Procedural Content Generation in Video Games

CS4099 Major Software Project

Benjamin Mathias Sonnet
Supervised by David Morrison

University of
St Andrews

Computer Science
University of St Andrews
March 2024

Contents

1 Literature Review **2**
1.1 Constraint Programming . . . 2
1.1.1 Overview . . . 2
1.1.2 Combinatorial Problems . . . 2
1.1.3 Logic Programming Languages . . . 3
1.1.4 Further Application to Video Games . . . 3
1.2 Wave Function Collapse . . . 3
1.2.1 Implementation Variations . . . 4
1.2.2 Limitations of Wave Function Collapse . . . 4
1.3 PCG . . . 6
1.3.1 Overview . . . 6
1.3.2 Current Applications and Research . . . 6

Chapter 1

Literature Review

Three key themes were identified to be investigated, including constraint programming, wave function collapse and procedural content generation. While these themes are presented in three different sections, the ideas discussed heavily overlap.

Constraint programming and procedural content generation have wide applications and have been a big area of recent research. For example, procedural content generation has seen an increased use of machine learning in recent years, allowing new content to be generated from large dataset models. Similarly, novel constraint programming techniques such as wave function collapse have seen use in generating new output from very limited input data. These techniques are typically applied to video games in order to help create an ever changing experience for the player.

1.1 Constraint Programming

1.1.1 Overview

Constraint programming deals with modelling problems through constraints and then running a solver to find solutions. In the context of PCG and video games, the use of constraints can be useful to tailor the output of PCG as desired and generate new content based on old content. Constraint programming has also been used to solve game tasks. The use of constraints allows for well-defined outputs to be created, but pay for this with more unpredictable generation times when compared to other methods [1].

Constraint programming has been applied to a large variety of problem categories, ranging from combinatorial mathematics to logistics and scheduling [2].

1.1.2 Combinatorial Problems

For example, three papers take a deeper look into combinatorial problems, which deal with finding an optimal solution among a finite set of possibilities. The first paper [3] recognises that deep reinforcement learning has been used to tackle these problems, but that this only provides approximate solutions. To find optimal solutions, it instead applies constraint programming, detailing its use for problems such as the travelling salesman problem with time windows and the 0-1 knapsack problem. Another paper [3] identifies the problem that large scale combinatorial problems may

huge search spaces, resulting in low solver performance. To address this, it introduces and automated process to add streamliner constraints, which focus effort on searching promising parts of the search space to improve performance. The last of the three papers [4] provides a survey of combinatorial problems and attempts at modelling and solving them effectively, identifying for example that algorithm selection techniques can achieve significant performance improvements for combinatorial search.

Another paper [5] recognises the difficulty in testing constraint solvers due to the vastness of searches they may perform. As a solution, it uses metamorphic testing, which generates new test cases from existing ones. However, it expresses the limitation that this should be used with other forms of testing as metamorphic testing does not recognise when a solver falsely identifies a problem as unsolvable.

1.1.3 Logic Programming Languages

Another interesting application of constraint programming is to AI deep reinforced learning. One paper [6] explores the use of the Prolog logic programming language to generate data sets to aid this learning, finding positive results in trained AI agent performance.

Other languages similar to Prolog such as Answer Set Programming (ASP) and the Video Game Description Language (VGDL) extend Prolog's concepts with application to video games. For example, ASP can be used to generate constrained dungeons [7]. One paper [8] also applies constraint programming to dungeon generation, but instead uses a graph constraint to generate high quality dungeons with distinct areas. Another paper [9] builds on the General Video Game AI framework (GVG-AI) and the Video Game Description Language (VGDL) to create high quality level generators.

1.1.4 Further Application to Video Games

In the context of video games, constraint programming research has looked into applying constraints to PCG and solving game tasks. One recent paper [10] aims to solve a planning problem presented in the game Plotting. In it, the player must plan a sequence of actions to clear blocks of varying types from a grid. The paper models the problem in two modelling language, namely the widely used Planning Domain Definition Language (PDDL) and the novel Essence Prime. Their effectiveness is then compared using several solvers, finding a SAT solver to be most effective for the problem.

