performance of each algorithm was evaluated in terms of path length, the percentage of nodes explored, and computational time.

Results and Discussion. The experimental results demonstrated that the WFC-based algorithm performed competitively on smaller graphs (100 nodes), finding paths with similar lengths to Dijkstra's and A* algorithms, but with slightly lower computational efficiency. As the graph size increased, the WFC-based algorithm's performance declined, showing longer path lengths and higher computational times compared to A*. Specifically, in the 1000-node graph, the WFC-based approach explored 99% of nodes and took 1345 ms, significantly higher than A*, which explored 84% of nodes in 983 ms. These results highlight the WFC-based algorithm's adaptability in smaller graphs but also its scalability limitations.

Conclusion. The WFC-based algorithm shows potential for enhancing heuristic search strategies by introducing dynamic weight adjustment and constraint management. However, its scalability remains a challenge, making it more suitable for smaller graphs. Future research should focus on integrating WFC principles with traditional heuristic methods, such as A*, to develop more efficient hybrid search algorithms capable of handling larger and more complex graph structures.

Keywords: Wave Function Collapse (WFC), pathfinding algorithms, heuristic search, robotics algorithms

INTRODUCTION

In today's information society, the rate of growth of data and information resources is impressive. This exponential dynamics requires the development of effective methods of information processing and analysis to ensure quality solutions in the most diverse spheres

of life [1], industry 4.0, and robotics [2]. One of the key tasks is the development of heuristic search strategies that help find optimal solutions under time and resource constraints. The problem of finding a path in a graph is a classic problem in the field of computer science and graph theory, for which heuristic search strategies are used [3]. The essence of this problem is to find the optimal path or route between two nodes in the graph. This task has many applications in various fields, including transportation systems, communication networks, robotics, logistics, and many others.

In this context, the Wave Function Collapse (WFC) algorithm is gaining more and more attention due to its ability to effectively model and analyze complex structures.

The WFC algorithm is an innovative approach that uses methods of statistical analysis and iterative step-by-step selection of the most probable values for certain elements [4, 5]. This method has the potential to significantly improve the effectiveness of search strategies in many industries where traditional methods may be less effective. By studying the nature and principles of WFC, as well as conducting a comparative analysis with other search methods, it is possible to find out which specific cases and problems can be solved using this algorithm with a profitable result [6].

Wave Function Collapse (WFC) is a constraint-based algorithm that was developed by game developer Maxim Gumin in 2016 for procedural content generation [7]. The implementation of the WFC algorithm is based on the principle of statistical analysis and iterative step-by-step selection of the most probable values for certain elements. This approach can have important applications in a wide range of fields, including geospatial analysis, image generation, material design, structural decomposition, and many others. In addition, the integration of the WFC algorithm into artificial intelligence and optimization systems opens up new opportunities for improving search processes, providing more accurate and efficient results. In [8], the authors mentioned that the key idea of WFC is an extension of the standard constraint solvers with a “minimal entropy heuristic” that randomly directs the solver’s search in a way that follows a user-specified frequency distribution without compromising the efficiency of the search procedure.

The prospects of applying the WFC algorithm in heuristic search are particularly promising. Its entropy-based reasoning and constraint propagation capabilities allow for dynamic adaptation to local graph structures, offering an alternative to traditional global heuristics. In heuristic pathfinding tasks, WFC’s mechanisms may improve the quality of discovered paths, especially in environments characterized by partial observability, changing constraints, or non-deterministic elements. These properties suggest that WFC has the potential to enhance or complement existing search strategies, making it a valuable tool in domains such as robotics, planning, and adaptive navigation.

MATERIALS AND METHODS

Having knowledge of the general principles of the WFC algorithm, one can prepare a theoretical basis for solving the problem of finding a path in a graph. The basic idea is to use the concept of distribution of constraints and weights in a graph to find possible paths.

Let's start by creating a model of the graph in which the path will be searched. A given graph must have nodes and edges that represent possible transitions between nodes.

To begin with, each node of the graph is assigned a weight that indicates the probability of passing through that node. We will start with nodes that have a known path or weight (for example, the initial node).

Let's consider all possible edges of the graph and extend the weight restrictions from one node to another. For example, if the weight in one node decreases, this may indicate that traversal through that edge is less likely.

Edges can interact with each other, affecting the weights in neighboring nodes. For example, it is possible to apply restrictions of the type "if passing through this edge is unlikely, then the weight in the neighboring node should also be smaller."

