buildings

Article

Underground Parking Layout Generation Based on the WaveFunctionCollapse Algorithm

Di Lan, Kezhen Chen and Zhen Xu *

School of Architecture, Tianjin University, Tianjin 300072, China; landi730@tju.edu.cn (D.L.)
* Correspondence: zhenxu@tju.edu.cn

Abstract: During the design process, architectural layout configuration is subject to complex constraints such as site conditions and design requirements, resulting in limited design efficiency. This research aims to provide architects with an effective design tool that can generate reference-worthy underground parking layout solutions based on the given site information. In this research, we extract spatial modules from underground parking layouts, and transform the design constraints into adjacency rules based on the analysis of the configuration process for underground parking layout, then develop a generation and optimization model of the underground parking layout based on the WaveFunctionCollapse algorithm (WFC) and Multi-objective Optimization (MOO), and verify the effectiveness of the model through experiments. The results show that with given plan contour and entrance/exit locations as inputs, the model can efficiently generate architectural layout solutions that meet the design objectives.

Keywords: underground parking layout configuration; generative design; WaveFunctionCollapse algorithm (WFC); multi-objective optimization (MOO)

check for updates

Citation: Lan, D.; Chen, K.; Xu, Z. Underground Parking Layout Generation Based on the WaveFunctionCollapse Algorithm. *Buildings* **2023**, *13*, 2898. https://doi.org/10.3390/buildings13112898

Academic Editor: Svetlana J. Olbina

Received: 17 October 2023
Revised: 9 November 2023
Accepted: 13 November 2023
Published: 20 November 2023

1. Introduction

Architectural layout configuration is a critical aspect of architectural design that is subject to various constraints, such as site conditions, design requirements, technical specifications, building codes, and aesthetic considerations. Architects often invest significant time and effort in seeking the optimal solution to meet these constraints [1]. In this context, generative design, which can be defined as a design approach that uses algorithms to generate designs [2], offers architects a more efficient means of exploring a wider design space through an iterative design process [3]. Based on mechanisms of generation, generative design can be categorized into two approaches: data-based generative design and rule-based generative design [4]. The former involves acquiring knowledge from case data, with artificial neural networks based on deep learning gaining significant attention in recent years [5–7]. On the other hand, rule-based generation methods generate results based on predefined rules. Existing methods in this category encompass L-systems, cellular automata, genetic algorithms, swarm intelligence, reinforcement learning and shape grammars [8–13]. However, both approaches have their limitations in practice. Data-based generative design relies on large volumes of case data and lacks adaptability to changing design conditions. On the other hand, rule-based generative design struggles with ensuring the overall quality of the generated results and effectively filtering valuable results from a large pool of generated outcomes, particularly when faced with complex design problems. Consequently, there is a demand for the exploration of efficient and adaptable generative design algorithms and workflows for architectural layout configurations that are capable of adapting to evolving design conditions and delivering optimized results that align with the design objectives.

WaveFunctionCollapse (hereinafter referred to as WFC), initially developed by Maxim Gumin in 2016 [14], is an algorithm for procedural content generation (PCG) that has gained prominence in recent years [15–17]. WFC draws inspiration from the concept of wave function collapse in quantum mechanics. Its fundamental mechanism is as follows: a system is set up with a specific number of slots, each capable of accommodating a range of predefined modules. In the absence of observation, the state of all the slots is indeterminate, resulting in maximum entropy within the system. Once the state of a particular slot is determined, the states of the neighboring slots are also determined based on a predefined set of neighboring rules. This reduces the entropy of the system, resulting in a collapse. Once the state of all the slots is determined, the collapse is complete, and every part of the system adheres to the predefined rules [14,18] (Figure 1).

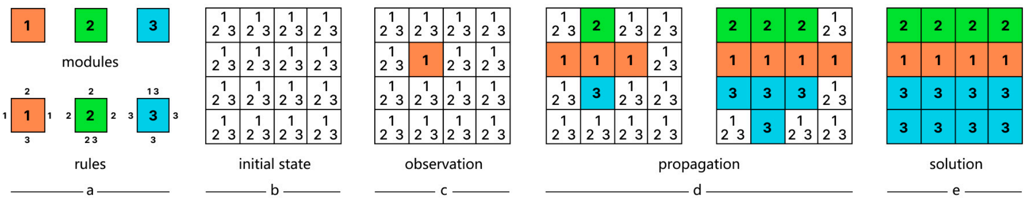


Figure 1. Basic mechanism of WFC. (**a**) Modules and their neighboring rules are predefined (colors represent types of modules). (**b**) In the initial state, each slot allows placement of any predefined module. (**c**) During the observation phase, a slot is randomly chosen and one of the predefined modules is placed in it at random. (**d**) During the propagation phase, the states of other slots are determined by the predefined neighboring rules. (**e**) Once the state of all slots is determined, a solution is attained.

As a PCG algorithm, WFC combines machine learning and constraint solving on the algorithmic level. This algorithm demonstrates an exceptional capability to generate rich and high-quality content without requiring extensive training data [18,19]. Consequently, this approach has the potential to overcome the limitations commonly associated with data- and rule-based generative design methods, thereby enabling its potential use in the field of architectural generative design. WFC possesses two notable features. Firstly, unlike many generative methods that rely heavily on parametric control, WFC synthesizes outcomes based on constituent elements (modules) and the relationships between them (rules). This guarantees a high level of control over the generated outcomes as all modules and rules are pre-established. Secondly, unlike some generation methods that require extensive training data, WFC achieves rich content generation using only a small number of module inputs and rule settings. This guarantees the efficiency of the generation process [14,20] (Figure 2).

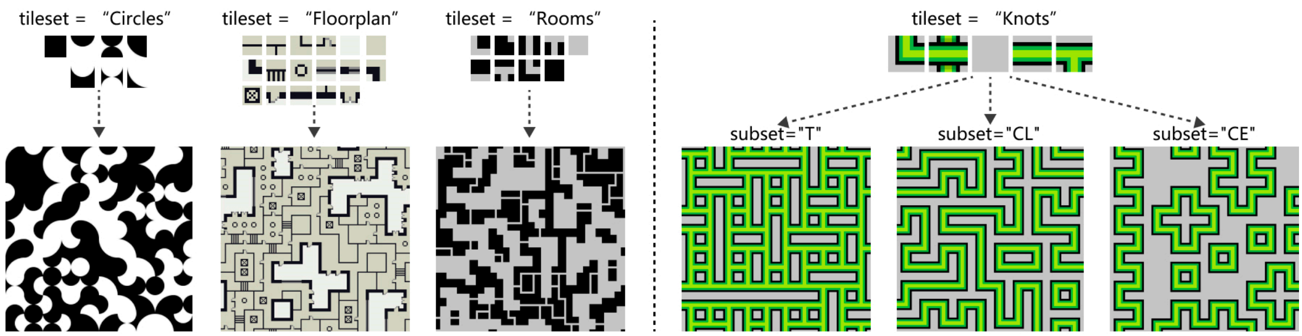


Figure 2. (**Left**): In WFC, the form of the input directly affects the generated results; (**Right**): by changing the rules, a minimal input can generate rich content.

Multi-objective optimization (MOO) is a discipline within the field of multi-criteria decision-making that seeks to identify the optimal trade-off solution for problems with multiple objectives. The main goal of optimization is to identify the optimal solution (or a set of optimal solutions, known as Pareto solutions) for a specified design problem [21]. Given the inherent complexity of architectural design problems, multi-objective optimization is recognized as a suitable approach for optimizing results in architectural generative design [22–26]. Its applications span from building performance optimization to neighborhood layouts optimization on an urban scale, demonstrating its ability to address diverse objectives within an architectural generative design process.