1.2 Wave Function Collapse

Wave Function Collapse (WFC) is a modern PCG method that has found use in games such as Townscaper [11] and Bad North [12]. It can be described as a family of algorithms, rather than one specific algorithm [13]. As such, a large variety of implementations are available online, each with their own specialisations to solve the problem they were designed for. One paper noted that Wave Function Collapse is often used as a black box, being incorporated into a workflow without being altered [1].

Wave Function Collapse builds off of the concepts of model synthesis [14], which is a method for procedurally modelling complex 3D shapes. In it, the user defines an input model detailing various dimensional, geometric and algebraic constraints [15]. This is then used to create output satisfying the modelled constraints.

1.2.1 Implementation Variations

Data Input

The data input stage is likely the WFC stage with the most variation across implementations. The original implementation [14] supports both an overlapping and simple tiled model. The overlapping version takes a sample image and defines overlapping patterns of $N \times N$ tiles, where the output is constructed of a random arrangement of these patterns. The simple tiled model instead defines single tile patterns, where each tile has its own defined set of possible neighbours.

2D vs 3D

While a lot of implementations focus solely on 2D input tiles and grids, much fewer implementations support 3D input or output. This makes it a challenge to apply WFC to 3D environments. Furthermore, the increased complexity from a 3D environment can lead to an increased failure rate, which requires techniques such as backtracking or modifying in blocks to counteract.

1.2.2 Limitations of Wave Function Collapse

Research papers on WFC frequently attempt to identify and find solutions for problems that implementations of the algorithm commonly face. Some of the most common problems are a lack of global constraints, overfitting and performance. These problems, as well as some other challenges, are discussed below.

Lack of Global Constraints

One problem with WFC is that, while output can be tailored to satisfy local constraints, global constraints are not inherently supported. This results in there being no inherent overall structure to the output.

In some applications, such as the game Caves of Qud [16], WFC is used only after other algorithms have defined distinct regions of the map [13], [17]. This multi-pass approach allows WFC to be used to its strengths.

Two paper add in additional constraints that are checked after each observation step. In the first paper [18], constraints added include a minimum tile count, maximum tile count and object distance constraint. These allow balancing of tile counts and more control on how objects should be placed in the level. Furthermore, the option to preset tiles at the start of generation and carry out generation in two passes is also included. This helps create levels of certain styles and to reduce conflicts arising from trying to generate everything in one pass, such as placing an object on an unsuitable tile. The second paper, [17], achieves the same goal as minimum and maximum tile count constraints through the use of a weighted choice. In addition to using entropy to choose the most constrained tile after each propagation step,

assigning a weight to each choice can encourage the algorithm to choose a different balance of tiles. Furthermore, this paper introduces a second observation step. This performs a second, smaller scale WFC algorithm, which can be used to refine item placement and create subregions within a map.

Solutions altering WFC directly to support global constraints are faced with the issue that the additional constraints can have a negative impact on performance. However, by combining such solutions with those discussed in the Performance subsection below, the impact can be reduced [13].

Overfitting

When adding a lot of detail to the input, such as through using complex tilesets, the output may become too constrained. This can result in overfitting and an increased failure rate.

Here, a multi-pass approach can not only be used to help globally constrain the output, but also to reduce risk of overfitting. This is done by reducing the detail of the input and instead adding additional details to the output in a second pass once wave function collapse has run.

Performance

Another issue identified with WFC is performance. While the chance of success is high with small inputs, larger inputs are much likelier to fail, especially with more complicated tilesets [13]. This can result in a significant generation time for large outputs. (More citations and graph of performance) Several solutions to WFC's scalability problem have been proposed.

