Article

Generative Design Methodology and Framework Exploiting Designer-Algorithm Synergies

Luka Gradišar *, Robert Klinc, Žiga Turk and Matevž Dolenc

Faculty of Civil and Geodetic Engineering, University of Ljubljana, Jamova 2, 1000 Ljubljana, Slovenia
* Correspondence: luka.gradisar@fgg.uni-lj.si

Abstract: Designing is a problem-solving activity. The process is usually iterative: a solution is proposed, then analysed and tested until it satisfies all constraints and best fulfils the criteria. Usually, a designer proposes a solution based on intuition, experience, and knowledge. However, this does not work for problems they are facing for the first time. An alternative approach is generative design, where the designer focuses on iteratively defining a problem with its constraints and criteria in the form of a parametric computational model, and then leaves the search for the solution to the algorithms and their ability to rapidly generate and test several alternatives. The result of this approach is not only a set of solutions embedding implicitly the knowledge but also a model where problem-defining knowledge is quite explicit. The idea of the proposed approach is the exploitation of synergies between the designer and the algorithms. The designer focuses on problem definition and the algorithm focuses on finding a solution, showing that the capacity of the generative approach to replace the designer is limited. In the paper, we first present the framework of generative design, then apply the process to a case study of designing an efficient shading solution, and in the end, we present the results and compare them with the traditional approach. The approach is general and can be applied in other areas of engineering. It is relevant both to designers as well as software developers who are expected to take this approach further. More theoretical work is needed to study problem definitions as a form of knowledge representation in engineering.

Keywords: generative design; optimisation; computational design; parametric modelling; BIM; automation; shading study

check for updates

Citation: Gradišar, L.; Klinc, R.; Turk, Ž.; Dolenc, M. Generative Design Methodology and Framework Exploiting Designer-Algorithm Synergies. *Buildings* **2022**, *12*, 2194. https://doi.org/10.3390/buildings12122194

Academic Editors: Yuan Chen, Xianfei Yin, Bo Xiao, Yinghua Shen and Hexu Liu

Received: 19 October 2022
Accepted: 8 December 2022
Published: 12 December 2022

1. Introduction

In recent years, technological development has accelerated drastically. This is reflected in the increasing use of digital tools being introduced in the construction industry. The work of designers over time has already changed by replacing pen and paper with computer-aided design (CAD) tools and now with Building Information Modelling (BIM). Building design is becoming more comprehensive, interconnected, and coordinated, but it still involves a lot of manual work and especially rework. Traditionally, the designer proposes the solution based on their experience, previous projects, and creativity under the given conditions and objectives, while the computers are to help with presentation, documentation, and analysis, but other alternatives may be considered to assist in design development.

One such approach is generative design, where the main idea is the collaboration between the designer and the computer/algorithm, which has complementary capabilities, such as, on the one hand, sorting large amounts of data, generating and analysing a large number of results, finding optimal solutions, and iterative improvement of the solution [1,2]; on the other hand, real world experience, deep knowledge, understanding, and history of working in the field. In this process, we shift the focus from creating the design solution to defining the design problem with its constraints and criteria, where a well-defined problem can then be analysed with various algorithms to generate a large set of results from which the designer can find the best suited solution [2].

Early concepts of today's generative design were explored by Frazer [3], and use cases in engineering were presented in 2002 [4], in which the thermal and light performance of a building was studied using genetic algorithms that controlled the size of windows and generated several possible solutions, which were then verified in a detailed thermal analysis program. It was suggested that the same procedure could be applied to other building design problems. The same authors [5] extended the earlier work by developing a generative design system capable of generating different architectural solutions, evaluating their performance with building energy simulation software, and searching for new solutions with a genetic algorithm, all in one iteration. They pointed out the limitations of computational models for design representation and also questioned the authorship of the design between the intelligent software and the architect. Other early work [6] explored the use of generative design for structural design. They utilised truss components with parametric relations and constraints used with genetic algorithms to generate various cantilever systems that were analysed and evaluated based on mass to find the optimal design.

Further exploration of the generative design process was defined by Krish [7], where it was stated that the generative process consists of three components: a design schema, a means of creating variations, and a means of selecting a desirable outcome. The designer's role in this process focuses on continuously modifying the model and constraints based on the judgment of the results until a viable design solution is found. In this approach, a parametric model is used in conjunction with CAD to generate a set of distinctive designs that are evaluated and filtered based on multiple objectives to produce a smaller set of possible designs that the designer can use for further development. This method is demonstrated in the development of an MP3 player and a coffee table from an early conceptual design to a detailed design. Sign and Gu [8] reviewed various generative design systems that use different solvers for design generation, such as Shape Grammar, L-System, cellular automata, genetic algorithms, and swarm intelligence. Although the different systems share common features, they are limited to a single solver and tailored to a specific design task. On this basis, they proposed an integrated generative design framework that supports multiple techniques. In this framework, the designer creates a knowledge base for design evaluation and a generation procedure from which design solutions are generated and evaluated against the knowledge base.

Generative design concepts have been used for various applications. In structural engineering, generative design was used to investigate different planar space truss systems [9], and for the spatial-structural optimisation, two different techniques were used: topology optimisation and evolutionary optimisation [10]. The same principles were used for generating different floor layouts and evaluating them according to multiple objectives to find an optimal floor plan of a single-story building [11], and designing the office layout based on the worker preferences, productivity, daylight, and outside view [12]. Various generative design techniques were also used to explore creative shape variations [13] and for searching topological designs with optimal thermal paths [14]. Recent research is also focused on using artificial intelligence methods to generate designs, as shown by the authors Yoo et al. [15], where they used neural networks to generate various designs of a 2D wheel.

Although generative design is seeing its increased use in design exploration, there are still some misconceptions, such as equating generative design with topology optimisation or AI that will replace the designer. The aim of this paper is to present a clear generative framework and process relevant to the current use of computational and BIM models, and to show its application on the case study of an efficient shading system.

2. Generative Design

Generative design is an iterative design process in which the designer defines the computational model and its design goals; next, the computer automates the generation of a variety of design alternatives, which are to help a designer better understand the design and the relationships between parameters and objectives, thus helping to further develop the design or find the final design solution [7].

Usually, the designer does not know which design solution best fits the given conditions. The designer proposes a design and then analyses it using various tools to determine whether the proposed solution is suitable or not. This works for well-known problems but fails for new problems due to lack of experience or for problems with a large number of possible solutions. The designer may also not know if the solution found is the best or if it can be improved. With iterations, a small sample can be produced to compare different design solutions, but this takes time that we sometimes cannot afford.

In the generative design approach, we define the problem and its objectives in a form of a computational model and let the algorithms generate multiple design alternatives. This automates the iterative process of creating and analysing design solutions, resulting in a wider range of viable solutions. With the help of a larger sample, we can gain knowledge about the behaviour and performance of the design problem, which allows us to further develop the model that will lead to the suitable solution.

2.1. Generative Design Framework

We can identify the three main features (Figure 1) that make up the generative design framework: Computational Model, Generator, and Design Evaluation.