As an important branch of architectural layout configuration, underground parking layout configuration is representative because of its clear design objectives and quantifiable design constraints. This study aims to investigate the potential of WFC in the realm of architectural generative design by applying it to the generative design of underground parking layouts, and to explore a new workflow that effectively combines WFC and MOO. A WFC-based generation and MOO-based optimization model is established, which is capable of generating the optimal layout solution in real time according to the given plan contour and entrance/exit locations of the underground parking in order to aid architects to cope with the continuous adjustments during the design process. In this paper, we use the layout generation of a single-level underground parking with an 8.4 m $\times$ 8.4 m column network as an example to demonstrate generation and optimization methods.

2. Related Work

Extensive research has been conducted to implement generative methods in architectural layout configuration. Verma and Thakur [10] utilized a genetic algorithm to generate consistent conceptual architectural layout solutions in accordance with design specifications. Veloso and Krishnamurti [12] proposed a method that employs multi-agent deep reinforcement learning to create spatial agents that interact within site to fulfill specific objectives associated with a house layout configuration. Wang et al. [13] utilized shape grammar to generate layouts for traditional village dwellings. Filtration rules have been implemented as an optimization approach by reducing low-quality results, thereby ensuring the efficiency of the generative design process. In relation to underground parking layout configuration, Carrasco and Pecanac [27] identified the advantages of computer-aided design (CAD) in the semi-automated development of parking layouts. The proposed approach involves the automatic generation of parking layout options, which are based on predefined construction lines and physical constraints of the parking lot. Yu [28] proposed a generative design framework based on reinforcement learning to facilitate the realization of underground parking layout generative design in a regular column network. Despite the increasing popularity of generative design approaches regarding architectural layout configuration, the field has yet to be thoroughly explored. There remains a dearth of workflows that can efficiently navigate the design space of a given problem and simultaneously filter the optimized solution from the generated results.

As an emerging generative algorithm, WFC has found widespread application in various fields due to its exceptional capabilities. For instance, in the game Townscaper (2022), the designer utilized WFC to construct a customizable virtual town world that can infinitely expand under rule-based control [29,30]. Moreover, WFC has also been employed in diverse domains such as poetry creation [31] and 3D city model generation [32], showcasing its adaptability to different fields. The ability of WFC to construct 2D and even 3D spatial systems using predefined modules and rules suggests its potential application in the realm of architecture, particularly in the field of generative design [33]. Van et al. [34] utilized PCG to effectively create complex architectural spaces. This was achieved by breaking down building spaces into minimal units according to their functions and establishing connection rules that were guided by architectural semantics and typology. This approach exhibits similarities with the mechanism utilized in WFC, highlighting the potential of this kind of approach within the realm of architectural generative design. However, application cases of WFC in this field are still scarce; the specific methods of utilizing WFC in this context require further exploration.

Many researchers have investigated the integration of generative design methods and multi-objective optimization. Mukkavaara et al. [22] emphasized the significance of optimization approaches in the design process, and proposed a framework for the exploration of architectural design solutions. This framework encompasses both the generative design process and multi-objective optimization process. Huang et al. [23] identified a lack of maturity in the techniques and workflows associated with optimization-based generative design. In response, they proposed a framework that integrates generative design methods with data-driven decision-making for urban design. Gerber and Lin [24] integrated parametric modeling with multi-objective optimization to provide a comprehensive platform for conducting trade-off analysis in the areas of design, energy use intensity, and finance. The results of their study underscore the advantages of utilizing multi-objective optimization, as it enhances design decision-making during the initial phases of the design process and accommodates a greater degree of design complexity. Nagy et al. [25] outlined a flexible generative design workflow for office space planning. Their approach involved employing a multi-objective genetic algorithm to generate design alternatives based on six architectural performance criteria. While the results of their study were promising, the researchers acknowledged that the calculation of each design is still relatively slow and there is a need for a more efficient method of generation and optimization. In another research, Nagy et al. [26] demonstrated the utilization of an optimization-based generative design process for developing residential neighborhood layouts on an urban scale, in which profitability of the project for the developer and the potential for energy generation of solar panels acted as objectives. They concluded that this project exemplifies the effectiveness of the generative design process in generating sound design strategies, while also unveiling valuable insights regarding the inherent conflicts and tradeoffs between design objectives. Despite the vast amount of existing research, there remains a necessity to investigate a more cohesive workflow that amalgamates generation and optimization processes in order to enhance the efficiency of the generative design process.

3. Framework of Generation and Optimization Process

The general process of generation and optimization consists of the following steps (Figure 3):

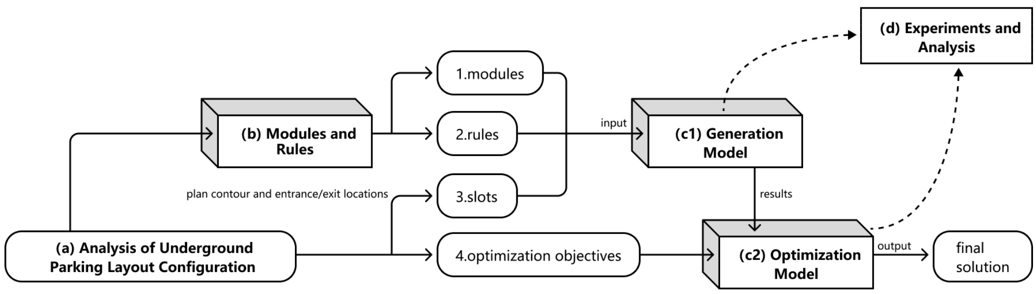


Figure 3. Basic framework of generation and optimization process.

(a) Analysis of the configuration process for underground parking layout: in this step, we summarize the design process and key characteristics of underground parking layout configuration by consulting relevant design codes, and transform the layout configuration problem into a constraint satisfaction problem based on the analysis.

(b) Modules and rules: based on the analysis of the configuration process for underground parking layout described in the previous step, the parking space is divided into basic spatial units to develop modules for WFC, and the neighboring rules between the modules are extracted by analyzing the functional relationship of the underground parking space to construct rules for WFC.

(c1) Construction of generation model: a generation model is developed based on the Monoceros plug-in on the Rhino–Grasshopper platform to generate layout solutions based on given plan contour and entrance/exit locations.

(c2) Construction of optimization model: a multi-objective optimization model utilizing the Octopus plug-in on the Rhino–Grasshopper platform is developed to optimize the outputs of the generation model.

(d) Experiments and analysis: two tests are conducted to verify the effectiveness of the generation model and the optimization model separately, and a test is conducted to evaluate the generation and optimization model's applicability to real-world design problems. The accuracy and efficiency of the model are verified through analysis of the test results.

4. Analysis of the Configuration Process for Underground Parking Layout

4.1. Design Process for Underground Parking Layout Configuration

The design process of underground parking is influenced by various factors such as site conditions and design requirements. Consequently, the design process often necessitates multiple modifications to accommodate changes in the overall project. This, in turn, leads to an increased workload and time costs. Based on relevant building design codes, the general design process for underground parking layouts can be summarized as the following steps:

(a) Clarify the design scope of the underground parking and define relevant economic and technical indicators in accordance with the planning requirements and design specifications.

(b) Determine the precise boundaries of the underground parking and strategically determine the number and placement of entrances and exits.

(c) Consider factors such as column network, building structure, location of elevator shafts, and equipment rooms to ensure a logical and efficient arrangement of the driveway and parking spaces.

