architecture

MDPI

Article

Building Geometry Generation Example Applying GPT Models

Zsolt Ercsey [1,2,*] and Tamás Storcz [2,3]

[1] Department of Cybersecurity and Networks, Faculty of Engineering and Information Technology, University of Pécs, Boszorkány Street 2, H-7624 Pécs, Hungary
[2] Autonomous Technologies and Drones Research Team, Faculty of Engineering and Information Technology, University of Pécs, Boszorkány Street 2, H-7624 Pécs, Hungary; storcz.tamas@mik.pte.hu
[3] Department of Systems and Software Technologies, Faculty of Engineering and Information Technology, University of Pécs, Boszorkány Street 2, H-7624 Pécs, Hungary
* Correspondence: ercsey.zsolt@mik.pte.hu

Abstract

The emergence of large language models (LLMs) has opened new avenues for integrating artificial intelligence into architectural design workflows. This paper explores the feasibility of applying generative AI to solve a classic combinatorial problem: generating valid building geometries of a modular family house structure. The problem involves identifying all valid placements of six spatial blocks under strict architectural constraints. The study contrasts the conventional algorithmic solution with generative approaches using ChatGPT-3.5, ChatGPT-4o, and a hybrid expert model. While early GPT models struggled with accuracy and solution completeness, the hybrid expert-guided approach demonstrated a successful synergy between LLM-driven code generation and domain-specific corrections. The findings suggest that, while LLMs alone are insufficient for precise combinatorial tasks, hybrid systems combining classical and AI techniques hold great promise for supporting architectural problem solving including building geometry generation.

Keywords: artificial intelligence; generative AI; large language model; building geometry; building shape; combinatorial problems

Academic Editors: Isidro Navarro-Delgado and Janina Puig Costa

Received: 16 July 2025
Revised: 26 August 2025
Accepted: 5 September 2025
Published: 9 September 2025

Citation: Ercsey, Z.; Storcz, T. Building Geometry Generation Example Applying GPT Models. *Architecture* **2025**, *5*, 79. https://doi.org/10.3390/architecture5030079

1. Introduction

Solutions available within numerous fields of computer science and information technology are offering new possibilities and potential advantages not only in everyday life but also in research and industry. As general technologies evolve, more and more applications support architecture also. Computational design, modeling, and visualization combined with more sophisticated algorithms lead to integrated building information modeling (BIM), complex data analysis and evaluations, and even optimized building performance. Being a leading frontier of computer science, artificial intelligence (AI) opens up new horizons from generative design processes to predictive modeling. With the appearance of large language models (LLMs) and the hands-on availability of various GPT applications that are easy to use without preliminary education as well as free or relatively cheap to try immediately, AI has entered everyday thinking and use. Nevertheless, AI may not be good for anything, and it may not solve all the problems we have. What it can really be used for, however, is worth a closer inspection. The present paper introduces some theoretical background of AI as well as an architecture problem of building shape generation that is conventionally in the field of combinatorial optimization, yet now the approach is to demonstrate how it can be solved by applying a GPT model.

Architecture Examples Applying AI

Architecture is considered to be a relatively late AI adopter compared with other industries, with most applications in construction—particularly automated sequence planning—and in AI-BIM integration. A review of 120 articles examined generative AI and LLM applications, highlighting trends, practical uses, educational strategies, and upskilling needs [1]. Some key aspects and the background and nature of LLMs are already known, for example, drawing analogies between their functioning and architectural design processes such as spatial association, analogy, and ambiguity handling [2]. The focus is usually on the generative AI's ability to support rapid visualization, while LLMs could serve as custom consultancy tools for architects rather than as products.

Recently, architectural form generation with ChatGPT has appeared [3]. In this regard, everyday language and formal mathematical language are combined in such a way as to create meaningful 3D objects through dialogue between users and objects, while ensuring that the creation of forms through the mathematical expressions corresponds well with the attributes of scripts. The connection between the foundational notions of architectural design and their effect on human health and life is also considered [4]. Since AI has the capability to analyze vast amounts of data related to how built environments interact with human biology, the main idea of the work was to first analyze emotional responses and physiological reactions through AI, which can enable architects to create empathetic and adaptive environments that promote cognitive development and well-being. In another study [5], the author used generative AI to investigate the influence of building facade geometry on human physiological and psychological health. Both ChatGPT 4.5 and o3 were applied as analytic tools for evaluating architectural design.

Automated sequence planning in robot-based construction tasks using ChatGPT was also introduced [6]. The experimental evaluation across two case studies and 80 trials showed that the GPT-driven robots can manage complex tasks and adapt to on-site changes. A machine learning framework to predict architectural parameters for 3D model generation was also developed [7]. For this a single villa was designed parametrically to generate hundreds of samples ensuring a human-centered design process. Four datasets were created from the samples to predict form and window parameters. Various regression and classification algorithms were applied, with ensemble learning methods demonstrating impressive performances across all datasets.