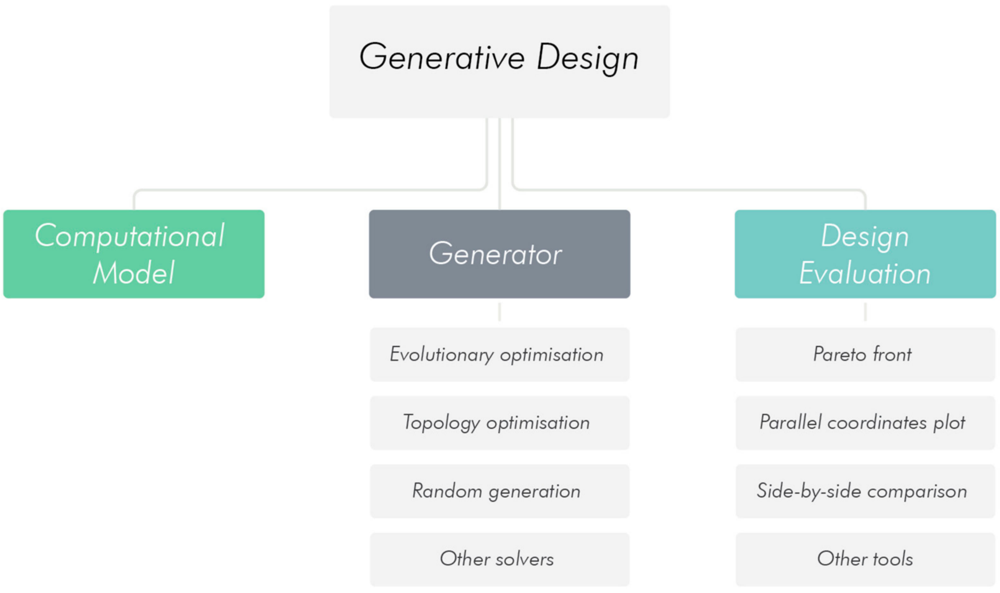


Figure 1. Generative design framework components.

2.1.1. Computational Model

In generative design, building a computational model is the main task that takes the most focus and time. This model can be viewed as a description of a design problem that includes all design parameters, constraints, and objectives. It can be understood as a deterministic function that, given a certain input, produces a certain output, which may be in the form of a shape, geometry, image, or data. Such models consist of a set of rules or instructions on how the model should behave given the initial conditions [16]. They require a computational environment in which such a model can be defined along with the input variables, analysis, and the output in order to objectively evaluate different solutions based on the given objectives [4]. In the BIM design process, the use of building information models can be seen as a starting point upon which a computational model can be built to utilise its geometry and information.

2.1.2. Generator

The generator is a collection of algorithms that generate and search for design solutions. It undertakes the process of changing design parameters, and testing and evaluating the results against each other to provide feasible solutions to the specified design problem instead of the designer. Using the parametric computational model, the generator automatically searches among the combinations of input variables to meet the design goals by utilising different solvers. As there are usually a large number of possible combinations of input variables and, therefore, design alternatives, the most commonly used solvers in building design are optimisation algorithms [17,18], such as evolutionary algorithms [4,5,9–12], particle swarm optimisation [19,20], topology optimisation [10,21], or a combination of them [13,14,22]. However, any other solver that can generate different design alternatives can be used [8], whether it is a simple random generator or a more sophisticated solver [23].

The optimisation algorithms frequently used in generative design are evolutionary algorithms, from which the initial methodology principles were developed [2,24]. This is mainly due to the nature of the design problems, which involve a large number of parameters, multiple local optima, discrete and noisy objective functions, and multiple objective functions that the evolutionary algorithms can handle with a reasonable convergence rate. They operate on the principle of evolution, where the successful members of the population survive, reproduce, evolve, and improve with each new generation until no improvement can be achieved with the next generation. A popular evolutionary algorithm is the genetic algorithm, which works in the following steps (Figure 2):

1. Randomly generate the first generation in the solution space;
2. Rank all members of the population by performance and keep the percentage of the most successful;
3. Generate a new generation by crossover and mutation of the previous population;
4. Repeat the steps until there is no improvement in the next generations.

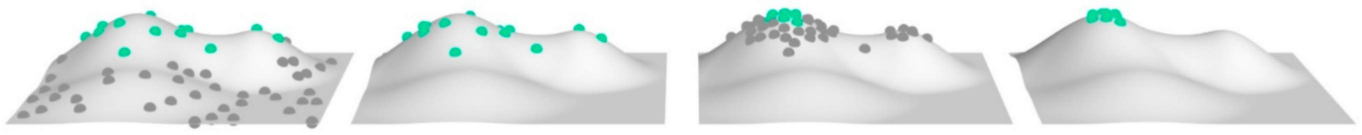


Figure 2. Example of finding the maximum with genetic algorithm. From left to right, the iterative improvement of solution performance can be seen.

Crossover combines genes (parameter values) from one or more parents to produce a child of a new generation, and mutation introduces random genes or values to prevent evolution from getting stuck at the local optimum and to ensure convergence to the global optimum [25]. By populating a solution space and using these operations, the genetic algorithm can quickly transverse among possible solutions and efficiently find the global extreme (maximum or minimum), without having to compute the entire solutions space. Using such an optimisation algorithm can lead to interesting and less distinct solutions, which we might never think of [26].

2.1.3. Design Evaluation

The generator produces many possible design solutions, each with associated input and output variables or a geometry, by which we can compare them to each other and see how different designs behave. As this generates a considerable number of results, it is beneficial to use appropriate tools to analyse them, such as the Pareto front, parallel coordinates plot, side-by-side comparison, clustering, and other data analysis tools. By evaluating different design alternatives, we gain a better understanding and knowledge of the design and how it performs, which, in turn, helps us to further develop the computational model or find the solution to the design problem [1].

When employing generative design, we often deal with complex design problems that have more than one design goal, which can be either objective (e.g., cost, time, volume, and efficiency) or subjective (e.g., appearance, purpose, and aesthetics), based on the designer's or client's preferences. Such optimisation problems are called multi-objective optimisation, where usually more than one optimal solution exists, which we call Pareto-optimal solutions and are located on the Pareto front [25]. For such solutions, we can say that there is no solution that is better in all objectives, but there can be those that are better in individual objectives, which can be seen in Figure 3, where we cannot find any point in the solution space that is better than the green points in both objectives, while we can for the grey points. Between Pareto-optimal solutions, we cannot say which is better than the other; while some are better in one objective and worse in others, they are all optimal given multiple objectives.

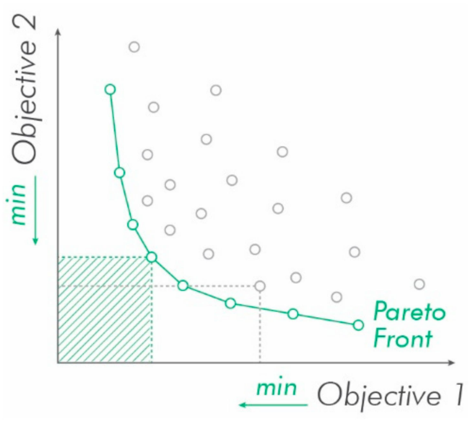


Figure 3. Two-dimensional example of Pareto front with two objectives that we want to minimise.

