POLITECNICO
MILANO 1863

DIPARTIMENTO DI
INGEGNERIA CIVILE E AMBIENTALE

Executive Summary of the Thesis

Truss Structure Optimization with Large Language Models

Laurea Magistrale in Civil Engineering - Ingegneria Civile

Author: Luigi Altieri

Advisor: Prof. Stefano Mariani

Co-advisor: Dr. Matteo Torzoni

Academic year: 2024-2025

1. Introduction

Artificial intelligence (AI) and machine learning have recently emerged as powerful tools for engineering applications, opening new perspectives in computational design. Among these approaches, transformer-based architectures and large language models (LLMs) have achieved remarkable success in modeling sequential data. Their core innovation lies in the attention mechanism, which allows the model to capture long-range dependencies and hierarchical relationships within data. Originally developed for natural language processing, these models operate on tokens by learning statistical patterns that are independent of semantic interpretation. Moreover, fine-tuning, a strategy in which a pretrained model is adapted to a specific task, has further strengthened their adaptability, enabling their application to domains beyond language [1].

In parallel, structural optimization has undergone a conceptual evolution. Rather than treating topology optimization purely as a global mathematical problem, recent research has reformulated structural synthesis as a sequential decision-making process [2, 3]. In this perspective, a structure is progressively generated from an initial seed configuration through a series of admissible modifications. So-called grammar rules regulate these transformations, ensuring that each intermediate configuration remains kinematically admissible.

The convergence of these research directions suggests a novel opportunity. If structural synthesis can be expressed as a sequence of admissible actions, and if LLMs are inherently designed to model sequences, then structural optimization can be reformulated as a token-generation problem. Within this framework, each design action can be encoded as a token, transforming the evolution of a truss structure into a textual sequence that can be expressed by a LLM.

Building upon this idea, the present work involves the use of a Generative Pretrained Transformer (GPT) properly fine-tuned for planar truss structure optimization. The progressive generation of a truss is represented as a text string. A customized loss function that accounts for structural performance is integrated into the learning process, allowing the model to be biased towards the generation of mechanically efficient structures.

Through this formulation, this thesis aims to bridge recent advances in generative AI and grammar-based structural synthesis, positioning LLMs as potential complementary tools for truss structure optimization.

2. Methodology

The planar truss optimization problem is formulated on a predefined grid and regulated through grammar rules that define admissible structural transformations. A dataset is constructed by randomly sampling sequences of admissible actions, generating truss configurations that comply with the grammar rules. Each generated structure is evaluated via finite element analysis to extract a structural performance indicator, which is used to weight the customized loss function. The sampled action sequences are encoded as text strings, so that the final dataset consists of structural configurations represented in text form and labeled with performance-based weights. A pretrained GPT-2 [4] is then fine-tuned on this weighted dataset. During inference, the trained model generates text action sequences, which explicitly guide truss structure generation.

2.1. Optimal Design Problem and Generative Grammar Rules

Starting from a mechanically admissible seed configuration, a truss topology is progressively constructed by activating grid nodes and connecting them through structural members. At each stage, nodes belonging to the current structure are considered active, while the remaining grid nodes are inactive.

The truss evolves through discrete actions. Each action is characterized by: (i) the selection of an existing structural member, (ii) the selection of an inactive node from the grid, and (iii) the choice of an operator that determines how the new node is connected.

Two operators are adopted. Operator $\mathcal{D}$ connects the new node while preserving the selected element, whereas operator $\mathcal{T}$ removes the selected element and reconnects the structure through the new node. Both operators generate triangular substructures, ensuring that static determinacy is preserved during the growth process. In this way, mechanical admissibility is embedded directly into the action space. A visual representation of the operators is shown in Fig. 1.

In addition to grammar-level constraints, geometric feasibility checks prevent bar intersections and degenerate configurations, ensuring that all intermediate states remain kinematically compatible.

Each generated truss is modeled as a pin-jointed linear elastic structure and analyzed under static equilibrium. The structural performance of each final configuration is evaluated through the maximum nodal displacement $U_{\max}$ under prescribed loads and boundary conditions.

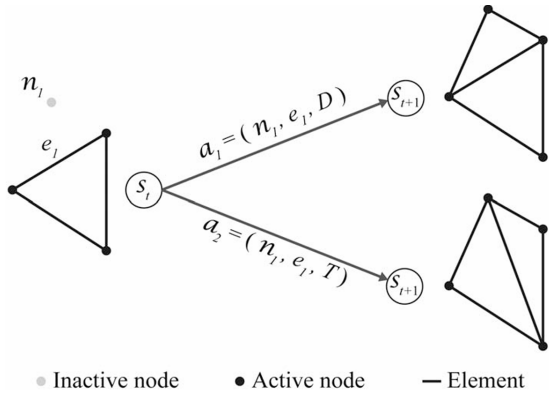

Figure 1: Visual representation of operators $\mathcal{D}$ and $\mathcal{T}$. The current structural state (left) evolves either through action a_1 (top right), obtained by applying operator $\mathcal{D}$, or through action a_2 (bottom right), obtained by applying operator $\mathcal{T}$, leading in both cases to a new configuration. In the two illustrated transitions, the selected truss member is e_1, and the inactive node introduced into the structure is n_1 (adapted from [2]).