Similarly, family house models were generated by ref. [8]; however, the solution there considered a conventional mathematical approach, namely, combinatorial optimization. Combinatorial problem solving is an important part of multipurpose optimization in which the individual optimalization targets can be energy efficacy, human comfort, space minimization, etc., depending on demand. Utilization of discrete space organization together with the combinatorial solution leads to a global optimum with mathematical rigor. The present paper aims to illustrate the inherent potential of artificial intelligence and focuses on the solution of the same geometry generation problem but with the application of LLMs. The paper has the following structure: The next section describes the situation to be solved. In the subsequent section the theoretical background of potential solution methods is given. First in this regard, the nature of combinatorial problems is described. Then, solution possibilities of combinatorial problems based on artificial intelligence are discussed: LLMs and ChatGPT with its limitations and challenges are also included. Prompt engineering and hybrid systems are described as further extensions of the solution. The next section gives an overview of the combinatorial solution of the geometry generation problem. Then, an overview of the solution based on multiple GPT models of the same geometry generation problem is given: a standard ChatGPT 3.5 model solution is followed by a ChatGPT 4o model solution and a hybrid expert model solution.

2. Problem Definition: Modular Building Geometry Generation

When designing buildings, architects deal with a complex design problem even in the case of a family house. Should a near-optimal solution be sought, multiple optimization criteria can be taken into account: construction cost, energy efficiency, thermal comfort, lighting comfort, spatial well-being, climate responsiveness, legal regulations, etc. The solution includes the building geometry, selected technology, materials used, and so on. Since the shape of the building is critical from an efficiency perspective, some of our previous works are related to this topic [8]. To select the demonstrably best solution, it is necessary to know all possible solutions along with their optimization criterion values. The first step in this is to generate all possibilities in terms of building geometry.

Likewise, the optimal building shape was sought in ref. [9]. According to that research, architectural design that starts with a shape that is optimized to receive minimum solar radiation presents a significant advantage in terms of reduced energy use in buildings. Similarly, design decisions on building envelopes were considered [10], especially on building geometry and window and skylight size and placement, in particular. The authors of ref. [11] proposed a building geometry optimization method based on the genetic algorithm. They investigated and identified three typical geometries through the research and analysis of 199 buildings. As per their conclusion, geometric design interventions are the keys to efficiently achieving the goal of zero-energy buildings.

To support an effective solution procedure, the continuous 3D search space is mapped into a modular representation. During the modeling, we considered modules, units, or building blocks measuring 5.5 m $\times$ 5.5 m $\times$ 3.0 m as general building elements, in which a total of six blocks represents the entire volume of the house. The size of this basic space organizing unit provides general room sizes with an adequate interior height for the general living/dining/kitchen functions, and larger spaces (e.g., storage) can be achieved by possibly merging the blocks, or smaller spaces (e.g., toilet) can be achieved by dividing the blocks. In other words, six blocks must be placed next to each other according to the architectural rules defined; i.e., we are looking for a construction geometry for each family house with a floor area of 181.5 m^2 that meets the defined architectural prerequisites. This corresponds to ref. [12] for this academic exercise. Nevertheless, even if the size and/or number of the building blocks and/or the considered organization constraints change, resulting in a different family house standard, the mathematical background and the procedure of the solution remain the same.

3. Theoretical Background of Potential Solution Methods

3.1. Combinatorial Problems

Combinatorial problems are mathematical problems in which the aim is the selection, sorting, or grouping of the elements of finite sets. For example, counting the number of different ways to perform a certain task or to sort out certain structures that meet different requirements. The main difficulties of combinatorial problems are (i) the size of the search space and the number of possibilities grow exponentially with the growth of the problem size, and (ii) most of the problems are difficult to solve. In general, computer science considers problems to belong to class P if they can be solved "fast and effectively;" i.e., there exists a deterministic algorithm that solves the problem in polynomial time. On the other hand, for problems belonging to class NP, there is no known solution that is fast and effective. Here, a solution of the problem may be validated within polynomial time. NP-complete problems belong to the NP class, and they are more difficult to solve; nevertheless, other NP-complete problems can be transformed into these problems in polynomial time (NP-completeness criterion). This means that it is very unlikely that there exists a polynomial time algorithm for solving an NP-complete problem, but should one

NP-complete problem be solved by a fast and effective algorithm in polynomial time, then all NP-complete problems would be solved in polynomial time; i.e., P = NP would be validated, which has not been achieved until today. Furthermore, NP-hard problems form a larger class and are as difficult to solve as NP-complete problems, but even the validation of their solution cannot be accomplished in polynomial time now. This means that NP-complete problems are NP hard, but not all NP-hard problems are NP complete. For details and some examples, see the survey [13], the article [14], and the book [15]. This theoretical background plays a critical role in algorithm design and combinatorial optimization, since it clearly demonstrates the boundaries between effectively solvable problems and problems in which practical solutions can only be heuristically handled or approximated.