2.2. Design Methodology

To approach a new design problem from the generative approach standpoint, a parametric computational model must be created from which a generator can produce and analyse various design alternatives. This can be a difficult step, given that the problem has to described with a set of rules and equations to model the correct behaviour, so the quality of the design solutions is strongly influenced by the quality and completeness of the computational model prepared by the designer.

Once the computational model is built, which includes input variables that modify the design and output variables that follow the objectives, the generator is used to generate different design alternatives, each given a fitness score allowing the generator to evaluate and iteratively improve the design through optimisation. This step is performed independently, apart from setting-up the solver and its optimisation parameters, resulting in a set of different possible design alternatives.

Running a generator is best performed multiple times to generate a wide variety of possible design solutions that we need to review and evaluate to find a suitable solution or adjust the model. As analysing a large number of solutions individually seems to be difficult, we use the Pareto front to identify the optimal solutions or as a reference to compare other possible solutions with how close they are to the optimal. Because there are multiple optimal solutions, additional weight functions can be assigned to each objective, depending on how important we consider it to be, to highlight optimal solutions that perform better in those objectives. To better understand how the input variables affect the design, we employ the parallel coordinates plot, where all (input and output) variables for each design solution are shown side-by-side. Additional boundaries can be added to the variables to limit the design solutions to a smaller sample, making it easier to examine designs individually and compare them visually by displaying them side-by-side.

The entire process is iterative, as shown in Figure 4, where by studying the generated solutions, we can evaluate the constructed computational model and its behaviour, and make adjustments or improvements to the model until we find a suitable solution to the design problem.

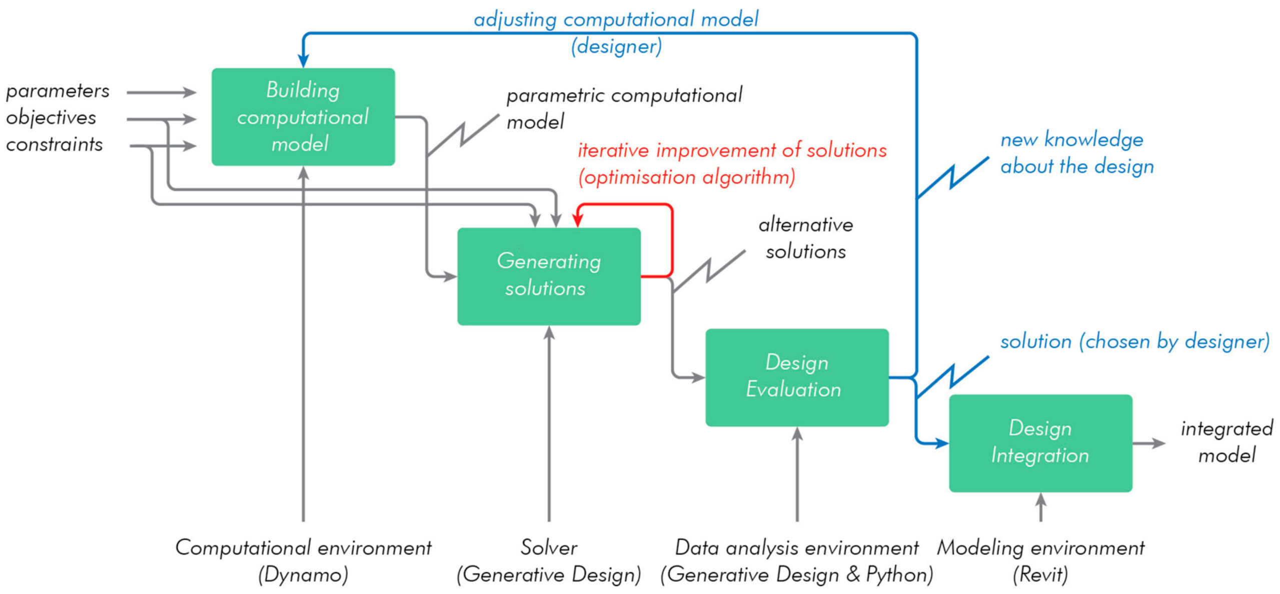

adjusting computational model
(designer)
parameters
objectives
constraints
Building computational model
parametric computational model
iterative improvement of solutions
(optimisation algorithm)
new knowledge about the design
alternative solutions
Generating solutions
solution (chosen by designer)
Design Evaluation
Design Integration
integrated model
Computational environment
(Dynamo)
Solver
(Generative Design)
Data analysis environment
(Generative Design & Python)
Modeling environment
(Revit)

Figure 4. IDEF0 diagram of the generative design process.

3. Case Study

Generative design was used to develop a shading solution for a building with a large glazed surface. The project was part of a study of what a modern, sustainable university building might look like. The site considered for the building (Figure 5a) was the San Francisco State University area in California, USA, where a moderate climate with a high number of sunny days can be expected. Specifically, an average annual amount of sun hours of 3070 h and 259 sunny days were considered, with average annual maximum temperatures of 17.8 °C and minimum temperatures of 10.5 °C. Because of this, it was important to consider energy efficiency as part of the building design [27,28]; therefore, a shading system was needed to reduce energy consumption for cooling and heating. This led to the design idea of passive shading elements that would illustrate the natural appearance of the wood grain texture, complementing the timber design of the building.

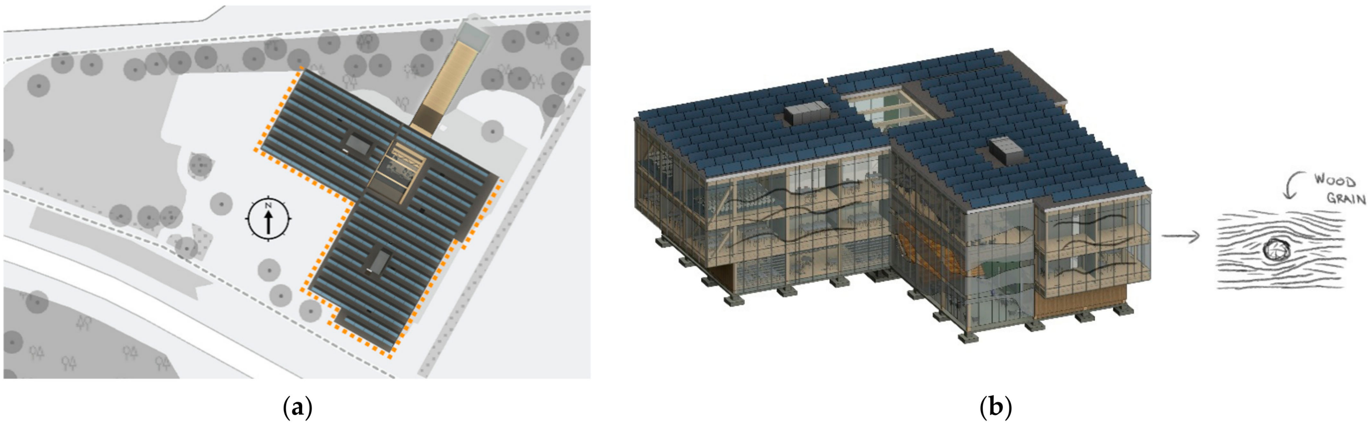


Figure 5. (**a**) Design situation. (**b**) Design problem illustration.