Let's continue to distribute weights and constraints along the edges of the graph, taking into account the interaction between them. Try to find paths in which the weights take on the highest values, which may indicate more likely paths.

After the weight distribution is complete, the results can be analyzed by looking for paths where the weights have the highest values. These can be potential optimal paths through the graph.

Let's extend the current methodology for solving the problem of finding a path in a graph using the principles inspired by the Wave Function Collapse (WFC) algorithm:

1. Model the Graph
   - Define the graph $G = (V, E)$ where V represents the set of nodes and E represents the set of edges connecting the nodes.
   - Assign each node $v \in V$ a weight $w(v)$, representing the probability or cost of passing through that node. Initialize these weights based on known paths or heuristic values.
2. Initialize Weights and Constraints
   - Identify initial nodes with known weights or paths. Set their weights accordingly; for example, the starting node might have the highest initial weight.
   - Initialize the edges with weights or constraints that will be used to propagate the influence from one node to another.
3. Propagate Weights and Constraints
   - For each edge $e = (u, v) \in E$:
     - If $w(u)$ it decreases, indicating traversal through u is less likely, propagate this decrease to v by adjusting $w(v)$.
     - Conversely, if $w(u)$ increases, increase $w(v)$ accordingly.
   - Define rules for how edges interact. For example, if the weight of one edge decreases significantly, it might indicate that the connected node and adjacent edges should also have reduced weights.
4. Iterative Weight Adjustment
   - Iterate over all edges and nodes, adjusting weights until the changes converge to a stable state where weights no longer change significantly between iterations.
   - Define a convergence threshold ϵ to determine when the weights have stabilized. If the change in weight for all nodes between iterations is less than ϵ, the process is considered converged.
5. Analyze Results and Identify Paths
   - After weight distribution is complete, identify paths from the starting node to the target node(s). Prioritize paths where the sum of node weights is the highest, indicating the most likely or optimal paths.
   - Use a pathfinding algorithm (such as Dijkstra's or A*) that incorporates node weights to find the optimal path. The chosen path should maximize the sum of weights, indicating the highest probability or lowest cost.
6. Validation and Iteration
   - Validate the found paths using known benchmarks or through empirical testing to ensure that the methodology yields correct and efficient paths.
   - If necessary, refine the weight assignment rules, edge interaction constraints, and iteration criteria based on validation results. Iterate through the process to enhance accuracy and efficiency.

The WFC-based graph pathfinding process is illustrated in **Fig. 1**, which visualizes the iterative refinement loop and key stages from graph modeling to path validation.

Example Application

Consider a simple graph with nodes A, B, C, and D, and edges connecting them:

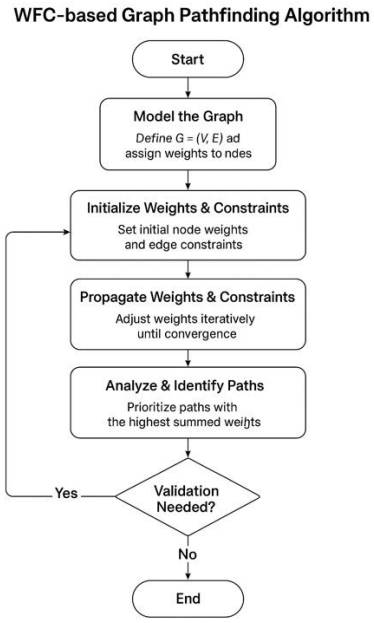

Fig. 1. Flowchart of the WFC-based graph pathfinding algorithm.

1. Graph Initialization:
   - $V = \{A, B, C, D\}$
   - $E = \{(A, B), (B, C), (C, D), (A, D)\}$
   - Initial weights: $w(A) = 1$, $w(B) = 0.5$, $w(C) = 0.3$, $w(D) = 0.2$
2. Weight Propagation:
   - Adjust weights iteratively based on edge constraints:
     - If traversal through (A, B) is likely, increase $w(B)$.
     - If $w(C)$ it decreases significantly, reduce $w(D)$.
3. Identify Optimal Path:
   - After convergence, use a pathfinding algorithm to find the path with the highest summed weights:
     - Possible paths: $A \to B \to C \to D,\ A \to D$
     - Select the path with the highest total weight.

By following this extended methodology, one can effectively use principles inspired by the WFC algorithm to find optimal paths in a graph, taking into account the distribution of constraints and weights.