(d) Continuously adapt the layout of driveway and parking spaces based on overall design adjustments, prioritizing parking efficiency, traffic convenience and other indicators. This stage often involves repeated design adjustments.

(e) Finalize the layout, therefore achieving the final design outcome.

According to the general design process, developing a generative design model for the generation of underground parking layouts is essential because it allows for real-time output of design results that are able to respond to changing design conditions, ultimately improving the overall efficiency of the layout configuration.

4.2. Transformation: From Layout Configuration to Constraint Satisfaction Problem

There are three key characteristics of underground parking layout: first, the layout exhibits a highly grid-like pattern due to the influence of the column network arrangement in the superstructure; second, the composition of parking space elements, such as driveways and parking spaces, demonstrates modularity; third, the design is subject to quantifiable constraint rules due to site conditions, design requirements, and other factors. Based on these characteristics, the layout configuration problem can be abstracted as a constraint satisfaction problem (CSP) represented by a ternary <X,D,C>, where:

X (variable) = {minimum space positions divided by the column network (each accommodating one driveway/parking space unit)};

D (domain of values) = {For each minimum space position, all possible forms of the driveway/parking space unit it accommodates};

C (constraints) = {all adjacency relationships between minimum space units}.

A solution to a CSP is by applying a set of assignments to a group of variables selected from their respective domain of values, which satisfies all constraints simultaneously [35,36]. In the context of underground parking layout configuration, a solution to this problem is achieved when each minimum space cell contains a specific driveway/parking space unit and there are no violations of the constraints. Transforming the layout configuration problem into a constraint satisfaction problem involves leveraging the logical correlation between the constituent units of the building plan to break down the holistic layout problem into adjacency problems between these units, thereby providing conditions for the WFC solution.

5. Modules and Rules

As a procedural content generation algorithm, WFC is capable of efficiently solving constraint satisfaction problems. WFC encompasses three fundamental elements: 1. slots, which are an array that accommodates all the minimal spatial units. In this research, the array in which the slots are located overlaps with the 8.4 m × 8.4 m column network; 2. modules, the smallest spatial units used to generate results; 3. rules, the connections between modules. The three fundamental elements align well with the three key characteristics of underground parking layout, which indicates that WFC is suitable for solving underground parking layout configuration problems. This chapter demonstrates how modules and rules are developed for WFC.

5.1. Modules

During the module construction stage, the underground parking layout is decomposed into several basic space units based on the column network. According to our analysis of underground parking layouts, the modules are categorized into three main types (Figure 4):

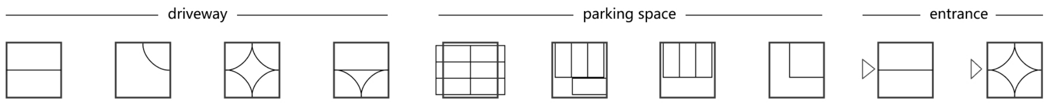


Figure 4. Basic types of driveway, parking space, and entrance modules.

(a) Parking space modules: parking spaces are the foundational components of any parking layout. They are demarcated zones expressly dedicated to the storage of vehicles. These spaces are meticulously planned, considering various aspects such as size, orientation, and layout, to accommodate a diverse range of vehicles. The effective design and organization of parking spaces are pivotal in optimizing the parking facility's capacity and accessibility. According to relevant building design codes, the most economical parking pattern is a six-space module, and there are also four-space and three-space modules.

(b) Driveway modules: driveways in underground parking units facilitate vehicular movement by connecting various parking spaces and enabling the circulation of vehicles. These pathways are crucial for ingress, egress, and inter-space navigation. The judicious placement and distribution of driveways significantly impact the flow of traffic within the parking structure, preventing congestion and traffic bottlenecks. Driveway modules can be categorized into straight sections, turning sections, intersections, and three-way intersections.

(c) Entrance modules: entrances function as the gateway for vehicles entering the facility from the street or ground level. Their design is pivotal in regulating the influx of vehicles and guiding them safely into the parking structure. Entrances play a fundamental role in ensuring the orderly traffic management and smooth transition from external roadways to the underground parking space. Entrance modules can be categorized into straight sections and intersections.

5.2. Rules

After the initial construction of modules, a set of rules is established to indicate their adjacency relationships. In the rule construction stage, we represent the potential neighboring relationships (rules) between all modules. The rules are categorized into four main types (Figure 5):

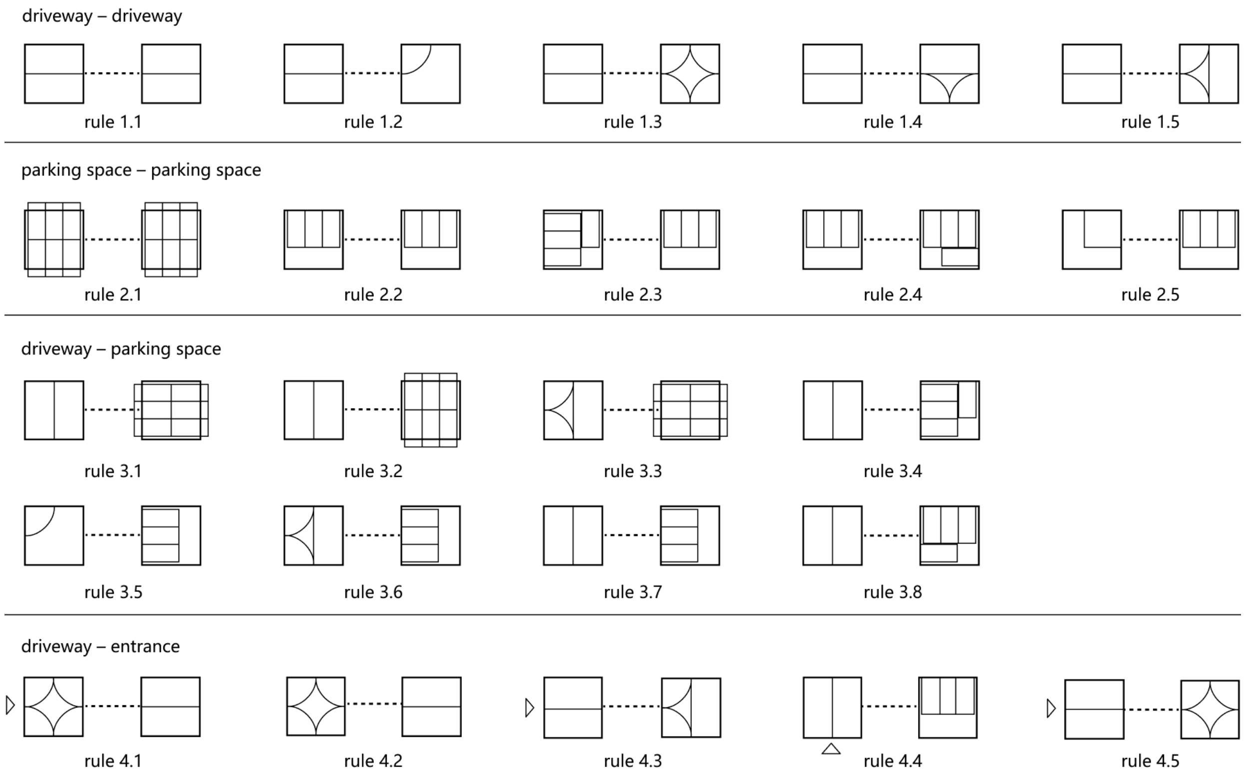


Figure 5. Neighboring relationships (rules) between modules.