To achieve this, horizontal and vertical louvres were considered. Preliminary analysis showed that the horizontal louvres performed better, which is consistent with other studies [29,30]. Additional visual inspection was performed using virtual reality, and it was found that the vertical design created a much more closed space than the horizontal orientation, which was then ultimately chosen for a detailed study. Subsequently, a generative approach was used to find an efficient layout and shape for the shading elements to satisfy identified design goals while still maintaining the visual complexity of the design.

The shading elements were created using a computational model that contains horizontal elements following the form of wood grains (Figure 5b). With this design, we wanted to achieve three main design goals: First, we wanted to create an efficient shading system for the building; second, we wanted to reduce the amount of material needed to construct the elements; third, we wanted to give the building an interesting appearance. To develop a good generative design, it is important to clearly define a design problem and its design goals; for this purpose, illustrations are recommended.

For the generative design, we needed an environment to build a computational model, a generator to generate design alternatives, and an interface to evaluate the design results. The software applications used for the presented use case were Autodesk Revit, Dynamo, Project Refinery, and a code editor for the Python programming language. In Revit, we had a BIM model to which we wanted to add an optimised shading system. Complex geometries such as the discussed shading elements cannot be modelled by hand, or it would take too much time. Therefore, the modelling was performed with Dynamo, a software for computational design [31]. In Dynamo, visual programming was used to build the computational model and manipulate with the data. To generate different design alternatives, Project Refinery was used, an add-on for Dynamo that has various solvers to generate and an interface to examine the results [32]. It is also included in the Revit 2021 and newer versions under the name Generative Design [33]. For the additional design evaluation, Python was used to organise the results using the Pareto front, parallel coordinates plot, and additional plots to better understand the relationships between variables and their impact on the design. This helped us to further develop the model and refine the overall design of the shading elements, as the process of creating the computational model and evaluating different designs was iterative. Once the final design parameters were found, the geometry was then exported back to Revit and added to the BIM model.

3.1. Computational Model

The starting point was a BIM model of the case study building. From the model, the façade surfaces, their orientation and position were extracted into the Dynamo environment. They were baselines on which the computational model of the shading elements was built. The model was created by dividing the façade surface into horizontal elements. A more complex geometry was derived from the horizontal elements by incorporating shape curves, trigonometric functions, and the addition of smaller openings. Figure 6 illustrates the process of building the computational model in Dynamo. This was done by connecting various Dynamo nodes together (Figure 7a), which represent specific functions that describe how the model looks and behaves [34].

Parameters that change the design were simplified to the number of horizontal divisions and the offset multiplier that represents the width of the elements. To evaluate the design alternatives, the sunlight analysis was programmed. Here, we simulated the sun rays in the sun vector direction at the different times and counted the number of sun rays that were either blocked by the geometry or transmitted through, as shown with different colours in Figure 7b. In addition, the total volume was extracted from the model geometry for the cost evaluation.

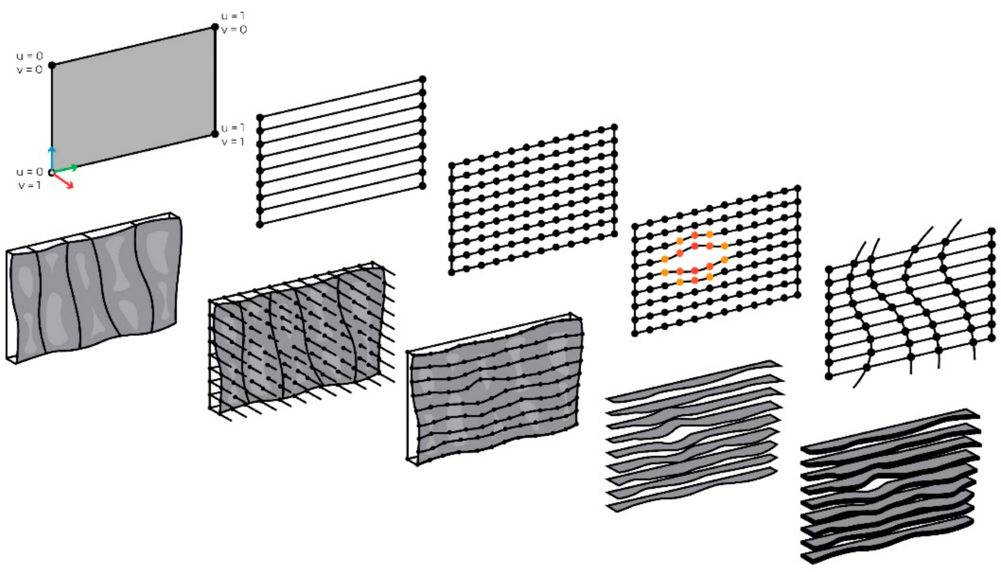
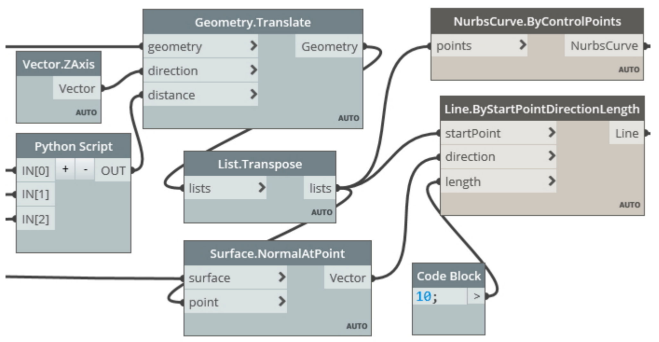
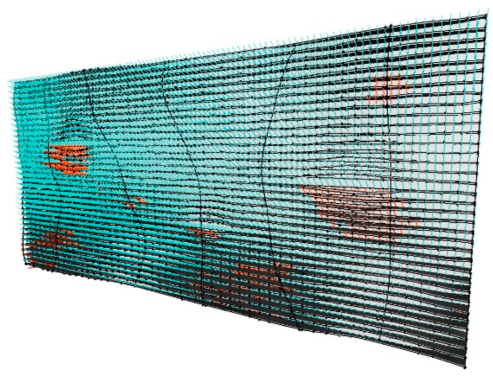


Figure 6. Computational model build process in Dynamo.



(**a**)



(**b**)

Figure 7. (**a**) Example of code blocks forming Dynamo script for generating shading elements. (**b**) Result of the shading analysis, where the blue colour represents the shaded sunlight and the red colour represents the light that passes through the gaps in the shading elements.

For the purpose of evaluating design alternatives, a fast and simplified shading analysis was developed. Here, the sun rays were simulated in reverse, i.e., they began at the surface and were projected towards the sun direction. Where the rays intersected with the shading elements, we considered the sunlight to be blocked. To speed up the calculations, two simplifications were made: first, the sun rays were generated at the outermost edge where they could come into contact with the surface, i.e., exactly at the edge of the horizontal element (Figure 8a), and second, the intersection check was performed only between the top element and the bottom row of sun rays (Figure 8b). The ratio of blocked or passing sun rays was used as a performance measure against which design alternatives were evaluated (Figure 8c).