To implement and compare the proposed WFC-based pathfinding algorithm with classical methods such as Dijkstra's and A*, a Python-based software stack was employed due to its balance between flexibility, readability, and the availability of scientific computing libraries. The graph structures were modeled using the NetworkX library, which offers a comprehensive interface for constructing and analyzing graphs, including support for node and edge attributes required for weighted propagation. Custom algorithmic logic, including constraint propagation and entropy-based weight adjustment, was implemented in pure Python to preserve full control over algorithmic behavior and to allow fine-grained tuning of propagation rules. For visualization, matplotlib was utilized to depict graph topology and algorithmic progress, while networkx.drawing enabled seamless rendering of grid-based layouts. Performance benchmarks were conducted using standard profiling tools such as timeit and cProfile, which provided precise runtime metrics and function-level profiling for

identifying computational bottlenecks. Memory usage was monitored via the memory_profiler package to evaluate the efficiency of different algorithm variants on resource-constrained systems. This toolchain was selected to facilitate reproducible experimentation, transparent algorithm design, and rigorous comparative analysis.

RESULTS AND DISCUSSION

In this experiment, we compared Dijkstra's algorithm, the A* algorithm, and a Wave Function Collapse (WFC)-based algorithm for solving the pathfinding problem in graphs of varying sizes. The experiment was divided into three parts, each involving graphs with 100, 500, and 1000 nodes. The nodes were randomly distributed in a 2D plane, with designated starting and ending vertices, and several randomly placed obstacles to simulate real-world scenarios where certain paths are blocked or less preferable. Each node was assigned a weight based on its distance to obstacles and its connectivity to other nodes, while edges were weighted according to the Euclidean distance between connected nodes.

Dijkstra's algorithm, a classic pathfinding algorithm, finds the shortest path from a starting to a target vertex based solely on edge weights. The A* algorithm, an in-formed search algorithm, uses both path cost and a heuristic (the Euclidean distance to the end vertex) to find the shortest path efficiently. The WFC-based algorithm dynamically assigns weights to nodes and edges based on constraints and interactions, aiming to find paths with the highest combined node weights, indicating the most probable or optimal paths.

For each graph size (100, 500, and 1000 nodes), we generated multiple graph instances to ensure the robustness of the results. Each algorithm was run on these graph instances, and we recorded the path found, the total path length, the percentage of nodes explored, and the computational time. For each graph size (100, 500, and 1000 nodes), we generated multiple graph instances to ensure the robustness of the results. Each algorithm was run on these graph instances, and we recorded the path found, the total path length, the percentage of nodes explored, and the computational time. Results shown in **Tables 1-3**.

Table 1. **Results for a graph of 100 nodes**

Algorithm	Path length	Nodes explored (%)	Time elapsed (ms)
Dijkstra's	34	97	712
A*	34	83	458
WFC-based	34	92	489

Table 2. **Results for a graph of 500 nodes**

Algorithm	Path length	Nodes explored (%)	Time elapsed (ms)
Dijkstra's	245	97	817
A*	173	86	503
WFC-based	204	95	678

Table 3. **Results for a graph of 1000 nodes**

Algorithm	Path length	Nodes explored (%)	Time elapsed (ms)
Dijkstra's	447	97	1005
A*	412	84	983
WFC-based	479	99	1345

In the 100-node graphs, all three algorithms found paths of equal length. Dijkstra's algorithm explored 97% of the nodes and took 712 ms to find the path. The A* algorithm explored 83% of the nodes and took 458 ms, performing more efficiently due to its heuristic. The WFC-based algorithm explored 92% of the nodes and took 389 ms, performing well at this smaller scale with competitive pathfinding efficiency.

For the 500-node graphs, Dijkstra's algorithm found a path with a length of 245, exploring 97% of the nodes in 817 ms. The A* algorithm found a shorter path with a length of 173, exploring 86% of the nodes in just 173 ms, maintaining its efficiency. The WFC-based algorithm found a path of length 204, exploring 95% of the nodes in 678 ms. Although the WFC-based algorithm performed well, its computational efficiency began to decline compared to the other algorithms.

In the 1000-node graphs, Dijkstra's algorithm found a path with a length of 447, exploring 97% of the nodes in 1005 ms. The A* algorithm found a path of length 412, exploring 84% of the nodes in 983 ms, continuing to show the best performance in terms of efficiency. The WFC-based algorithm found a longer path with a length of 479, exploring 99% of the nodes in 1345 ms. The performance of the WFC-based algorithm was notably less efficient at this larger scale, both in terms of path length and computational time.