(a) Driveway—driveway rules: driveway modules are interconnected to form a transportation network within the underground parking.

(b) Parking space—parking space rules: parking modules are separated by the structural column network, resulting in a juxtaposition of plan configurations. This arrangement maximizes the utilization of space, ensuring an optimal layout of parking spaces.

(c) Driveway—parking space rules: driveway modules seamlessly connect with parking space modules, running perpendicular to the alignment of parking spaces. This connection enhances the accessibility of parking spaces and streamlines the flow of vehicles.

(d) Driveway—entrance rules: the transportation network formed by driveway modules is connected to the outside through entrance modules.

According to the basic mechanism of WFC [14], the generation process begins by randomly selecting a slot and allocating a specific module to that slot (stage 1 in Figure 6). In the subsequent propagation stages (stage 2–4 in Figure 6), the slots adjacent to the defined slots are determined according to the predefined rules. When all slots are defined by a specific module without any rule contradictions, the generation process is considered complete. Otherwise, the generation process will be restarted.

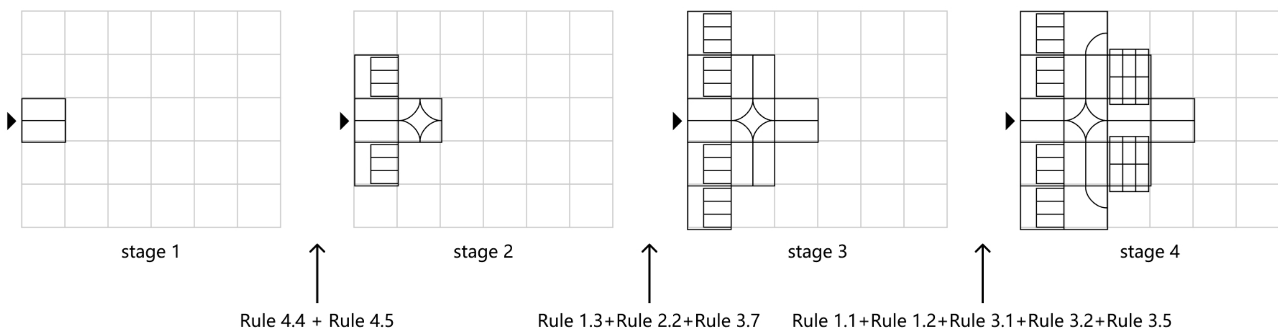


Figure 6. Application of rules during the generation process.

6. Construction of Generation and Optimization Model

6.1. Generation Model

In this study, we utilize Monoceros, a Grasshopper plug-in developed by Subdigital Studio [37], to construct the generation model. Monoceros aims to provide a comprehensive framework for integrating architectural or industrial design with WFC. This plug-in offers full control over input and output data in a Grasshopper-compatible manner while adhering to the principles of WFC.

The generation model encompasses three key sections: data input, WFC solver, and result collector. Within the data input section, all modules and rules, as well as the given plan contour and entrance/exit locations (interpreted as slots in Monoceros), are input in the form of the Rhino digital model. Considering symmetry, rotation and other special spatial situations, the modules and rules developed in the previous chapter are modified and increased in number. For the layout generation of a single-level underground parking discussed in this paper, a total of 77 modules and 278 rules are input in the generation model (Figure 7).

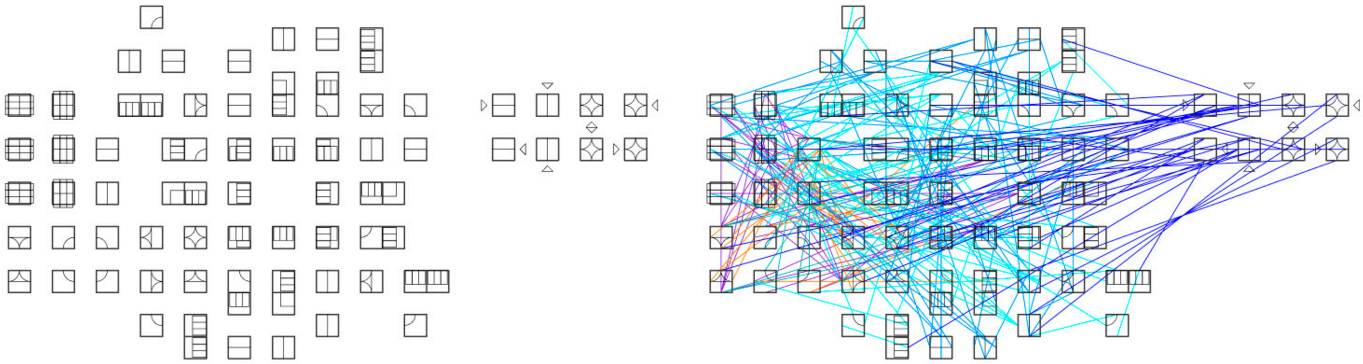


Figure 7. 77 modules (**left**) and 278 rules ((**right**), colors represent types of rules).

These inputs are then fed into the WFC solver. During the iterative process within the WFC solver, multiple observations gradually transition the state of slots from non-deterministic (allowing multiple modules) to deterministic (allowing exactly one module). This process occurs within the solver component, where the slots are observed using pseudo-random numbers until either every slot achieves a deterministic state (success), or any slot reaches a contradictory state (failure). In the event of contradiction, the solver component internally re-attempts the process up to a predefined number of times, with each attempt using the modified random state to produce a distinct outcome. By inputting random values (seeds), the WFC solver can generate a series of results. Results are collected and stored in the result collector section (Figures 8 and 9).

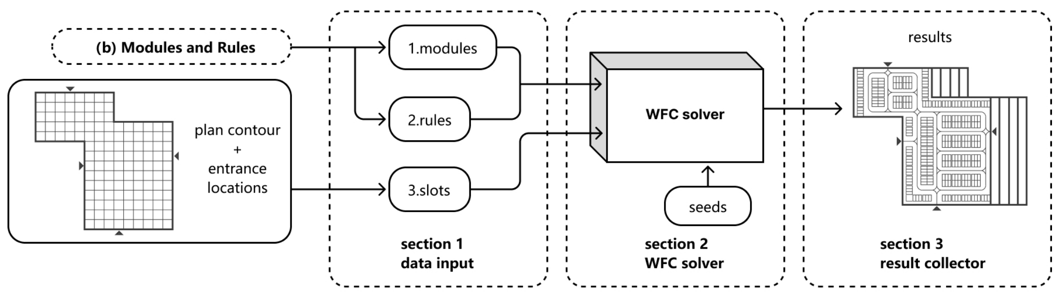


Figure 8. Basic framework of the generation process.