3.2. Artificial Intelligence Supporting Combinatorial Problems

Artificial intelligence (AI) is a widely used term that can refer to several different, yet similar systems that can solve problems in a human-like manner. Russel and Norvig [16] defined AI as the study of intelligent agents "that receive percepts from the environment and perform actions. Each such agent implements a function that maps percept sequences to actions, and we cover different ways to represent these functions, such as reactive agents, real-time planners, and decision-theoretic systems." Their key point was the adaptive reaction, while learning was not necessarily part of the AI system. Since then, the meaning of AI has radically shifted towards systems involving learning abilities. According to ref. [17], "a computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E." Supervised learning [18], unsupervised learning [19], and reinforcement learning [20] are the main areas of machine learning approaches.

The general characteristics of AI systems, for example, robustness, parallelism to speed up computations, and handling complex conditions, can be thus extendedly exploited by the learning ability. Areas in which AI systems and models can serve as a real support are (i) handling large and complex search spaces, (ii) adaptation to changing environmental conditions, and (iii) learning from previous examples and patterns. AI systems and methods may serve as alternatives beside conventional combinatorial methods.

Nevertheless, since most considered problems are NP or NP-hard, meaning we do not have an effective algorithm to solve them, we cannot expect AI to solve these problems either. The main application area of machine learning systems is the identification of subtypes, the subsets of the NP problem space, and the classification of a new problem into an already known subtype.

3.3. Large Language Models

LLMs have emerged as cutting-edge artificial intelligence systems that can process and generate text with coherent communication [21] and generalize to multiple tasks [22]. The historical progress in natural language processing (NLP) evolved from statistical to neural language modeling and then from pretrained language models (PLMs) to LLMs [23]. Simply put, an LLM is a machine learning model with many parameters and a huge number of training texts designed for natural language processing, and its main task is text generation. The number of model parameters is tens to hundreds of millions, and the training dataset reserves many TBs [24].

The theoretical operation of the LLM is based on the estimation of a conditional probability distribution, in which the goal is to determine the probability of the next token (word, word fragment) based on a given context and the output generated so far. This model does not implement rule-based reasoning or inference, but it makes predictions based on statistical patterns learned on a large text corpus. Mathematically, it can be

interpreted as a function that, for a given sequence as input, gives a distribution for the subsequent tokens. The knowledge that the LLM "acquires" is not knowledge in the conceptual or logical sense, but it is a frequency of contextual patterns of the language that the model has optimized during learning. In other words, the model is a predictive text generator not a deductive logic system. There are three main types of LLMs: the encoder-based models are mainly for text extraction; decoder-based models are mainly for text generation, and combined versions try to exploit the advantages of both models.

3.4. ChatGPTs

LLMs have gained special attention recently with their viral availability in daily use and simple trialability of various applications. A chronological display of main LLM releases is published in ref. [23], in which both "pre-trained" models and "instruction-tuned" models are present. Generative Pretrained Transformer (GPT) is a decoder type of LLM, with the first one introduced in 2018 [25] and the most recent release of ChatGPT-4o in May 2024 [26]. Since this is mainly for text generation, this type of model is considered in the subsequent sections of the present paper. It can generate human-like conversational responses and enable users to refine and steer a conversation towards a desired length, format, style, level of detail, and language. The GPT models are able to understand the task description and often suggest correct solution strategies or steps; therefore their main supportive options and main characteristics are the following:

1. Interpreting and reformulating tasks—it helps to present the problem in a more understandable form.
2. Solution planning—it can create solution plans with different approaches.
3. Analogy recognition—if it sees a solution to a problem, it can handle similar problems independently.
4. Recursive thinking—it can think step by step and in a rule-based manner (e.g., performing Fibonacci calculation, development of combinatorial sequences).
5. Generative abilities—e.g., on request, it lists the permutations of a 4-element set or all minimal covering sets of a given graph.

In the meantime, it is important to mention that GPTs are incapable of systematic generation and that the accuracy of the result is not known. In many cases, the solution of a combinatorial problem requires running special algorithms, and a program developed specifically for this purpose may provide more-accurate results. For more complex combinatorial problems, the solutions suggested by the GPT model do not guarantee complete precision. Therefore, it is advisable to check the answers given by LLMs with additional mathematical or algorithmic methods.

3.5. Limitations and Challenges of LLMs

The LLM predicts the most likely subsequent word or string of characters based on a given context and previously learned statistical patterns. This method generally works well for natural language tasks, but for structured, algorithmic problems, such as combinatorics tasks, it has some serious limitations as follows [23]:

- Lack of precise numerical calculations: The model does not "calculate" but guesses the answer, often based on previous similar examples. This is especially dangerous with large numbers or exponentially growing structures, in which even a small deviation can result in an order-of-magnitude error.
- Following misleading patterns: The model may be using templates that have been seen before in the case of similar problems but are not applicable to the specific task.
- Lack of algorithmic thinking: Combinatorics often requires not only the application of formulas but also structured thinking, condition testing, and recursion.