One of the most common solutions is to include some form of backtracking, which allows further searching of the search space upon a contradiction instead of having to restart. However, with complex tilesets, care must be taken to reduce the chance of backtracking exploring an unpromising search space for an extended time. Using a search heuristic could help with this.

Nested WFC [19] is one technique that aims to improve scalability of WFC. It splits a larger grid into smaller grids, evaluating sub-grids diagonally from the top left. Each sub-grid overlaps constraints from its left and upper neighbour to satisfy constraints between adjacent sub-grids. This can be extended to an infinite space by overlapping new cells with old cells. While this technique does improve the performance of WFC, evaluation found that that a large number of conflicts from edge data still occur.

Another technique, infinite modifying in blocks [20], applies WFC multiple times in small chunks. Keeping the chunks small keeps the performance of generation high, while running WFC in four layers per chunk ensures that constraints are satisfied between adjacent chunks. The layering also addresses the limitation of conflicts that Nested WFC struggles with. As each chunk is made up of four WFC layers, failed layers can usually be ignored rather than having to be regenerated. However, this comes with computational overhead from running WFC four times per chunk.

Other Challenges

Environments of a certain style, especially those trying to create a realistic feel, may struggle from WFC's use of a grid structure for its output. However, if this regular grid is transformed into an irregular quadrilateral grid, more complex shapes can be used. One paper achieves this by using a graph-based data structure, which can be integrated with a navigation mesh in 3D [21]. However, this solution is limited due to a lack of control over solution order. Another paper explores the use of a growing grid neural network to augment WFC [22]. This paper also found that players have higher self-confidence in navigating the map and an increased ability to form mental maps of their environment.

Very simple implementations may ignore symmetry when defining tile neighbours in the simple tiled method [23]. Instead, every neighbour for each direction of each tile is listed explicitly. While this keeps the code simple, it means that a huge amount of work is required when defining neighbours for complex tilesets, with high chance of human error. What the original WFC implementation and several others do instead is to define a symmetry type for each tile. This means that much less data about the input has to be provided, lessening the work required and chance of human error. The original WFC implementation defines five symmetry types, which it applies to a variety of 2D images. Another paper instead defines nine symmetry types, which supports a greater variety of tilesets [18].

1.3 Procedural Content Generation

1.3.1 Overview

Procedural Content Generation (PCG) describes the use of algorithms to pseudo-randomly generate content. In the context of video games, this randomisation is often used to provide players with variety that can make games more enjoyable to play multiple times. Procedural Content Generation can be used in many facets of video game development. One frequent use of PCG is in randomised level generation. For example, in exploration games such as Minecraft [24] and No Man's Sky [25], PCG is used to generate the player's world, offering virtually infinite locations to explore. Other uses include randomising enemy characteristics in Shadow of Mordor [26] and generating randomised weapons in Borderlands [27]. The book 'Procedural Content Generation in Games' [7] is a great resource to read more about PCG.

1.3.2 Current Applications and Research

A lot of modern PCG research looks into using AI and machine learning to generate content. Some current applications of machine learning are art, music and code generation as well as chatbots [28], text-to-3D synthesis [29] and even self-driving cars [30].

Applications like Stable Diffusion [31] can be used to create high quality text-to-image content, while others such as Magic3D [29] offers text-to-3D synthesis. In terms of text-to-text, ChatGPT [32] serves as a leading language model that can be used to converse about any topic, while GitHub Copilot [33] can generate code snippets from user prompts.

In the context of games, machine learning is frequently used in areas like text, character model, texture, music and sound generation [34]. Languages such as the Video Game Description Language (VGDL) have even be used to generate entire games using AI [35], [36]. However, by themselves, such languages have limited expressiveness, making it difficult to create interesting games [35].

One paper [37] identifies that Generative Adversarial Networks (GANs) can also be used for image generation, but that it is difficult to incorporate constraints. As a solution, it proposes a Conditional Embedding Self-Attention Generative Adversarial Network (CESAGAN). This allows the embedding of a feature vector to the input, enabling the network to model non-local constraints. As a result, this produces higher quality outputs with fewer duplicates. In WFC, it is also difficult to enforce non-local constraints without additional modifications.