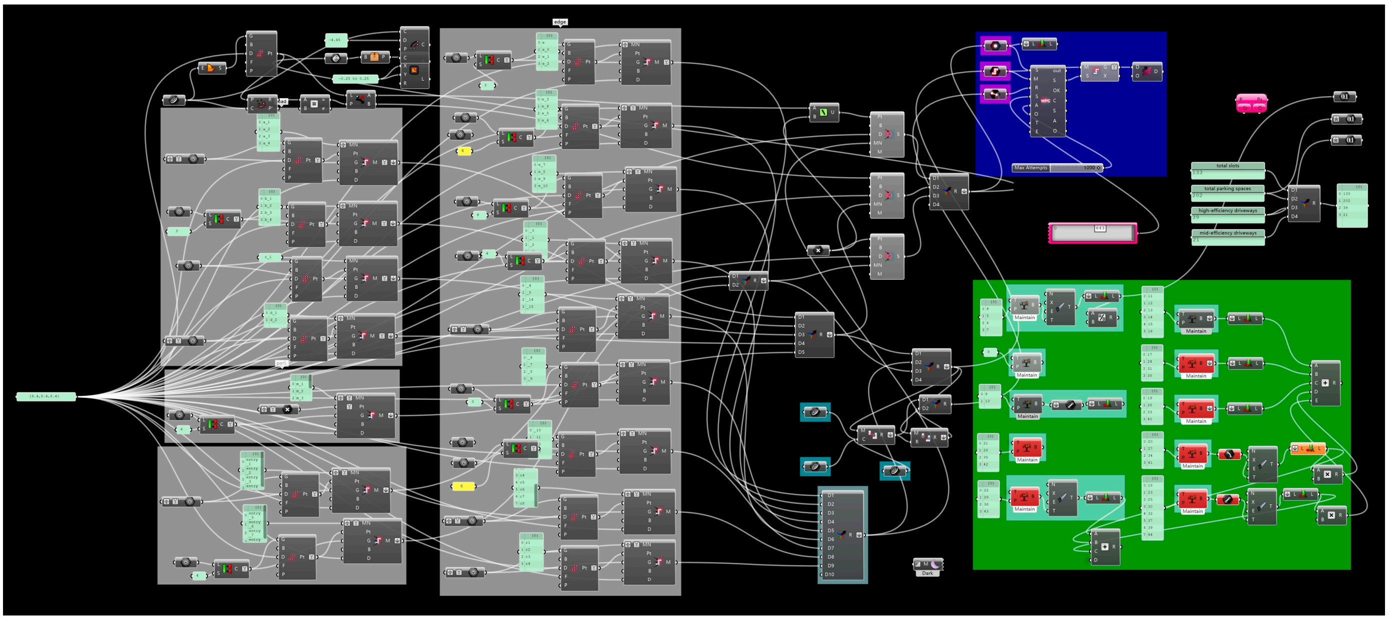


Figure 9. Grasshopper interface of the generation model. The gray area on the left represents the data input section, the blue area on the top right represents the WFC solver section, and the green area on the bottom right represents the result collector section.

6.2. Optimization Model

Because of the robust content generation capability of WFC, the generation model is able to quickly produce a large volume of results. By efficiently filtering these results based on the general evaluation criteria for the underground parking layout configuration, the generation process can be significantly improved in terms of efficiency. In this research, a multi-objective optimization model is developed to filter the generated results and identify the optimal layout solution in accordance with the design objectives for the underground parking layout configuration.

Mathematically, a multi-objective optimization problem can be expressed as:

$$\min_{x \in D}(f_1(x), f_2(x), \ldots, f_k(x)) s.t. g_i(x) \geq 0, i \in [1, M] h_j(x) = 0, j \in [1, L] \tag{1}$$

Let D represent the feasible domain of this multi-objective optimization problem:

$$D = \left\{ x \middle| g_i(x) \geq 0,\ i \in [1, M],\ h_j(x) = 0,\ j \in [1, L] \right\} \tag{2}$$

The goal is to find an x within the feasible domain D, such that all the objective functions *f(x)* attain their minimum values. In this study, the optimization objectives are defined as the maximization of parking efficiency and traffic convenience. The parking efficiency is quantified by the total number of parking spaces, while traffic convenience is quantified by the total number of efficient driveway modules and the total number of redundant driveway modules. By considering the set of all generated results as the feasible domain D, the optimization model aims to identify a specific result x within D that maximizes the optimization objectives. By converting the above two optimization objectives into quantifiable metrics and inputting them into the multi-objective optimization plug-in Octopus in Grasshopper, a number of Pareto-optimal solutions that maximally satisfy the optimization objectives can be obtained among all the generated results (Figure 10).

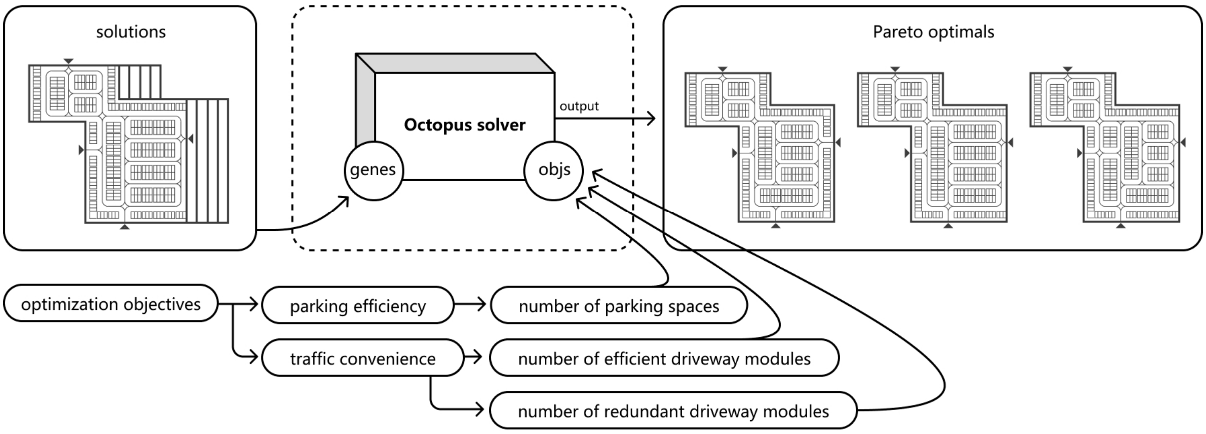


Figure 10. Basic framework of the optimization process.

7. Experiments and Analysis

7.1. Evaluation of the Generation Model

The effectiveness of the generation model was evaluated through an experimental process. As depicted in Figure 11, the generation model successfully produces distinct outcomes when changes are made solely to the location of entrances and exits (a,b), or when both the input plan contour and the location of entrances and exits are altered (a-c-d). Comparatively, a more complex plan contour (d) exhibits a narrower range of outcomes due to the imposition of additional constraints on the rules governing the boundary region, in contrast to a simpler plan contour (a). It is important to highlight that while all the generated results conform to the fundamental principles of underground parking layout configuration, certain outcomes prove to be significantly better than others. For instance, the result with seed = 500 in group (b) exhibits notably lower traffic redundancy compared to those with seed = 596 and seed = 673. This indicates that the identification of a suitable approach to filter and optimize the generated outcomes is imperative.

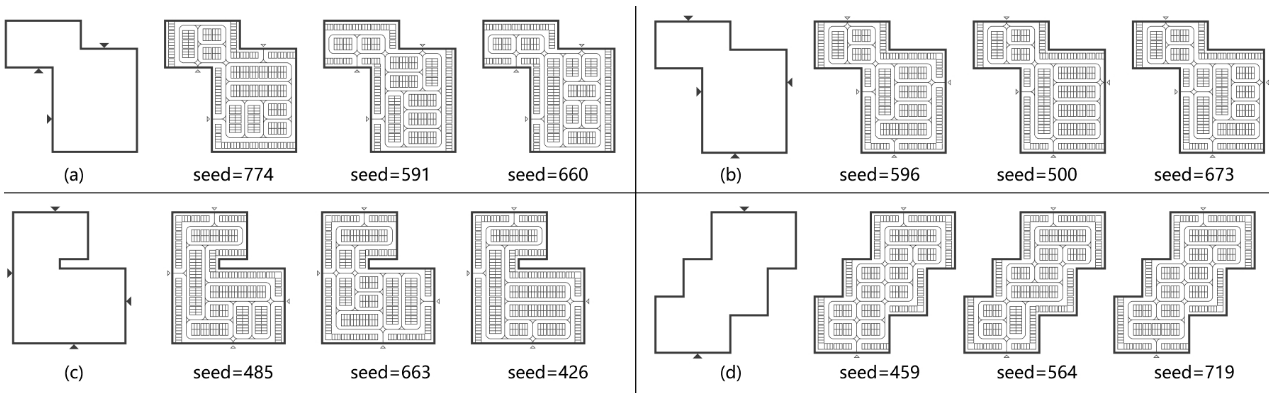


Figure 11. Experimental results of the evaluation of the generation model. Four sets of experiments were conducted (labeled (**a**–**d**) in the figure). For each set, three randomly selected generated results are presented.