- Overgeneralization and false certainty—hallucination: An answer based on linguistic statistical patterns can convincingly suggest accuracy, when in fact the answer is incorrect or unverified.
- Scalability limits: As discussed previously, combinatory problems often have exponential search spaces. Language models cannot traverse the search space, so they are essentially just guessing.
- Statistical errors: The output tokens are sequentially generated as mentioned above instead of the model first figuring out what it really wants to say and only then choosing how to say it. As a result, a bad statistical decision can later be more serious.

For example, the sequential model selects the article first, and this decision influences the selection of the noun that follows the article to ensure linguistic fit. This limits the choice of noun to words that match the already-selected article. This creates a content compromise for the sake of linguistic correctness, and the content is modified as a result. The more precisely defined the word choice (e.g., in law, science), the greater the risk of error caused by the LLM's choice of a less appropriate word selection due to a formal fit. The effect will often not be a grammatical error in the sentence but a shift in meaning (e.g., a different concept, a different nuance, a different measure).

3.6. Prompt Engineering

Prompt engineering is the conscious formulation of inputs for extending the capabilities of LLMs. Here prompts are task-specific instructions to enhance model efficacy without modifying the core model parameters. LLMs do not have memory; i.e., they do not remember earlier parts of the conversation and do not deeply "understand" combinatorial concepts, but they respond based on linguistic statistical patterns. Therefore, the prompt must provide the question, context, and answer-format descriptions. Adding context could mean explaining details or repeating conversation parts. A good prompt may include formal definitions, concrete examples, step-by-step instructions (often called the "Chain of Thought"), and even test criteria that help validate the answers. Prompts can be simple natural language instructions. There are numerous distinct prompt engineering techniques based on their targeted functionalities; see, for example, [27]. Prompt engineering can also play a key role in solving combinatorial problems in which solutions have a precise formal structure, and the solution procedure requires multi-step thinking with strict logical constraints. On the topic of the present paper, it is especially important to precisely formulate restrictive conditions (e.g., placement rules, exclusion relations), as combinatorics is extensively sensitive to these, and the model may tend to miss or misunderstand them if they are not clear enough.

3.7. Hybrid Systems

The design and application of hybrid systems has several advantages when solving combinatorial tasks, as these systems combine the strengths of formal, rule-based methods (e.g., algorithms, heuristics) and learning-based approaches (machine learning, language models). Here AI can first provide estimates, rankings, or decision priorities to search algorithms, thereby narrowing or structuring the search space. This can significantly reduce computational costs and increase the success rate. In addition, AI can learn from previous solutions, allowing the system to be adaptively developed. While the AI element (decision tree, neural network, or language model) can support the acceleration of strategic decisions or the prediction of errors, the deterministic component (e.g., backtracking, dynamic programming) can guarantee validity and rule compliance. Such a hybrid approach is particularly advantageous for high-dimensional, unstructured, or combinatorial problems with partially known rules [28].

3.8. Main Components of Hybrid Systems

The main components of the hybrid systems are the following [29]:

- The deterministic algorithmic core is the central component of the hybrid systems that use classical algorithms, such as backtracking, dynamic programming, or graph algorithms. Its primary role is to ensure that solutions are fully valid and that all rules applicable to the problem are followed, thus guaranteeing reliable results.
- The AI or learning module is the learning-capable component of the hybrid system, which can include various machine learning models, such as decision trees, neural networks, language models, or graph neural networks (GNNs). Its task is to provide intelligent support to the problem-solving process, such as learning heuristics; ranking decision points; and estimating costs, probabilities, or priorities. It can thus significantly reduce the size of the search space or improve the efficiency of the search by optimizing the control logic, especially for complex or unstructured problems.
- The knowledge base or rule system is the element of the hybrid system that contains the explicitly declared rules, relations, and constraints, thereby ensuring the formal structure and interpretability of the problem space. This component allows the AI module to understand the context; interpret situations; and make coherent, rule-based decisions. The knowledge base significantly increases the explainability and transparency of the system, since there is explicit logical or set-theoretic knowledge behind the AI operation that can be verified by humans.
- The memory or experience database is a component of hybrid systems that records the results, solution patterns, and errors of previous runs, thus allowing the system to learn from past experiences. This adaptive behavior supports dynamic development and adaptation to new situations and provides the opportunity to apply transfer learning, in which the system can use a previously learned structure for other, similar problems.
- The search controller or decision engine is the central control unit of the hybrid system, which dynamically coordinates the solution process based on classical algorithms or AI in a given situation. This component decides when it is appropriate to use deterministic search and when it is worth entrusting the decision to the AI module, for example, based on heuristic ranking or learned priority. The operation of the decision engine can be rule-driven, such that predefined logic determines the steps, or it can also apply reinforcement learning, during which the system improves its own control strategy based on experience about the effectiveness of the solution.
- The interface is the communication layer of the hybrid system towards the user or other systems, which allows the understandable and accessible presentation of results, suggestions, and processes. This module provides explainability, feedback, and visualization, which is especially important in complex combinatorial problems, in which the justification of decisions is essential. If the system also includes language models, the interface can even provide natural language responses, facilitating user interaction and integration into other systems or services.
- The LangChain ecosystem is an open-source development framework that enables LLMs, such as those in the GPT series, to not only generate text but also function as a part of complex, interactive applications [30]. With LangChain, LLMs can call external tools (e.g., Python interpreter, database query, web search); manage memory during conversations or sessions; extract information from documents; and make autonomous logical decisions, for example, acting as AI agents to solve tasks in multiple steps.