Another transforms 2D level design problems into Markov decision processes [38]. This aids reinforced learning to produce high quality output levels. It suggests this reinforced learning could be applied to self-play agents to improve the content generated through simulated playtesting. Three other papers similarly suggest that machine learning is useful for evaluating content through methods like simulated playtesting [34], [36], [39]. In the context of WFC, it might be useful to apply machine learning content evaluation and adjust the output to increase its quality.

One paper [39] also comments specifically on two limitations of PCG via machine learning. It states that the playability of output produced through machine learning is not guaranteed to be playable, but rather biassed towards generating playable content through the input. The second limitation is that most machine learning has been applied to 2D content. Similarly, whether WFC output is playable or not is not always guaranteed but heavily depends on how the input is defined. Furthermore, the core WFC implementation and many of its offshoots only support 2D content generation [14]. As a result, both machine learning PCG and WFC require carefully tweaked input data and an output evaluator when applied to level generation.

Bibliography

[1] I. Karth and A. M. Smith, 'Wavefunctioncollapse is constraint solving in the wild,' in *Proceedings of the 12th International Conference on the Foundations of Digital Games*, ser. FDG '17, Hyannis, Massachusetts: Association for Computing Machinery, 2017. DOI: 10.1145/3102071.3110566. [Online]. Available: https://doi.org/10.1145/3102071.3110566.

[2] C. Jefferson *et al.*, *Csplib problems library*. [Online]. Available: https://www.csplib.org/Problems/categories.html (visited on 31/10/2023).

[3] Q. Cappart, T. Moisan, L.-M. Rousseau, I. Prémont-Schwarz and A. A. Cire, 'Combining reinforcement learning and constraint programming for combinatorial optimization,' *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 5, pp. 3677–3687, May 2021. DOI: 10.1609/aaai.v35i5.16484. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/16484.

[4] *Data Mining and Constraint Programming: Foundations of a Cross-Disciplinary Approach* (Lecture Notes in Computer Science), en. Cham: Springer International Publishing, 2016, vol. 10101. DOI: 10.1007/978-3-319-50137-6. [Online]. Available: http://link.springer.com/10.1007/978-3-319-50137-6.

[5] Ö. Akgün, I. P. Gent, C. Jefferson, I. Miguel and P. Nightingale, 'Metamorphic testing of constraint solvers,' in *Principles and Practice of Constraint Programming*, J. Hooker, Ed., Cham: Springer International Publishing, 2018, pp. 727–736.

[6] G. De Gasperis, S. Costantini, A. Rafanelli, P. Migliarini, I. Letteri and A. Dyoub, 'Extension of constraint-procedural logic-generated environments for deep Q-learning agent training and benchmarking,' *Journal of Logic and Computation*, exad032, Jun. 2023. DOI: 10.1093/logcom/exad032. eprint: https://academic.oup.com/logcom/advance-article-pdf/doi/10.1093/logcom/exad032/50530031/exad032.pdf. [Online]. Available: https://doi.org/10.1093/logcom/exad032.

[7] N. Shaker, J. Togelius and M. J. Nelson, *Procedural Content Generation in Games: A Textbook and an Overview of Current Research*. Springer, 2016. DOI: 10.1007/978-3-319-42716-4. [Online]. Available: https://www.pcgbook.com/ (visited on 12/09/2023).

[8] G. Glorian, A. Debesson, S. Yvon-Paliot and L. Simon, 'The Dungeon Variations Problem Using Constraint Programming,' in *27th International Conference on Principles and Practice of Constraint Programming (CP 2021)*, L. D. Michel, Ed., ser. Leibniz International Proceedings in Informatics (LIPIcs), vol. 210, Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021, 27:1–27:16. DOI: 10.4230/LIPIcs.CP.2021.27. [Online]. Available: https://drops.dagstuhl.de/opus/volltexte/2021/15318.