7.2. Evaluation of the Optimization Model

To evaluate the efficiency of the optimization model, a multi-objective optimization process was conducted on 300 generation results obtained from the generative model. The outcomes are illustrated in Figure 12. The left side of the figure displays the distribution of the 300 generated results in the multi-objective space, with each axis representing an optimization objective. The shaded gray area signifies the Pareto-optimal front, within which three generated results are identified as Pareto-optimal solutions.

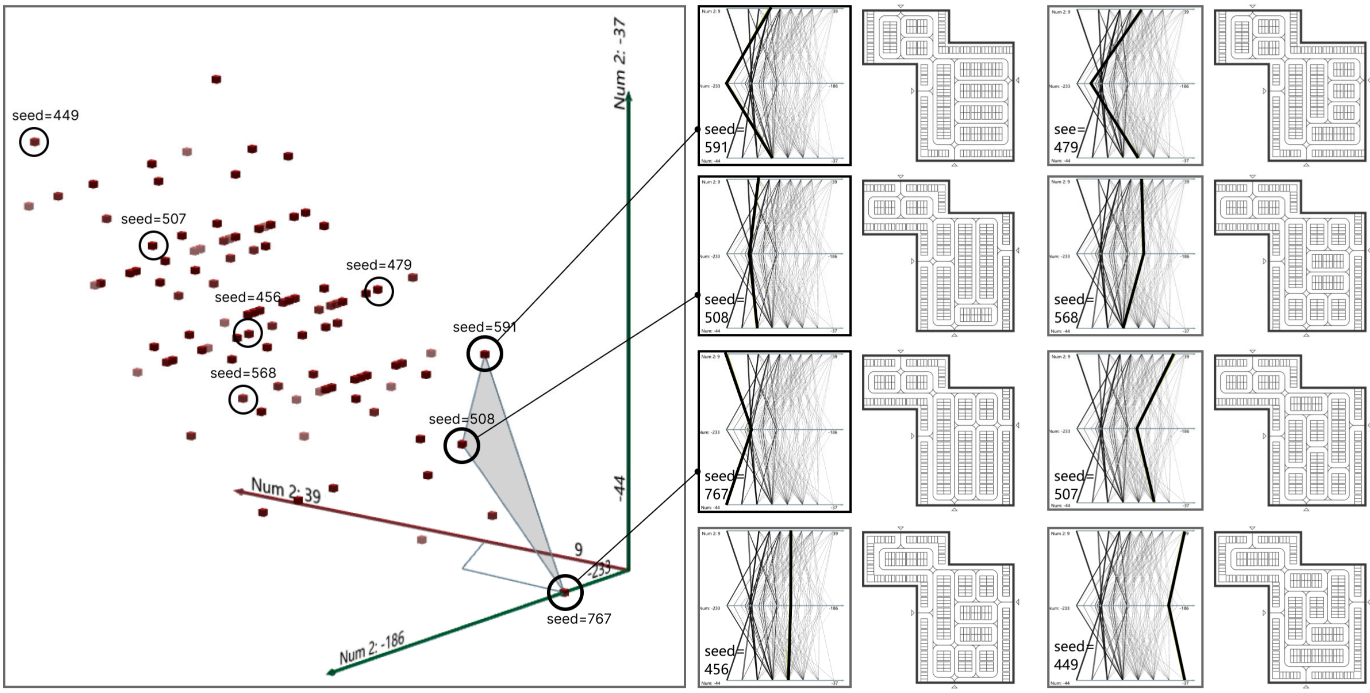
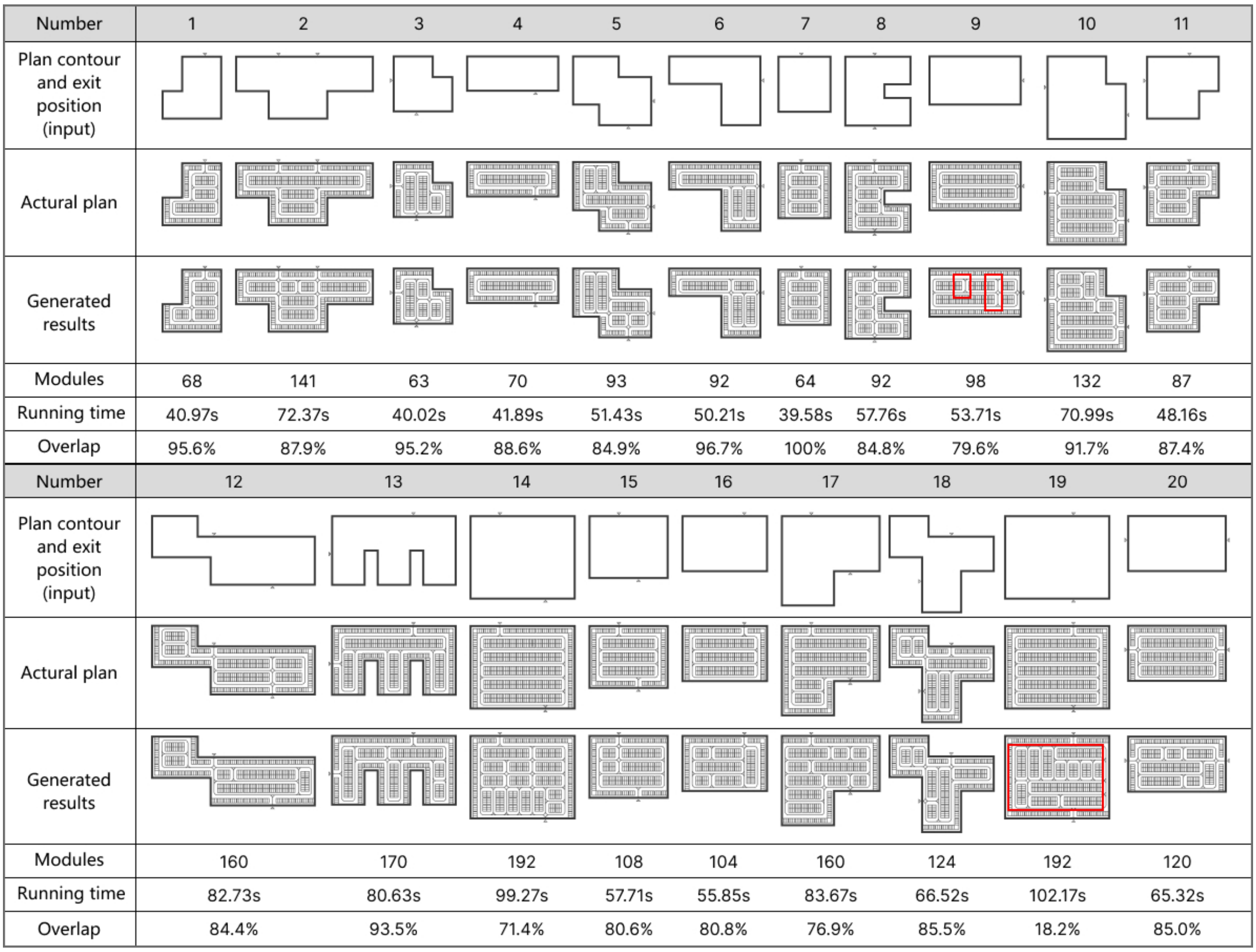


Figure 12. Experimental results of the evaluation of the optimization model.