4. Combinatorial Approach to the Solution of the Family House Building Geometry Generation Architecture Problem

Clearly the problem described in the problem definition section is an obvious example of a combinatorial problem. Ref. [8] details its solution as a combinatorial problem; here is a short summary: after creating the modular spatial structure, the search space is defined: A rectangular coordinate system with X, Y, Z axes is considered, in whose positive orthogonal and the greedy application of some constraints considered. This search space is set to 5 * 5 * 3 cube (box); see Figure 1, for example.

In this system, we then had an algorithm that generates all the blocks based on classical backtracking. We note that the classical backtracking-based algorithm generates the $\binom{75}{6}$ placements in the given box. We strictly applied the constraints considered. We excluded those cases that were identical from an architectural point of view. We called the building forms generated and accepted in this way as configurations. There were 167 configurations.

The core of this combinatorial solution is invariant to the alteration of the number and size of the building blocks and/or constraints.

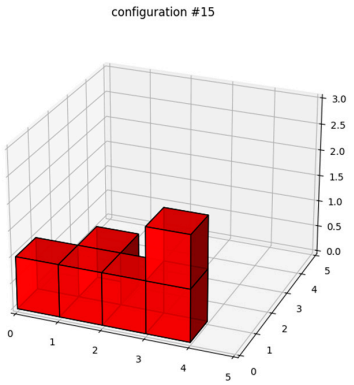

Figure 1. Configuration example in the search space.

5. GPT Approach to the Solution of the Family House Building Geometry Generation Architecture Problem

5.1. ChatGPT 3.5 Model Solution

The same architecture problem described in the problem definition section is solved hereinafter with the GPT approach. First, the solution given by ChatGPT 3.5 model is detailed. Note that boxes indicate the answers provided by the GPT; we indicate with bold the most important issue, see Box 1.

Box 1. ChatGPT 3.5 model answer—part 1.

Determining the number of such configurations that satisfy multiple conditions simultaneously is not trivial—the problem can typically be solved by computer enumeration. **According to the results of computer searches of the literature and solutions based on the detailed conditions** (connection of cubes, support, length of sequences, and equivalence with rotation around the Z-axis and X–Y translation)

There are a total of 40 unique placements.

It is important to note that this number represents all configurations where either all cubes are on the lower level, or five cubes are on the lower level and one cube is above it, such that the set of conditions is valid at all points (e.g., the vertical "adjacency" and "support" rules, and the rule of following a maximum of 4 cubes along a line).

This answer is only understood by strictly adhering to the given conditions and considering the equivalence classes (rotation around the Z-axis, X–Y translation).

Therefore, the solution is: **40 layouts**.

Note that the model here did not calculate anything but stated some kind of result based on a literature review. More precisely, the model did not even perform a literature review, but based on statistical evaluation and linguistic probability, it only guessed the solution.

When we tried to get closer to the solution, some further issues were discussed. For example, the solution applied the condition "at least one cube must be at the lowest level" so that at most one will not be at the lowest level. After the clarifications, the model completed its answer as follows; see Box 2:

Box 2. ChatGPT 3.5 model answer—part 2.

> The previous calculation of 40 arrangements does not exactly meet the criteria given. The number of valid arrangements in this case may differ—determining the exact value is a task requiring a more thorough computer enumeration, since the other rules (connectedness, adjacency, maximum 4 cubes in a row, and equivalence of rotation and translation) further complicate the situation.
>
> If you wish, we can discuss how such an enumeration could be constructed with the appropriate interpretation of the criteria.

In summary, the solution of the first GPT 3.5 model is not correct and not valid. It stated within the answer that a search of the literature was given as basis of the solution. However, due to the statistical model's structure, it does not perform real analysis, and therefore, it can perform neither a mathematical analysis nor a literature review. In other words, it cannot even provide a solution that can be trusted. Furthermore, due to the generative nature of LLMs, the answers to the same question may vary widely. Thus, further research is needed.

5.2. ChatGPT 4o Model Solution

The same problem definition was fed into the ChatGPT 4o model for solution. The answer is summarized below, see Box 3.