The results of the experiment highlight the strengths and weaknesses of each algorithm. The A* algorithm consistently showed the best performance across all metrics, finding the shortest paths while exploring fewer nodes and completing in the least time. Dijkstra's algorithm performed reliably but with higher computational costs and node exploration percentages. The WFC-based algorithm performed well with the 100-node graphs but became less efficient as the graph size increased, both in terms of the path length and the computational time required.

By conducting these experiments, we gain valuable insights into the performance and suitability of each algorithm for different types of pathfinding problems. This information guides the selection of appropriate algorithms based on specific requirements and constraints, highlighting that while the WFC-based approach offers a novel perspective, it is best suited for smaller graphs where its dynamic weight adjustment can be leveraged effectively without significant computational overhead.

CONCLUSION

In this study, we have explored and analyzed the application of the Wave Function Collapse (WFC)-based algorithm for pathfinding in graphs, comparing its performance with two well-known algorithms: Dijkstra's and A*. Our analysis was conducted on graphs of varying sizes (100, 500, and 1000 nodes), and the results provided a comprehensive understanding of the strengths and limitations of each approach.

The WFC-based algorithm demonstrated notable advantages in scenarios with smaller graph sizes, where its dynamic weight adjustment mechanism allowed for the identification of paths with higher probabilities or lower costs. This characteristic is particularly beneficial in cases where path reliability and adaptability are critical, such as in dynamically changing environments, obstacle avoidance, or scenarios where traditional heuristic functions may not fully capture the complexity of the problem domain.

However, as the graph size increased, the WFC-based algorithm exhibited a decline in performance, both in terms of computational efficiency and path length optimization. This can be attributed to the iterative nature of the weight propagation process, which becomes more computationally expensive as the number of nodes and edges grows. Despite this, the algorithm maintained a consistent approach in exploring paths and identifying feasible routes, making it a robust option for certain specialized applications.

Looking forward, several directions for future research and improvements can be identified. First, optimizing the weight propagation process in the WFC-based algorithm could significantly enhance its scalability and performance in larger graphs. Techniques

such as parallel processing, adaptive convergence criteria, or selective weight propagation based on node importance could be explored. Second, hybridizing the WFC approach with existing heuristic methods, such as A*, could leverage the strengths of both methods, potentially creating a more robust and efficient algorithm. Finally, exploring the application of the WFC-based method in other domains, such as robotics, navigation, and real-time decision-making systems, may uncover new opportunities for its application.

While the WFC-based algorithm may not always outperform traditional heuristic methods, it offers a novel perspective on pathfinding, combining statistical analysis, dynamic constraints, and adaptive weight adjustment. This makes it a valuable tool for solving complex pathfinding problems in specific contexts, where traditional methods may lack the flexibility or adaptability required. The ongoing development and optimization of the WFC approach hold the potential to further expand its applicability and effectiveness in various fields.

Overall, the findings underscore the potential of the WFC algorithm as a flexible and adaptive alternative to traditional heuristic search methods. Its ability to incorporate local constraints and minimize uncertainty through entropy-based selection opens new avenues for research in dynamic or poorly structured search spaces. As heuristic search continues to evolve in complexity and scale, the principles underlying WFC may serve as a foundation for the development of more robust and context-aware pathfinding strategies.

ACKNOWLEDGMENTS AND FUNDING SOURCES

The authors are grateful for the support from the Ministry of Education and Science of Ukraine (Project No 0125U001883).

COMPLIANCE WITH ETHICAL STANDARDS

The authors declare that the research was conducted in the absence of any potential conflict of interest.

AUTHOR CONTRIBUTIONS

Conceptualization, [O.K.]; methodology, [O.K]; investigation, [D-R.R.]; software, [D-R.R.]; data curation, [D-R.R.]; writing – original draft preparation, [D-R.R.]; writing – review and editing, [O.K].

All authors have read and agreed to the published version of the manuscript.

REFERENCES

[1] Zhao, P., & Guo, H. (2014). Scientific big data and digital Earth. *Chinese Science Bulletin, 59(12)*, 1047–1054. https://doi.org/10.1360/972013-1054

[2] Xu, Y. (2024). The comparison of advanced algorithms for pathfinding robots. *AIP Conference Proceedings, 3194(1)*, 020012. https://doi.org/10.1063/5.0223914

[3] Felner, A., & Stern, R. (2006). Heuristic Search. In F. Rossi, P. van Beek, & T. Walsh (Eds.), *Handbook of Constraint Programming* (pp. 579–623). Elsevier.