On the right side of Figure 12, a comparison is shown between the three Pareto-optimal solutions and five randomly selected results. The axial plot indicates the extent to which the corresponding generation results fulfill the optimization objective. The comparison elucidates that the space arrangement of the Pareto-optimal solutions is notably more rational, offering increased parking capacity and fewer redundant traffic units. These results demonstrate that the present optimization model effectively identifies the optimal layout that maximizes the satisfaction of the optimization objective from the output results of the generation model.

7.3. Evaluation of the Generation and Optimization Model

The aforementioned evaluations provide evidence that the generation and the optimization model is competent in generating a multitude of underground parking layout solutions that adhere to the design conditions specified in the given plan contour and designated entrance/exit locations. Furthermore, it successfully identifies the optimal layout solutions according to relevant optimization objectives. To evaluate the model's applicability to real-world design problems, as well as the model's overall accuracy and efficiency, the experiment selects 20 underground parking layouts with an 8.4 m $\times$ 8.4 m column network from the internet. These layouts are then input into the generation and optimization model along with their contours and entrance/exit locations. Subsequently, the output layout solutions are compared to the actual layouts. The experiment involves a total of 20 groups, conducted on a computer equipped with an Intel(R) Xeon(R) W-2123 CPU @ 3.60 GHz. The generation model generates 500 results for each group, which are then optimized by the optimization model. From the several Pareto-optimal solutions obtained, the optimal solutions are manually selected for each group based on the general experiences of underground parking layout configuration. Each group successfully yields one final result. The experimental results are shown in Figure 13.



Number	1	2	3	4	5	6	7	8	9	10	11
Plan contour and exit position (input)											
Actural plan											
Generated results											
Modules	68	141	63	70	93	92	64	92	98	132	87
Running time	40.97s	72.37s	40.02s	41.89s	51.43s	50.21s	39.58s	57.76s	53.71s	70.99s	48.16s
Overlap	95.6%	87.9%	95.2%	88.6%	84.9%	96.7%	100%	84.8%	79.6%	91.7%	87.4%

Number	12	13	14	15	16	17	18	19	20
Plan contour and exit position (input)									
Actural plan									
Generated results									
Modules	160	170	192	108	104	160	124	192	120
Running time	82.73s	80.63s	99.27s	57.71s	55.85s	83.67s	66.52s	102.17s	65.32s
Overlap	84.4%	93.5%	71.4%	80.6%	80.8%	76.9%	85.5%	18.2%	85.0%

Figure 13. Experimental results of the evaluation of the generation and optimization model. The red boxes in result number 9 indicate the locations where redundant driveways encroach upon parking spaces. The red box in result number 19 indicates an area where the layout of parking spaces is cluttered.

This experiment evaluates the accuracy of the generation and optimization model by measuring the degree of overlap between the generated layouts and the actual layouts, and evaluates the efficiency of the model by calculating its average running time.

Regarding the overlap, out of the 20 sets of experimental results, 80% of the generated layouts (16) exhibit an overlap of 80% or higher with the actual layouts. Additionally, 30% of the generated layouts (6) display an overlap of 90% or higher with the actual layouts. The disparity between certain areas of the generated layouts and the actual layouts can be attributed to two factors: first, the presence of redundant traffic layouts (e.g., redundant driveways occupying parking spaces, as seen in the red box in result number 9); second, the irregular arrangement of parking spaces, which deviates from the conventional principles of underground parking space design (e.g., cluttered layout of parking spaces, as indicated in the red box in result number 19). The unsatisfactory distribution of driveway and parking space modules indicates that the generation model still needs to be optimized in terms of rule setting.

In terms of running time, the average running time of the generation and optimization model is 47.08 s when the total number of modules ranges from 50 to 100. For a total number of modules between 101 and 150, the average running time is 64.79 s. Finally, when the total number of modules falls within the range of 151 to 200, the average running time is 89.69 s. Specifically, the generating model requires an average of 166.95 milliseconds to produce a single result. The experimental results indicate that the generation and optimization model can efficiently generate reasonably accurate underground parking layouts within a short timeframe, showcasing high levels of accuracy and efficiency.

8. Discussion

The findings of this study provide insight into various aspects of generative design applications. Firstly, the WFC algorithm has been proven to be effective for generative design. The inherent mechanisms of the WFC algorithm are compatible with the modular and constraint-intensive nature of architectural design. This alignment allows for the integration of the design problem, after being transformed into a constraint satisfaction problem, into the generative model without oversimplification of the design objectives. In contrast, older generative methods sometimes encounter the oversimplification of the design problems, resulting in incomplete generation results [8,9,27].

Secondly, WFC demonstrates adaptability when confronted with a range of design conditions. By decomposing underground parking layouts into basic spatial modules and establishing adjacency rules among them, WFC utilizes these modules and rules to generate layout solutions. Consequently, once the modules and rules are defined, the specific design conditions, including layout size and shape, as well as the location and number of entrances, will no longer impede the precision of the generative design process. This feature is demonstrated in the third experiment, although the design conditions differ from group to group, the generation and optimization model can still successfully produce results for each group. This highlights the advantage of WFC over some data-based methods, which are constrained by specific design conditions [5–7].

Lastly, based on the experimental results, the proposed generative method has successfully demonstrated its efficiency in generating architectural layout solutions within a short timeframe. This is partly due to the algorithmic-level features of the WFC technique, which facilitate effective exploration of the design space [18–20]. Furthermore, the integration of the generation model and the optimization model within the Grasshopper platform enhances the effectiveness of the generative process. This integration allows for real-time collection and input of the generated results into the optimization process, thereby reducing the overall execution time. This addresses the previously identified limitation of slow optimization calculation in earlier research [25]. Overall, the research suggests that WFC, along with the proposed workflow that integrates WFC and MOO, exhibits a significant degree of accuracy, efficiency, and adaptability. These findings underscore the substantial potential of this approach in the domain of generative design.

9. Conclusions and Future Work

This study presents a novel approach utilizing the WaveFunctionCollapse algorithm and multi-objective optimization to generate optimized underground parking layout solutions. A generation and optimization model of underground parking layouts, which incorporates WFC and MOO, is demonstrated in this paper. Experiments show that the model can efficiently generate underground parking layouts that meet the relevant design objectives based on given design conditions. This study showcases the application value of WFC in the field of architectural generative design.

Future research will delve deeper into the advantages and limitations of WFC in this context. First, WFC's module-based and rule-based characteristics make its application not only limited to underground parking layout configuration, but also has exploratory value in the application of residential layout generation, cityscape generation and other architectural generative design problems. Furthermore, the current WFC model is confined to generating building layouts within a single-size orthogonal column network, limiting its practical applicability. Future research will explore its application in irregular column networks and other complex design conditions, alongside further optimization of the existing model.