[9] A. Khalifa, D. Perez-Liebana, S. M. Lucas and J. Togelius, 'General video game level generation,' in *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, ser. GECCO '16, Denver, Colorado, USA: Association for Computing Machinery, 2016, pp. 253–259. DOI: 10.1145/2908812.2908920. [Online]. Available: https://doi.org/10.1145/2908812.2908920.

[10] J. Espasa, I. Miguel and M. Villaret, 'Plotting: A Planning Problem with Complex Transitions,' in *28th International Conference on Principles and Practice of Constraint Programming (CP 2022)*, C. Solnon, Ed., ser. Leibniz International Proceedings in Informatics (LIPIcs), vol. 235, Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022, 22:1–22:17. DOI: 10.4230/LIPIcs.CP.2022.22. [Online]. Available: https://drops.dagstuhl.de/opus/volltexte/2022/16651.

[11] O. Stålberg, *Townscaper*, 2020. [Online]. Available: https://www.townscapergame.com/ (visited on 13/09/2023).

[12] O. Stålberg, *Bad north*, 2018. [Online]. Available: https://www.badnorth.com/ (visited on 13/09/2023).

[13] I. Karth and A. M. Smith, 'Wavefunctioncollapse: Content generation via constraint solving and machine learning,' *IEEE Transactions on Games*, vol. 14, no. 3, pp. 364–376, 2022. DOI: 10.1109/TG.2021.3076368.

[14] M. Gumin, *Wave Function Collapse Algorithm*, version 1.0, Sep. 2016. [Online]. Available: https://github.com/mxgmn/WaveFunctionCollapse.

[15] P. Merrell and D. Manocha, 'Model synthesis: A general procedural modeling algorithm,' *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 6, pp. 715–728, 2011. DOI: 10.1109/TVCG.2010.112.

[16] Freehold Games, *Caves of qud*, 2023. [Online]. Available: https://www.cavesofqud.com/ (visited on 30/10/2023).

[17] A. Sandhu, Z. Chen and J. McCoy, 'Enhancing wave function collapse with design-level constraints,' in *Proceedings of the 14th International Conference on the Foundations of Digital Games*, ser. FDG '19, San Luis Obispo, California, USA: Association for Computing Machinery, 2019. DOI: 10.1145/3337722.3337752. [Online]. Available: https://doi.org/10.1145/3337722.3337752.

[18] D. Cheng, H. Han and G. Fei, 'Automatic generation of game levels based on controllable wave function collapse algorithm,' in Jan. 2020, pp. 37–50. DOI: 10.1007/978-3-030-65736-9_3.

[19] Y. Nie, S. Zheng, Z. Zhuang and X. Song, *Extend wave function collapse to large-scale content generation*, 2023. arXiv: 2308.07307 [cs.AI].

[20] Boris, *Infinite modifying in blocks*, Jun. 2022. [Online]. Available: https://www.boristhebrave.com/2021/11/08/infinite-modifying-in-blocks/.

[21] H. Kim, S. Lee, H. Lee, T. Hahn and S. Kang, 'Automatic generation of game content using a graph-based wave function collapse algorithm,' in *2019 IEEE Conference on Games (CoG)*, 2019, pp. 1–4. DOI: 10.1109/CIG.2019.8848019.

[22] T. Møller and J. Billeskov, 'Expanding wave function collapse with growing grids for procedural content generation.,' Ph.D. dissertation, May 2019. DOI: 10.13140/RG.2.2.23494.01607.

[23] G. Kane, *Easy wave function collapse*. [Online]. Available: https://github.com/GarnetKane99/WaveFunctionCollapse (visited on 30/10/2023).

[24] J. Fingas, *Here's how "minecraft" creates its gigantic worlds*, Jul. 2019. [Online]. Available: https://www.engadget.com/2015-03-04-how-minecraft-worlds-are-made.html (visited on 12/09/2023).