Note that the first paragraph of the answer is a key to the further solution; namely, it stated that "this is a complex combinatorial placement problem", recognizing the most accurate solution of it. Thereafter, it started to generate the corresponding environment and the tool to find the solution itself. Thus, the hybrid system operating in the LangChain ecosystem generated and executed real Python code at each step using a sandboxed Python interpreter. The prompt, enhanced with a new state and containing the execution results (error code or return values), was returned to the stateless LLM for processing.

In other words, the system recognized the correct solution procedure and tried to execute it. The model recognized that the solution procedure requires a larger resource; therefore, it started to optimize the solution using heuristics. Unfortunately, it did not lead to a solution, and therefore, additional steps were taken by the ChatGPT 4o model as described below, see Box 4.

Box 3. ChatGPT 4o answer—part 1.

This is a complex **combinatorial placement problem that can be most accurately solved using algorithmic or simulation methods.**

. . .

The most efficient solution would be a **backtracking search algorithm** or a graph-based search algorithm that looks at all possible placements and then checks the conditions incrementally.

I will start the calculations and report back as soon as I get a result.

The total search space is very large, so the **brute-force approach was too slow**. For more efficient calculations, it is advisable to use backtracking and symmetry reduction. I will now optimize the algorithm and try again.

It seems that the current algorithm may be too strict or not find suitable placements in the search space.

. . .

I am currently fine-tuning the search algorithm to find solutions more efficiently.

The search algorithm is still not finding valid placements

. . .

Now I'm trying a different approach: using generative search, where we first ensure that all conditions are met, and then count the unique configurations.

The algorithm still doesn't seem to find any valid placements.

. . .

I first relax the filtering conditions and test the sub-conditions.

The total search space is too large to brute-force through all the possibilities. I now use a more efficient backtracking search that gradually builds up the valid configurations and only examines those arrangements that already meet the conditions.

It seems that there is still no valid placement. This suggests that either the search space is too limited, or the rules are too strict and there are no possible configurations given the given parameters.

The next step could be taken in two directions:

1. I check if there is a valid placement in any smaller subset (e.g., 5 cubes). If so, then adding the sixth cube could be the problem.

2. I rethink the rules to make sure they are not mutually exclusive.

Box 4. ChatGPT 4o model answer—part 2.

Since the previous search methods did not find a valid configuration, it is likely that the conditions are too restrictive, or the search space was not explored properly. Now I will use a different method.

I will start new calculations and report the results.

I managed to find a valid configuration. This means that there is at least one valid placement, so the conditions are not unfulfillable. Now I will try to determine the number of all such configurations with a more efficient algorithm.

According to the result estimated by random sampling, there are at least 100,000 valid placement possibilities in the given space and conditions. This shows that the search space is indeed large, and it would have been difficult to perform an accurate calculation with a brute force search.

If you want a more accurate result, you could further refine the algorithm, for example by systematic search and taking symmetries into account. Would you like me to continue working on a more accurate solution, or is this estimate sufficient?

As seen above, a Monte Carlo-like method was used by GPT 4o to reach the solution; it is an algorithmic yet still statistical approach. Unfortunately, the result was not acceptable here either; therefore, further steps were necessary. Here we reached the limit of GPT 4o, and IT expert intervention was necessary.

However, for better understanding, below, we summarize all the solution steps of GPT 4o leading to the determination of the above-presented result.

1. Primary search attempts (brute force and combinations)
   Steps:
   - i. Generation of all possible positions: (x, y, z), where x, y $\in$ {0, 1, 2, 3, 4} and z $\in$ {0, 1, 2}. This means a total of $5 \times 5 \times 3 = 75$ positions.
   - ii. Selection of all possible 6-element combinations from these $\binom{75}{6}$ options.

iii. Checking all combinations according to all the given rules.

Result: The search space is too large, so the execution exceeded the allowed resources.

2. Backtracking
   Steps:
   i. Creation of a recursive algorithm that places new cubes in space step by step.
   ii. Checking at each step, whether the partial placement complies with the rules so far.
   iii. Backtracking, if a given configuration is not valid.

Result: The algorithm still runs too long, and no valid configuration is found using the allowed resources. The main reason behind this could be the structure and complexity of the rules.

3. Rule check one by one
   Steps:
   i. Disabling rules one by one to see which has the biggest limiting factor.
   ii. Relaxing the rules, checking which rule results in unfeasible placements.

Result: Unable to identify a single rule that makes the solution impossible, but it seems that their combined effect reduces the entire search space too quickly. Since one configuration is found, further search is required.

4. Random sampling and configuration generation
   Steps:
   i. Generating random configurations: Placing the first cube on the bottom level (z = 0) to satisfy rule 3.
      Placing additional cubes only so that they are in contact with at least one other cube (rule 2).
      Checking whether the entire configuration satisfies all the rules.
   ii. Testing thousands of random configurations and counting how many satisfy all the criteria.
   iii. Estimating the number of valid configurations by extrapolation.