Author Contributions: Conceptualization, D.L. and K.C.; methodology, D.L. and K.C.; software, K.C.; validation, D.L. and K.C.; formal analysis, D.L. and K.C.; investigation, D.L. and K.C.; resources, Z.X.; data curation, D.L.; writing—original draft preparation, D.L. and K.C.; writing—review and editing, D.L. and K.C.; visualization, D.L.; supervision, Z.X.; project administration, Z.X. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are available on request from the corresponding author. The data are not publicly available due to privacy concerns.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Michalek, J.; Ruchi, C.; Panos, P. Architectural layout design optimization. *Eng. Optim.* **2002**, *34*, 461–484. [CrossRef]
2. Caetano, I.; Santos, L.; Leitão, A. Computational design in architecture: Defining parametric, generative, and algorithmic design. *Front. Archit. Res.* **2020**, *9*, 287–300. [CrossRef]
3. Krish, S. A practical generative design method. *Comput.-Aided Des.* **2011**, *43*, 88–100. [CrossRef]
4. Yan, S.; Liu, N. The development history and applicability analysis of building generative design algorithms. In Proceedings of the 2022 National Symposium on Teaching and Research on Digital Technology in Architecture for Architecture Faculties, Xiamen, China, 3 December 2022.
5. Huang, W.; Zheng, H. Architectural drawings recognition and generation through machine learning. In Proceedings of the 38th Annual Conference of the Association for Computer Aided Design in Architecture, Mexico City, Mexico, 18–20 October 2018.
6. Deng, Q.M.; Lin, W.Q.; Liu, Y.B.; Liang, L.Y. Exploration of generative design of campus general layout based on generative adversarial network: Taking primary school campuses as example. *World Archit.* **2021**, *9*, 115–119+136.
7. Nauata, N.; Chang, K.H.; Cheng, C.Y.; Mori, G.; Furukawa, Y. House-gan: Relational generative adversarial networks for graph-constrained house layout generation. In Proceedings of the Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, 23–28 August 2020.
8. Parish, Y.I.; Müller, P. Procedural Modeling of Cities. In Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, New York, NY, USA, 1 August 2001.
9. Krawczyk, R.J. Architectural Interpretation of Cellular Automata. In Proceedings of the 5th International Conference on Generative Art, Milan, Italy, 11–13 December 2002.
10. Verma, M.; Thakur, M.K. Architectural Space Planning Using Genetic Algorithms. In Proceedings of the 2010 The 2nd International Conference on Computer and Automation Engineering (ICCAE), Singapore, 26–28 February 2010.
11. Wiesenhuetter, S.; Wilde, A.; Noennig, J.R. Swarm Intelligence in Architectural Design. In Proceedings of the Advances in Swarm Intelligence: 7th International Conference, Bali, Indonesia, 25–30 June 2016.
12. Veloso, P.; Krishnamurti, R. An Academy of Spatial Agents: Generating Spatial Configurations with Deep Reinforcement Learning. In Proceedings of the 38th eCAADe Conference, Berlin, Germany, 14 September 2020.
13. Wang, J.; Fan, W.; Zhao, B.; Yang, Y.; Zhang, Z. A Rule-Based Design Approach to Generate Mass Housing in Rural Areas of the North China Plain. *Buildings* **2023**, *13*, 2539. [CrossRef]
14. WaveFunctionCollapse. Available online: https://github.com/mxgmn/WaveFunctionCollapse/ (accessed on 15 January 2023).
15. Shaker, N.; Julian, T.; Mark, J.N. *Procedural Content Generation in Games*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 978–983.
16. Togelius, J.; Yannakakis, G.N.; Stanley, K.O.; Browne, C. Search-based procedural content generation: A taxonomy and survey. *IEEE Trans. Comput. Intell. AI Games* **2011**, *3*, 172–186. [CrossRef]
17. Smith, G. Understanding procedural content generation: A design-centric analysis of the role of PCG in games. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Toronto, ON, Canada, 26 April–1 May 2014.
18. Karth, I.; Smith, A.M. WaveFunctionCollapse: Content generation via constraint solving and machine learning. *IEEE Trans. Games* **2021**, *14*, 364–376. [CrossRef]
19. Karth, I.; Smith, A.M. WaveFunctionCollapse is constraint solving in the wild. In Proceedings of the 12th International Conference on the Foundations of Digital Games, Hyannis, MA, USA, 14–17 August 2017.
20. Sandhu, A.; Chen, Z.; McCoy, J. Enhancing wave function collapse with design-level constraints. In Proceedings of the 14th International Conference on the Foundations of Digital Games, San Luis Obispo, CA, USA, 26–30 August 2019.
21. Ngatchou, P.; Zarei, A.; El-Sharkawi, A. Pareto multi objective optimization. In Proceedings of the 13th International Conference on Intelligent Systems Application to Power Systems, Arlington, TX, USA, 6–10 November 2005.
22. Mukkavaara, J.; Sandberg, M. Architectural Design Exploration Using Generative Design: Framework Development and Case Study of a Residential Block. *Buildings* **2020**, *10*, 201. [CrossRef]
23. Huang, X.; Yuan, W.; White, M.; Langenheim, N. A Parametric Framework to Assess Generative Urban Design Proposals for Transit-Oriented Development. *Buildings* **2022**, *12*, 1971. [CrossRef]
24. Gerber, D.J.; Lin, S.H.E. Designing in Complexity: Simulation, Integration, and Multidisciplinary Design Optimization for Architecture. *Simulation* **2014**, *90*, 936–959. [CrossRef]

25. Nagy, D.; Lau, D.; Locke, J.; Stoddart, J.; Villaggi, L.; Wang, R.; Zhao, D.; Benjamin, D. Project discover: An application of generative design for architectural space planning. In Proceedings of the Symposium on Simulation for Architecture and Urban Design, Toronto, ON, Canada, 22–24 May 2017.
26. Nagy, D.; Villaggi, L.; Benjamin, D. Generative urban design: Integrating financial and energy goals for automated neighborhood layout. In Proceedings of the Symposium for Architecture and Urban Design Design, Delft, The Netherlands, 4–7 June 2018.
27. Carrasco, M.P.; Pecanac, M. Innovative CAD-based Application for Parking Lot Design. In Proceedings of the 2005 Annual Conference of Transportation Association of Canada (TAC), Calgary, AB, Canada, 18–21 September 2005.
28. Yu, G. Underground Garage Generative Design Based on Reinforcement Learning. Master's Thesis, South China University of Technology, Guangzhou, China, 2020.
29. Townscaper. Available online: https://www.townscapergame.com (accessed on 8 November 2023).
30. Stojanovski, T. Townscaper (Software review). *Urban Morphol.* **2022**, *26*, 113–115. [CrossRef]
31. Oisin: Wave function Collapse for Poetry. Available online: https://github.com/mewo2/oisin/ (accessed on 18 January 2023).
32. Wave Function Collapse. Available online: https://marian42.itch.io/wfc/ (accessed on 24 January 2023).
33. Phan, H.T.T. Exploring Wave Function Collapse as a Design Tool. Ph.D. Theis, University of Cincinnati, Cincinnati, OH, USA, 2021.
34. Van, A.; Levi; Rafael, B. Declarative procedural generation of architecture with semantic architectural profiles. In Proceedings of the 2020 IEEE Conference on Games (CoG), Osaka, Japan, 24–27 August 2020.
35. Fan, R.M.; Zhao, C.Y.; Li, F.L. Heuristic backtracking algorithm to solve constraint satisfaction problems. *Appl. Res. Comput.* **2021**, *38*, 1438–1442.
36. Russell, S.J. *Artificial Intelligence a Modern Approach*, 3rd ed.; Pearson Education, Inc.: Hoboken, NJ, USA, 2010; pp. 202–227.
37. Monoceros. Available online: https://monoceros.sub.digital/ (accessed on 28 December 2022).