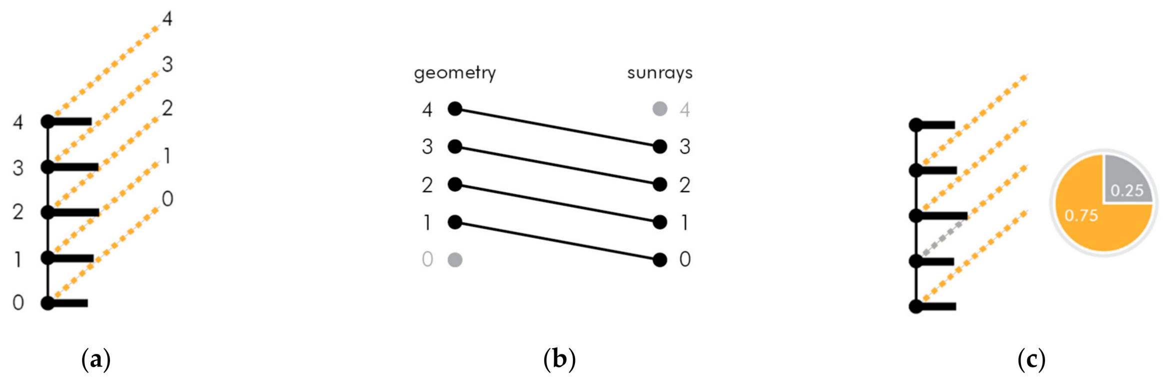


Figure 8. (**a**) Sun rays projected from the surface towards the sun. (**b**) Matching shading elements with sun rays a row below for intersection check. (**c**) Performance measure as the ratio of blocked or passing sunrays.

3.2. Generator

In the next step of the generative design, we determined what type of generator we would use and matched it with a compatible computational model. For the presented problem, we wanted to perform a multi-objective optimisation with a solver using the evolutionary algorithms. To evaluate the different possible shapes, we defined output variables that represented the objectives we wanted to maximise and minimise. Based on the design goals, three targets were selected for optimisation: (i) the total material usage that we wanted to minimise, and (ii) the shading efficiency in summer and (iii) in winter. We wanted to block direct sunlight in the summer and not block it in the winter in order to warm the interiors. The following objectives were defined in the computational model as numerical values, by which we were able to compare and rank different design alternatives. Figure 9 shows the input and output variables, which were as follows:

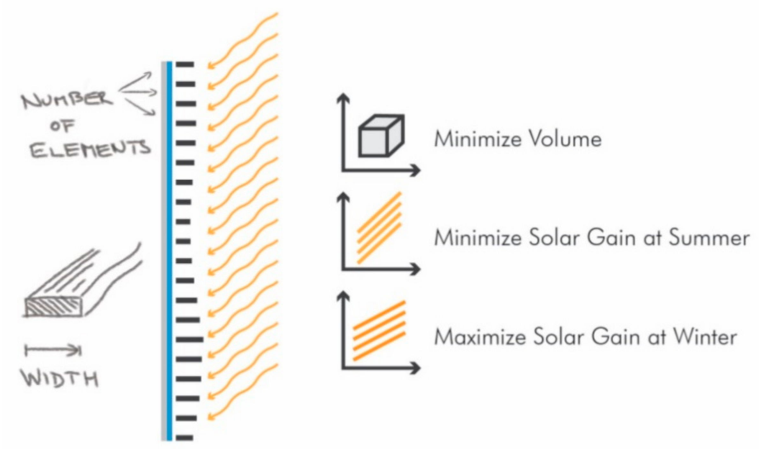


Figure 9. Design optimisation objectives.

Input variables:

- Number of horizontal elements;
- Width of the elements.

Output variables:

- Total volume;
- Number of sunrays passing through the shading elements in summertime;
- Number of sunrays blocked by shading elements in wintertime.

Through Project Refinery, the computational model was used to create different design solutions with associated output variables by changing the input variables in order to find optimum output values.

3.3. Design Evaluation

From the very beginning, it was decided to have as few design variables as possible. If the problem had many design parameters and goals, the best approach would be to break the problem down into smaller ones and solve them step-by-step. This way, we could focus on the whole problem with the aim to find input variables that created an efficient design and met the design goals. By analysing the output variables, we were able to compare different designs and their performance. This helped us understand how the overall design behaves and how efficient it can be.

From the generated results and their output values, we plotted the Pareto front, as shown in Figure 10. The Pareto front gave us an indication of which design alternatives were optimal. This allowed us to see how close we were to the optimal design when we examined alternatives in other design goals, such as visual appearance. We also decided that instead of the total volume, i.e., the initial cost of the elements, we should focus on efficiency, which would be important more over time. Here, the use of an interactive parallel coordinates plot in Figure 11 and the visual comparison in Figure 12 helped in the additional evaluation of the solutions.

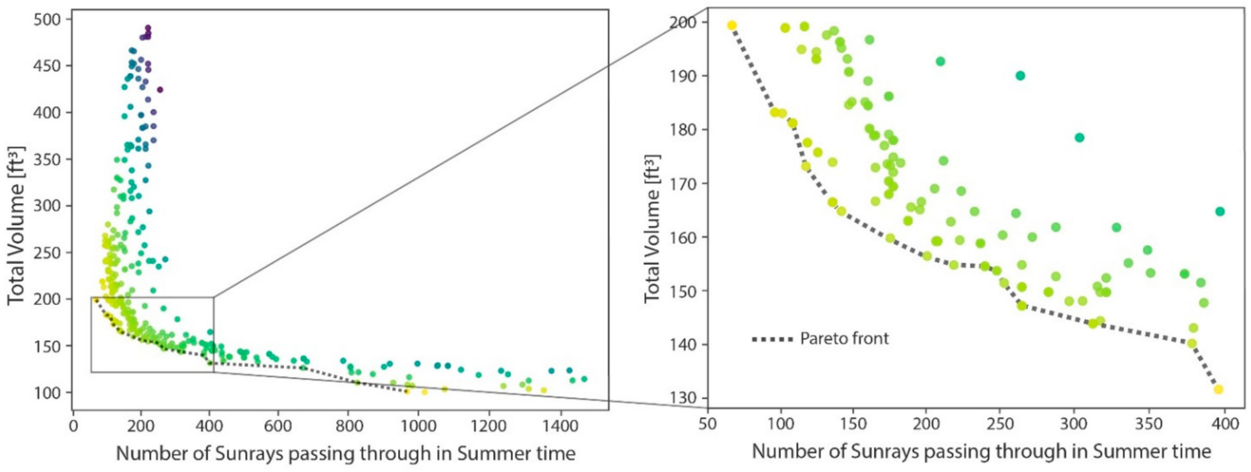


Figure 10. Pareto Front constructed on the analysis results.