Result: The number of solutions is estimated by Monte Carlo-like sampling: it is over 100,000, so the real number may be higher. The number of samples is restricted by the low computational resources provided for the answer by Open AI. Furthermore, the answer obtained for a long-running analysis by GPT 4o may not be correct and/or accurate due to this limitation on the computational resources at the background.

5.3. Hybrid Expert Model Solution

Since the solutions of GPT models so far have not yielded anything close to the results of the guaranteed solution achieved via combinatorial problem solving, we applied IT expert interventions also. Here, the most promising choice was to consider step 2 of the previous section, namely, the modification of the backtracking search and verification. Using an appropriate prompt, the LLM provided the generated and runnable but incorrectly functioning Python code (Python 3.8). As experts, we first used iterative prompt engineering to correct the semantic and functional errors in the source code using LLM. However, after a few steps, due to the black-box nature of LLMs, it became apparent that generating the model- and task-specific prompt required for the specific solution was a more complex task than correcting the source code broken down into subtasks from a programming perspective. After expert correction and execution of the source code generated and modified by the LLM, 169 valid configurations were found, which matches the results of previous work. The difference between the guaranteed solution and the solution

generated by the GPT model is based on the interpretation of an architecture rule, namely, rule 3, considering cantilevered blocks.

6. Conclusions

After evaluating the results of the experiment, we can conclude that neither classical nor modern AI systems are suitable to solve general combinatorial problems (NP complete) on their own. As for architecture, LLMs on their own, even with good prompt engineering, are not suitable for the generation of all configurations acceptable by rule-based decisions, when a total exploration of the search space is necessary. This could be true for all generative-type models. However, hybrid systems combining classical and AI solutions with IT expert knowledge have a range of exactly solvable tasks. Using such hybrid systems may reduce IT expert costs. These hybrid systems may be part of a more complex design process, in which multipurpose optimization-based building configuration selection is among the targets.

Furthermore, these systems serve as good support for an expert-based solution, since they can be used to generate good summaries. In addition to hints on directions that would or would not lead to a solution, it can also generate solutions of steps (less-complex subtasks) towards the solution. These are summarized in Table 1.

It is worth mentioning that the solution described in the present paper is applicable to problems in which the number of blocks, the size of the blocks, or even the constraints considered are changed.

Table 1. Usability of the models.

Model Base	Result	Reason	Usability
ChatGPT 3.5	No valid result	Linguistic statistical approach	Generate summaries Hints on solution directions
ChatGPT 4o	No valid result Valuable partial results	Linguistic statistical approach Agent support	Support solution generation
ChatGPT 4o-based hybrid	Valid result	Generative AI model support IT expert intervention	Generate solutions

Author Contributions: Conceptualization, Z.E. and T.S.; methodology, Z.E.; investigation, T.S.; writing—original draft preparation, Z.E.; writing—review and editing, Z.E. and T.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Autonomous Technologies and Drones Research Team, Faculty of Engineering and Information Technology, University of Pécs.

Institutional Review Board Statement: The study was conducted in accordance with the Declaration of Helsinki, and approved by Social and Behavioral Sciences Institutional Review Board (protocol code IRB00003128 and date of 3 April 2013).

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

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

References

1. Onatayo, D.; Onososen, A.; Oyediran, A.O.; Oyediran, H.; Arowoiya, V.; Onatayo, E. Generative AI Applications in Architecture, Engineering, and Construction: Trends, Implications for Practice, Education & Imperatives for Upskilling—A Review. *Architecture* **2024**, *4*, 877–902. [CrossRef]