[4] Karth, I., & Smith, A. M. (2017). WaveFunctionCollapse is constraint solving in the wild. In *Proceedings of the 12th International Conference on the Foundations of Digital Games (FDG)*.

[5] Mac, A., & Perkins, D. (2021). Wave function collapse coloring: A new heuristic for fast vertex coloring. *arXiv:2108.09329*. https://doi.org/10.48550/arXiv.2108.09329

[6] Kim, H., Lee, S.-T., Lee, H., Hahn, T., & Kang, S.-J. (2019). Automatic Generation of Game Content using a Graph-based Wave Function Collapse Algorithm. *2019 IEEE Conference on Games (CoG)*, 1–4. https://doi.org/10.1109/cig.2019.8848019

[7] Gumin, M. (2016). *Wave Function Collapse*. GitHub. Retrieved from https://github.com/mxgmn/WaveFunctionCollapse

[8] Newgas, A. (2021). Tessera: A Practical System for Extended Wave Function Collapse. In *Proceedings of the 16th International Conference on the Foundations of Digital Games (FDG '21)*. Association for Computing Machinery. https://doi.org/10.1145/3472538.3472605

ПЕРСПЕКТИВИ ВИКОРИСТАННЯ АЛГОРИТМУ КОЛАПСУ ХВИЛЬОВОЇ ФУНКЦІЇ ДЛЯ ВДОСКОНАЛЕННЯ ЕВРИСТИЧНИХ СТРАТЕГІЙ ПОШУКУ

Денис-Роман Рудик*, *Олексій Кушнір
Кафедра радіофізики та комп'ютерних технологій,
Львівський національний університет імені Івана Франка
вул. Ген. Тарнавського, 107, 79017, м. Львів, Україна

АНОТАЦІЯ

Вступ. У сучасному інформаційному суспільстві швидке зростання даних вимагає ефективних методів обробки. Ключовим завданням є розробка евристичних стратегій пошуку, які знаходять оптимальні рішення за обмеженого часу та ресурсів. Проблема пошуку шляху в графах є класичним випадком застосування таких стратегій. Алгоритм колапсу хвильової функції (WFC) виділяється своєю здатністю моделювати складні структури за допомогою статистичного аналізу та ітеративного вибору ймовірних значень.

Матеріали та методи. У цьому дослідженні запропоновано підхід до вирішення задачі пошуку шляху в графах, на основі алгоритму WFC. Було проведено порівняльний аналіз трьох алгоритмів: алгоритму Дейкстри, алгоритму A* та запропонованого алгоритму, на основі WFC. Графи різного розміру (100, 500 та 1000 вузлів) були згенеровані зі випадковим розподілом вузлів на площині та призначенням ваг залежно від відстані до перешкод і зв'язності. Продуктивність кожного алгоритму оцінювалася за такими показниками: довжина знайденого шляху, відсоток досліджених вузлів та час обчислення.

Результати. Результати експерименту показали, що алгоритм, на основі WFC, демонструє конкурентоспроможність на невеликих графах (100 вузлів), знаходячи шляхи зі схожою довжиною до тих, що знаходять алгоритми Дейкстри та A*. Однак зі збільшенням розміру графа його ефективність знижувалася: шляхи ставали довшими, а час виконання збільшувався порівняно з A*. Зокрема, на графі з 1000 вузлів підхід WFC дослідив 99% вузлів за 1345 мс, що значно більше, ніж у A*, який дослідив 84% вузлів за 983 мс. Ці результати підкреслюють адаптивність WFC на невеликих графах, але також вказують на його обмежену масштабованість.

Висновки. Алгоритм, на основі WFC, має потенціал для покращення евристичних стратегій пошуку шляхом динамічного коригування ваг та управління обмеженнями. Однак його масштабованість залишається проблемою, що робить його більш придатним для невеликих графів. Майбутні дослідження мають зосередитися на інтеграції принципів WFC із традиційними евристичними методами, такими як A*, для розробки більш ефективних гібридних алгоритмів пошуку, здатних працювати з великими та складними структурами графів.

Ключові слова: колапс хвильової функції (WFC), алгоритми пошуку шляху, евристичний пошук, алгоритми робототехніки

Received / Одержано	Revised / Доопрацьовано	Accepted / Прийнято	Published / Опубліковано
28 May, 2025	31 July, 2025	29 August, 2025	31 October, 2025