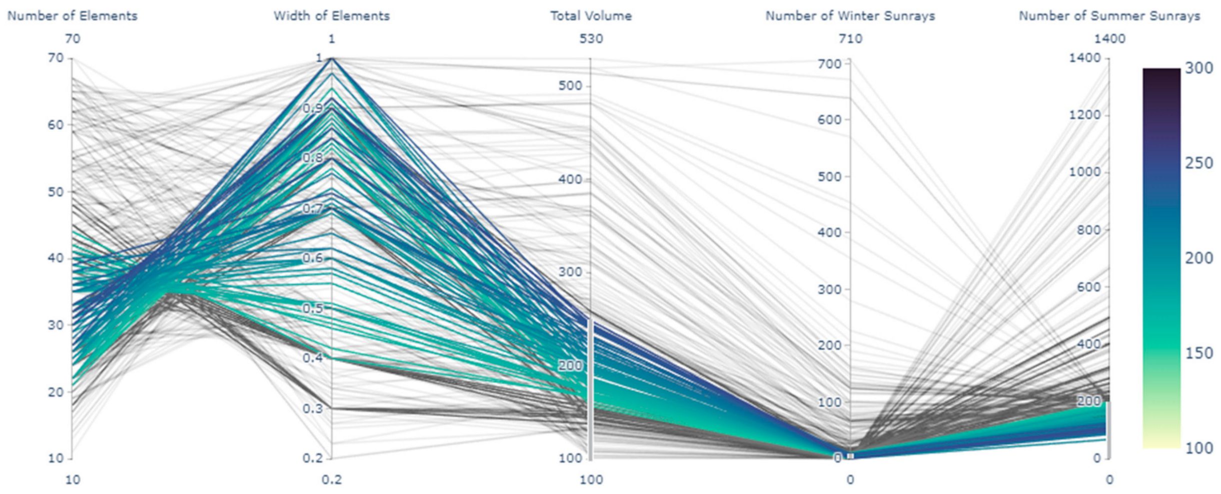


Figure 11. Parallel coordinates graph for filtering results.

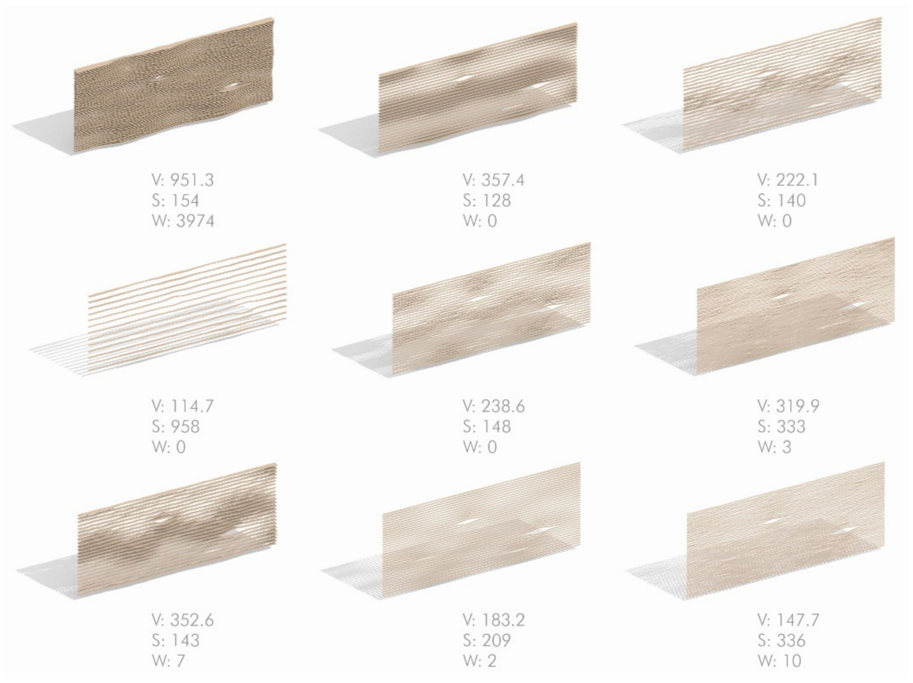


Figure 12. Visual comparison between the design alternatives.

With the parallel coordinates plot, we filtered the results. Here, we added additional constraints for each variable to limit the result pool. This helped us to understand the relationships between the variables. One of these relationships showed that the width of the elements had a greater influence on the efficiency than the number of elements, while the total volume of elements was still not significantly affected compared to a higher number of elements and a smaller width. This helped us in the design process, as we focused more on the width than on the number of the elements in the next iterations of adjusting the computational model. This helped us to evolve the design based on the initial design evaluation. By comparing and exploring different design alternatives, we were now prepared to find a final design solution.

3.4. Design Solution

The design was based on a simple idea of horizontal louvres for passive shading. This idea was developed by adding various complexities to the design to create an interesting shape in the form of a wood grain. We added depth to the elements by using equations to create a wavy surface. Small openings and additional curvatures were introduced to the elements to simulate the irregularities in nature. As we built the computational model, it was also easy to change the design later if it came to any changes to the building.

Through the development and research of design alternatives, we were now ready to find a final design and integrate it into the building. As we had several design goals, there was not just one, but multiple possible solutions. Designing is also subjective, either from the designer's point of view or from the investor's point of view. It was, therefore, up to us, the designers, to evaluate different options and choose one that, in our expert opinion, best met the design goals.

To find the final design parameters for the shape to integrate it into the building, we used the knowledge gained from the design evaluation process. We found out what values of the input parameters would give us optimal results, and so we had a threshold above which we could adjust to see which overall design solution would fit us better. When searching for the design solution, we also had to consider the visual appearance of the building. Once we determined the values of the input variables, additional details were added to complete the final design, which was then integrated into the building as shown in Figures 13 and 14. In the end, the final design was not one of the Pareto-optimal solutions, but it was the one that we decided would best meet all design goals, including the appearance of the facade. This solution was still close to the pareto front, and we can say that it is an efficient solution of the overall problem.

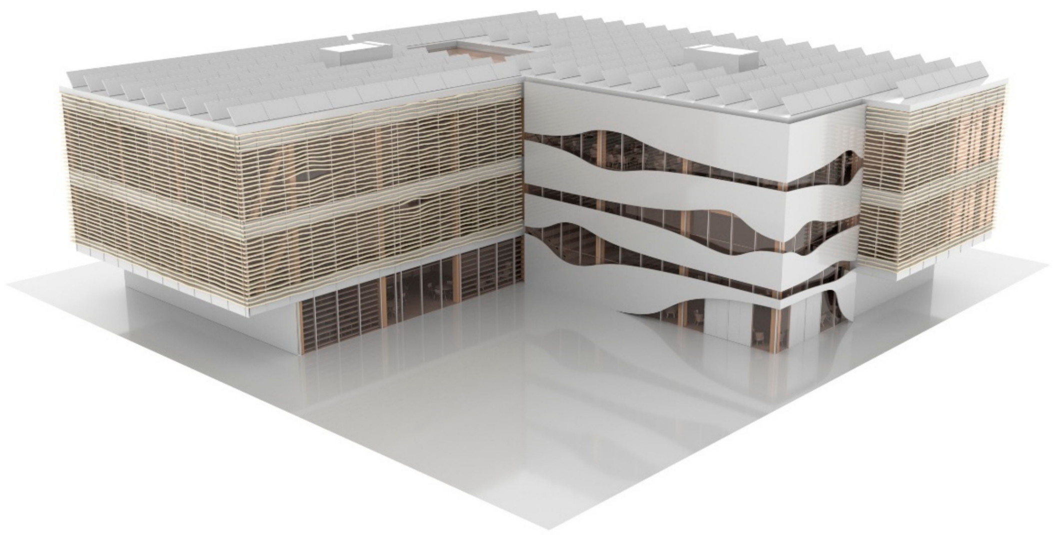


Figure 13. Render of the shading elements integrated into the building design.