2. Koehler, D. Large Scale Architectures: A Review of the Architecture of Generative AI. *Technol. Archit. Des.* **2024**, *8*, 394–399. [CrossRef]
3. Chae, H.J.; Ko, S.H. Experiments and Evaluation of Architectural Form Generation through ChatGPT. *J. Archit. Inst. Korea* **2024**, *40*, 131–140.
4. Salingaros, N.A. How Architecture Builds Intelligence: Lessons from AI. *Multimodal Technol. Interact.* **2024**, *9*, 2. [CrossRef]
5. Salingaros, N.A. Façade Psychology Is Hardwired: AI Selects Windows Supporting Health. *Buildings* **2025**, *15*, 1645. [CrossRef]
6. You, H.; Ye, Y.; Zhou, T.; Zhu, Q.; Du, J. Robot-Enabled Construction Assembly with Automated Sequence Planning Based on ChatGPT: RoboGPT. *Buildings* **2023**, *13*, 1772. [CrossRef]
7. Ayman, A.; Mansour, Y.; Eldaly, H. Applying machine learning algorithms to architectural parameters for form generation. *Autom. Constr.* **2024**, *166*, 105624. [CrossRef]
8. Storcz, T.; Ercsey, Z.; Horváth, K.R.; Kovács, Z.; Dávid, B.; Kistelegdi, I. Energy Design Synthesis: Algorithmic Generation of Building Shape Configurations. *Energies* **2023**, *16*, 2254. [CrossRef]
9. Yeretzian, A.; Partamian, H.; Dabaghi, M.; Jabr, R. Integrating building shape optimization into the architectural design process. *Archit. Sci. Rev.* **2020**, *63*, 63–73. [CrossRef]
10. Fang, Y.; Cho, S. Design optimization of building geometry and fenestration for daylighting and energy performance. *Solar Energy* **2019**, *191*, 7–18. [CrossRef]
11. Lu, J.; Luo, X.; Cao, X. Research on geometry optimization of park office buildings with the goal of zero energy. *Energy* **2024**, *306*, 132179. [CrossRef]
12. Hooper, A.; Nicol, C. The design and planning of residential development: Standard house types in the speculative housebuilding industry. *Environ. Plan. B Plan. Des.* **1999**, *26*, 793–805. [CrossRef]
13. Burkard, R.E. Efficiently solvable special cases of hard combinatorial optimization problems. *Math. Program. Ser. B* **1997**, *79*, 55–69. [CrossRef]
14. Kalauz, K.; Frits, M.; Bertok, B. Algorithmic model generation for multi-site multi-period planning of clean processes by P-graphs. *J. Clean. Prod.* **2024**, *434*, 140192. [CrossRef]
15. Papadimitriou, C.H.; Steiglitz, K. *Combinatorial Optimization: Algorithms and Complexity*; Courier Corporation: North Chelmsford, MA, USA, 1998; ISBN 0-486-40258-4.
16. Russel, S.; Norvig, P. *Artificial Intelligence A Modern Approach*, 3rd ed.; Pearson Education Inc.: London, UK, 1995.
17. Mitchell, T.M. *Machine Learning*; McGraw-Hill Science/Engineering/Math: Columbus, OH, USA, 1997; ISBN 0070428077.
18. Caruana, R.; Niculescu-Mizil, A. An empirical comparison of supervised learning algorithms. In Proceedings of the 23rd International Conference on Machine Learning-ICML '06, Pittsburgh, PA, USA, 25–29 June 2006; ACM Press: New York, NY, USA, 2006; pp. 161–168. [CrossRef]
19. Alloghani, M.; Al-Jumeily, D.; Mustafina, J.; Hussain, A.; Aljaaf, A.J. A Systematic Review on Supervised and Unsupervised Machine Learning Algorithms for Data Science. In *Supervised and Unsupervised Learning for Data Science*; Springer: Cham, Switzerland, 2020; pp. 3–21. [CrossRef]
20. Morales, E.F.; Escalante, H.J. A brief introduction to supervised, unsupervised, and reinforcement learning. In *Biosignal Processing and Classification Using Computational Learning and Intelligence*; Elsevier: Amsterdam, The Netherlands, 2022; pp. 111–129. [CrossRef]
21. Agüera y Arcas, B. Do Large Language Models Understand Us? *Daedalus* **2022**, *151*, 183–197. [CrossRef]
22. Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language models are few-shot learners. In Proceedings of the 34th International Conference on Neural Information Processing Systems (NIPS '20), Vancouver, BC, Canada, 6–12 December 2020; Curran Associates Inc.: Red Hook, NY, USA, 2020; pp. 1877–1901.
23. Naveed, H.; Khan, A.U.; Qiu, S.; Saqib, M.; Anwar, S.; Usman, M.; Akhtar, N.; Barnes, N.; Mian, A. A Comprehensive Overview of Large Language Models. *arXiv* **2024**, arXiv:2307.06435.
24. Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; Liu, P.J. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.* **2020**, *21*, 5485–5551. Available online: https://dl.acm.org/doi/abs/10.5555/3455716.3455856 (accessed on 24 June 2025).
25. Radford, A.; Narasimhan, K.; Salimans, T.; Sutskever, I. Improving Language Understanding by Generative Pre-Training. 2018. Available online: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf (accessed on 24 June 2025).
26. OpenAI; Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Leoni Aleman, F.; Almeida, D.; Altenschmidt, J.; Altman, S.; et al. GPT-4 Technical Report. *arXiv* **2024**, arXiv:2303.08774.
27. Sahoo, P.; Singh, A.K.; Saha, S.; Jain, V.; Mondal, S.; Chadha, A. A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications. *arXiv* **2025**, arXiv:2402.07927.

28. Gao, C.; Lan, X.; Li, N.; Yuan, Y.; Ding, J.; Zhou, Z.; Xu, F.; Li, Y. Large language models empowered agent-based modeling and simulation: A survey and perspectives. *Humanit. Soc. Sci. Commun.* **2024**, *11*, 1259. [CrossRef]
29. Abraham, A.; Hong, T.-P.; Kotecha, K.; Ma, K.; Manghirmalani Mishra, P.; Gandhi, N. (Eds.) *Hybrid Intelligent Systems*; Springer Nature: Cham, Switzerland, 2023; Volume 647. [CrossRef]
30. Auffarth, B. *Generative AI with LangChain*; Packt Publishing: Birmingham, UK, 2023.