[25] H. Alexandra, *A look at how no man's sky's procedural generation works*, Oct. 2016. [Online]. Available: https://kotaku.com/a-look-at-how-no-mans-skys-procedural-generation-works-1787928446 (visited on 12/09/2023).

[26] 2. Game DeveloperStaffJanuary 01, *7 uses of procedural generation that all developers should study*, Jan. 2016. [Online]. Available: https://www.gamedeveloper.com/design/7-uses-of-procedural-generation-that-all-developers-should-study (visited on 12/09/2023).

[27] W. Yin-Poole, *How many weapons are in borderlands 2?* Jul. 2012. [Online]. Available: https://www.eurogamer.net/how-many-weapons-are-in-borderlands-2 (visited on 12/09/2023).

[28] Y. Cao *et al.*, *A comprehensive survey of ai-generated content (aigc): A history of generative ai from gan to chatgpt*, 2023. arXiv: 2303.04226 [cs.AI].

[29] C.-H. Lin *et al.*, *Magic3d: High-resolution text-to-3d content creation*, 2023. arXiv: 2211.10440 [cs.CV].

[30] A. Gambi, M. Mueller and G. Fraser, 'Automatically testing self-driving cars with search-based procedural content generation,' in *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*, ser. ISSTA 2019, Beijing, China: Association for Computing Machinery, 2019, pp. 318–328. DOI: 10.1145/3293882.3330566. [Online]. Available: https://doi.org/10.1145/3293882.3330566.

[31] Stability AI, *Stable diffusion generative models*. [Online]. Available: https://github.com/Stability-AI/generative-models (visited on 26/10/2023).

[32] OpenAI, *Chatgpt*. [Online]. Available: https://chat.openai.com/ (visited on 26/10/2023).

[33] GitHub, *Github copilot*. [Online]. Available: https://github.com/features/copilot (visited on 26/10/2023).

[34] J. Liu, S. Snodgrass, A. Khalifa, S. Risi, G. N. Yannakakis and J. Togelius, 'Deep learning for procedural content generation,' *Neural Computing and Applications*, vol. 33, no. 1, pp. 19–37, Jan. 2021. DOI: 10.1007/s00521-020-05383-8. [Online]. Available: https://doi.org/10.1007/s00521-020-05383-8.

[35] D. Perez-Liebana, J. Liu, A. Khalifa, R. D. Gaina, J. Togelius and S. M. Lucas, 'General video game ai: A multitrack framework for evaluating agents, games, and content generation algorithms,' *IEEE Transactions on Games*, vol. 11, no. 3, pp. 195–214, 2019. DOI: 10.1109/TG.2019.2901021.

[36] X. Neufeld, S. Mostaghim and D. Perez-Liebana, 'Procedural level generation with answer set programming for general video game playing,' in *2015 7th Computer Science and Electronic Engineering Conference (CEEC)*, 2015, pp. 207–212. DOI: 10.1109/CEEC.2015.7332726.

[37] R. Rodriguez Torrado, A. Khalifa, M. Cerny Green, N. Justesen, S. Risi and J. Togelius, 'Bootstrapping conditional gans for video game level generation,' in *2020 IEEE Conference on Games (CoG)*, 2020, pp. 41–48. DOI: 10.1109/CoG47356.2020.9231576.

[38] A. Khalifa, P. Bontrager, S. Earle and J. Togelius, 'Pcgrl: Procedural content generation via reinforcement learning,' *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 16, no. 1, pp. 95–101, Oct. 2020. DOI: 10.1609/aiide.v16i1.7416. [Online]. Available: https://ojs.aaai.org/index.php/AIIDE/article/view/7416.

[39] A. Summerville *et al.*, 'Procedural content generation via machine learning (pcgml),' *IEEE Transactions on Games*, vol. 10, no. 3, pp. 257–270, 2018. DOI: 10.1109/TG.2018.2846639.