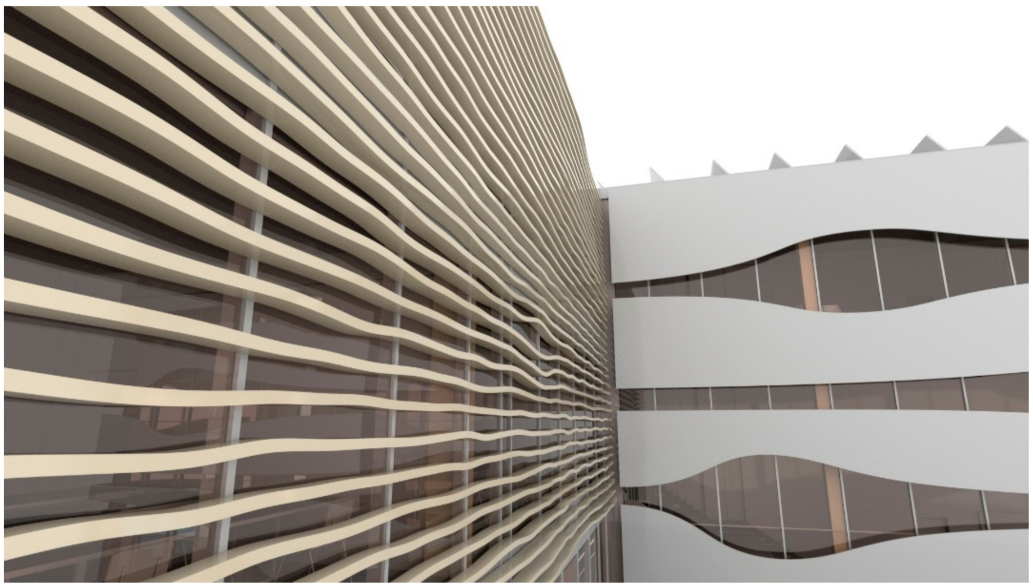


Figure 14. Close-up render of the integrated design solution.

When we first approached this problem without the generative design process, it was difficult to consider all design goals at once. We tried a few different solutions and picked one that we thought had a good appearance and would perform well for shading in the summer, but we did not know if it was optimal or if it could be improved upon in other design goals. It was difficult to compare different design solutions because we had no reference values to see how our design performed, and it would be too time-consuming to manually check all the possibilities. When we compared the solutions before and after using generative design, we found that the total material required for the design was reduced by 307%, while the shading was still efficient. In the initial design, we placed too much emphasis on the overall appearance of the façade and summer efficiency, and less on the other goals. With the generative design process, we were able to clearly see the design behaviour for all design goals and chose one that fit all of them according to our own assessment.

4. Discussion

Generative design, as an alternative approach to traditional methods, changes the perspective from a solution-oriented to a problem-oriented methodology. In this paradigm, more time is spent on iteratively improving a problem definition in the form of a computational model than on finding a specific solution. However, to understand how the model behaves, the solution space must be studied, which requires the generation of multiple design alternatives. Although there is overlap between generative design and optimisation methodologies, it is not in itself intended to solve a strict search problem, but the same tools and methods can be utilised to inform about the design. In this process, the problem definition is constantly changing until one is found that satisfies all of the designer's objective and subjective criteria. Once the problem is adequately defined, it can be considered solved [2].

The generative design methodology can be seen as beneficial for application to new or unique design problems where a general solution is not necessarily known. This can lead to a more informed solution that can also efficiently meet all the objectives of the design problem. In our case study, we approached a new problem that we were not familiar with. Through the generative process, we were able to learn about the behaviour of the problem and select a solution that was much more efficient in meeting all of the design criteria compared to the solution found in the traditional way. The drawback of such a process is the time that must be spent to define a computational model, time that is sometimes not available in conventional projects.

This process raises the question of whether generative design replaces the human component in design. The current answer is no. The designer is still responsible for two of the three parts of the generative design framework: first, building the computational model, and second, evaluating the results and selecting the best fitting solution. To eliminate the human component, the artificial system would need to be creative first to develop a computational model and second to have experience and critical thinking to evaluate the solution and either adjust the computational model or select a final solution to the design problem. Recent research has shown that the computational model can be replaced to some extent by machine learning image generators. However, this is mostly limited to existing and well-defined problems where data can be used to train the machine learning models [15,35,36].

Currently, we can say that the generative design combines a designer with an intelligent tool that complements the designer in the weaker areas: analysis of large amounts of data, analysis of repetitive tasks, and automation. We also know that it is almost impossible to find the perfect concept at the beginning; we have to iterate to find the best solution, which is especially difficult when we need to meet multiple design goals. We can define the problem and let the generator come up with various design alternatives, from which we can then either select a final one or continue to further develop our design.

5. Conclusions

In this paper, we presented the framework and methodology of generative design, which was tested and validated on the design of an efficient and aesthetic shading system for a building façade. Using a generative design framework, we created the computational model of the shading elements, generated different design alternatives, and found a solution that increased the shading efficiency and reduced the overall material requirements compared to the initial design.

This process is general and can be applied to any problem that can be described as a computational model. It can be used at various design stages to come up with a conceptual design or a final solution, or it can be used throughout the whole process by breaking the larger complex problems into smaller parts and solving them one at a time.

Here, the time spent on creating and analysing a single solution is instead used to describe a real-world problem with its objectives and constraints as a set of rules and equations, and we used them to automate the search for multiple design iterations at once by utilising the solver algorithms, which can save us a lot of time. By analysing these solutions, the designer can better understand the problem and find the efficient design faster. The main advantages of generative design compared to the traditional design process, which were found in the course of the work, are:

- This process can produce results that we would never think of or that are too complex to create manually;
- We can better understand the design problem by evaluating and analysing a large number of alternatives;
- Instead of finding a single solution, we are presented with several possible design solutions from which we can choose one that suits us best;
- Using the computational model and generative solver, we can save time by automating creation and analysis of a variety of design alternatives.

The result of such a design process is not only a final solution to the problem, but a computational model that was iteratively developed and led to the solution. It is in this model that knowledge derived from both the designer as well as from the algorithm that generates and tests the solutions is embedded. Further studies should investigate how this model could be used as a form of knowledge representation and applied against other design problems.

Author Contributions: Conceptualization, L.G. and M.D.; methodology, L.G., R.K. and Ž.T.; investigation, L.G.; resources, R.K. and Ž.T.; data curation, L.G.; writing—original draft preparation, L.G.; writing—review and editing, R.K., M.D. and Ž.T.; visualization, L.G.; supervision, M.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partially supported by the Slovenian Research Agency under the Young Researcher funding program and research programme E-construction (E-Gradbeništvo: P2-0210).

Data Availability Statement: The computational model and the data from the case study are available upon request from the corresponding author.

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

References

1. Renner, G.; Ekart, A. Genetic algorithms in computer aided design. *Comput. Aided Des.* **2003**, *35*, 709–726. [CrossRef]
2. Janssen, P.; Frazer, J.; Ming-Xi, T. Evolutionary Design Systems and Generative Processes. *Appl. Intell.* **2002**, *16*, 119–128. [CrossRef]
3. Frazer, J. Creative Design and the Generative Evolutionary Paradigm. In *Creative Evolutionary Systems*; Morgan Kaufmann: San Francisco, CA, USA, 2002; pp. 253–274. [CrossRef]
4. Caldas, L.G.; Nordford, L.K. A design optimization tool based on a genetic algorithm. *Autom. Constr.* **2002**, *11*, 173–184. [CrossRef]
5. Caldas, L. Generation of energy-efficient architecture solutions applying GENE_ARCH: An evolution-based generative design system. *Adv. Eng. Inform.* **2008**, *22*, 59–70. [CrossRef]
6. Shea, K.; Aish, R.; Gourtovaia, M. Towards integrated performance-driven generative design tools. *Autom. Constr.* **2005**, *14*, 253–264. [CrossRef]
7. Krish, S. A practical generative design method. *Comput. Aided Des.* **2010**, *43*, 88–100. [CrossRef]
8. Singh, V.; Gu, N. Towards an integrated generative design framework. *Des. Stud.* **2012**, *33*, 185–207. [CrossRef]
9. Johan, R.; Chernyavsky, M.; Fabbri, A.; Gardner, N.; Hank Haeusler, M.; Zavikeas, Y. Building intelligence through generative design. In Proceedings of the 24th International Conference of the Association for Computer-Aided Architectural Design Research in Asia (CAADRIA), Wellington, New Zealand, 15–18 April 2018.
10. Hofmeyer, H.; Davila Delgado, J.M. Automated design studies: Topology versus One-Step Evolutionary Structural Optimisation. *Adv. Eng. Inform.* **2013**, *27*, 427–443. [CrossRef]
11. Zawidzki, M.; Szklarski, J. Multi-objective optimization of the floor plan of a single story family house considering position and orientation. *Adv. Eng. Softw.* **2020**, *141*, 16. [CrossRef]
12. Hands-on with Project Rediscover: Generatively Designing the Autodesk Toronto Office. Available online: https://www.autodesk.com/autodesk-university/article/Hands-Project-Rediscover-Generatively-Designing-Autodesk-Toronto-Office-2020 (accessed on 15 July 2021).

13. Khan, S.; Awan, M.J. A generative design technique for exploring shape variations. *Adv. Eng. Inform.* **2018**, *38*, 712–724. [CrossRef]
14. Lohan, D.J.; Dede, E.M.; Allison, J.T. Topology for heat conduction using generative design algorithms. *Struct. Multidiscipl. Optim.* **2016**, *55*, 1063–1077. [CrossRef]
15. Yoo, S.; Lee, S.; Kim, S.; Hwang, K.H.; Park, J.H.; Kang, N. Integrating deep learning into CAD/CAE system: Generative design and evaluation of 3D conceptual wheel. *Struct. Multidiscipl. Optim.* **2021**, *64*, 2725–2747. [CrossRef]
16. Stasiuk, D. Design Modelling Terminology. Available online: https://archinate.files.wordpress.com/2018/06/dstasiuk-design-modeling-terminology1.pdf (accessed on 15 May 2020).
17. Stewart, R.H.; Palmer, T.S.; DuPont, B. A survey of multi-objective optimization methods and their applications for nuclear scientists and engineers. *Prog. Nucl. Energy* **2021**, *138*, 103830. [CrossRef]
18. Eckart, Z.; Lothar, T. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Trans. Evol. Comput.* **1999**, *3*, 257–271. [CrossRef]
19. Sun, Y.; Li, H.; Shabaz, M.; Sharma, A. Research on building truss design based on particle swarm intelligence optimization algorithm. *Int. J. Syst. Assur. Eng. Manag.* **2021**, *13*, 38–48. [CrossRef]
20. Lou, H.; Xiao, Z.; Wan, Y.; Quan, G.; Jin, F.; Gao, B.; Lu, H. Size optimization design of members for shear wall high-rise buildings. *J. Build. Eng.* **2022**, *61*, 105292. [CrossRef]
21. Li, B.; Tang, W.; Ding, S.; Hong, J. A generative design method for structural topology optimization via transformable triangular mesh (TTM) algorithm. *Struct. Multidiscipl. Optim.* **2020**, *62*, 1159–1183. [CrossRef]
22. Kwok, T.H. Improving the diversity of topology-optimized designs by swarm intelligence. *Struct. Multidiscipl. Optim.* **2022**, *65*, 20. [CrossRef]
23. Saremi, S.; Mirjalili, S.; Lewis, A. Grasshopper Optimisation Algorithm: Theory and application. *Adv. Eng. Softw.* **2017**, *105*, 30–47. [CrossRef]
24. Bentley, P.J. An introduction to evolutionary design by computers. In *Evolutionary Design by Computers*; Bentley, P.J., Ed.; Morgan Kaufman: San Francisco, CA, USA, 1999; pp. 1–73.
25. Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **2002**, *6*, 182–197. [CrossRef]
26. Konak, A.; Coit, D.W.; Smith, A. Multi-objective optimization using genetic algorithms: A tutorial. *Reliab. Eng. Syst. Saf.* **2006**, *91*, 992–1007. [CrossRef]
27. Akadiri, P.O.; Chinyio, E.A.; Olomolaiye, P.O. Design of A Sustainable Building: A Conceptual Framework for Implementing Sustainability in the Building Sector. *Buildings* **2012**, *2*, 126–152. [CrossRef]
28. Oropeza-Perez, I. The Influence of an Integrated Driving on the Performance of Different Passive Heating and Cooling Methods for Buildings. *Buildings* **2019**, *9*, 224. [CrossRef]
29. Grobman, Y.J.; Austern, G.; Hatiel, Y.; Capeluto, I.G. Evaluating the Influence of Varied External Shading Elements on Internal Daylight Illuminances. *Buildings* **2020**, *10*, 22. [CrossRef]
30. Mangkuto, R.A.; Koerniawan, M.D.; Apriliyanthi, S.R.; Lubis, I.H.; Atthaillah, A.; Hensen, J.L.M.; Paramita, B. Design Optimisation of Fixed and Adaptive Shading Devices on Four Façade Orientations of a High-Rise Office Building in the Tropics. *Buildings* **2022**, *12*, 25. [CrossRef]
31. What is Dynamo. Available online: https://primer.dynamobim.org/01_Introduction/1-2_what_is_dynamo.html (accessed on 20 April 2020).
32. Generative Design for Revit and Dynamo. Available online: https://www.generativedesign.org/01-introduction/01-05_gd-for-revit (accessed on 7 May 2020).
33. Generative Design in Revit Now Available. Available online: https://blogs.autodesk.com/revit/2020/04/08/generative-design-in-revit/ (accessed on 13 May 2020).
34. Nodes. Available online: https://primer.dynamobim.org/03_Anatomy-of-a-Dynamo-Definition/3-1_dynamo_nodes.html (accessed on 15 May 2020).
35. Oh, S.; Jung, Y.; Kim, S.; Kee, I.; Kang, N. Deep Generative Design: Integration of Topology Optimization and Generative Models. *J. Mech. Des.* **2019**, *141*, 13. [CrossRef]
36. Qian, C.; Kai Tan, R.; Ye, W. An adaptive artificial neural network-based generative design method for layout designs. *Int. J. Heat Mass Transf.* **2022**, *184*, 122313. [CrossRef]