2.2. Dataset Construction

To train the language model, a dataset of mechanically admissible truss topologies is constructed. For each seed configuration, a dedicated pipeline generates a large collection of distinct trusses by sampling admissible actions. Each structure is grown iteratively by randomly selecting an inactive node, an existing truss element, and an operator. Each candidate action is validated through grammar and geometric feasibility checks before being applied to update the configuration.

Once a valid sequence is completed, the resulting truss is evaluated via finite element analysis by solving the linear equilibrium system. The maximum nodal displacement $U_{\max}$ is extracted as a scalar performance indicator. The action sequence is then converted into a text representation of the form

$$T\langle id_{\text{task}}\rangle\langle k\rangle\left[element_i, node_i, operator_i;\right]_{i=1}^{k},$$

where:

- id_{task} denotes the seed configuration;
- k is the number of applied actions;
- $element_i$ represents the i-th selected element, identified by its end nodes;
- $node_i$ denotes the selected node at step i;
- $operator_i$ is the operator applied at step i, either $\mathcal{D}$ or $\mathcal{T}$.

Mechanical performance is incorporated into the learning process by assigning a scalar weight w to each dataset sample. After removing extreme outliers, the displacement values $U_{\max}$ are normalized by extracting the minimum and maximum values observed within the dataset, yielding a normalized performance measure $u_{\text{norm}} \in [0, 1]$. The weights are then defined as

$$w = \exp(-\alpha u_{\text{norm}}), \tag{1}$$

so that structurally efficient configurations receive larger weights, while less efficient ones contribute less to the learning process. The bounded nature of $w \in (0, 1]$ promotes stable optimization during training.

For each task, two weighted datasets are constructed using $\alpha = 6$ and $\alpha = 10$, corresponding to moderate and stronger performance emphasis, respectively. The rationale behind this dual weighting strategy will be discussed later. Duplicate configurations are discarded to ensure structural diversity.

2.3. GPT-2 Fine-Tuning and Tokenization

Once truss configurations are represented as text strings, they must be tokenized, i.e., converted into token sequences before being processed by the model. In this work, GPT-2 is adopted as the backbone architecture, selected for its adequate capacity to handle the considered tasks without requiring larger-scale models.

Tokenization is performed using the standard GPT-2 tokenizer based on Byte Pair Encoding. The tokenizer first operates at the byte level and then merges frequent symbol pairs to generate subword tokens.

Rather than training from scratch, a pretrained GPT-2 model is adopted and fully fine-tuned on the task-specific dataset. Pretraining equips the model with the ability to capture sequential dependencies between tokens, a capability preserved during fine-tuning.

The training strategy is staged: two epochs are first carried out on the dataset weighted with $\alpha = 6$, encouraging the model to learn the grammar rules governing admissible structural transformations. Subsequently, four additional epochs are performed on the dataset weighted with $\alpha = 10$, progressively biasing the model toward mechanically efficient configurations.

2.4. Customized Loss Function

Fine-tuning follows the standard autoregressive training paradigm: given a token sequence $\{x_t\}_{t=1}^{S}$ of length S, the model learns to predict the next token conditioned on the preceding ones.

For each token position t, the negative log-likelihood defines the token-level loss as

$$\ell_t = -\log p_\theta(x_t \mid x_{<t}). \tag{2}$$

These token-level contributions are averaged to obtain the sequence-level loss as

$$\mathcal{L}_{\text{GPT-2}} = \frac{1}{S-1} \sum_{t=2}^{S} \ell_t. \tag{3}$$

Up to this point, the training objective coincides with the standard GPT-2 loss. In conventional language modeling, this objective has a purely syntactic interpretation: all sequences contribute equally, and the goal is to reproduce admissible token patterns.

To bias generation toward mechanically efficient configurations, the sequence-level loss is reweighted using the performance-based weight assigned during dataset construction. The customized loss is therefore defined as

$$\mathcal{L}_{\text{final}} = w\, \mathcal{L}_{\text{GPT-2}}, \tag{4}$$

where w is the performance weight defined in Eq. 1. Through this modification, the grammatical objective of next-token prediction is augmented with a physical bias and implicitly embeds structural performance into the learned model parameters.

2.5. Inference

After fine-tuning, the model is used to generate new truss configurations in text form. Given a prompt containing the task identifier and the prescribed number of actions, together with a

temperature parameter T, it produces the continuation of the sequence token by token.
The temperature parameter T controls sampling by adjusting the sharpness of the next-token probability distribution. Low values of T lead to near-deterministic behavior, favoring the most probable tokens at each step. Higher values of T increase diversity by allowing less probable tokens to be sampled, thereby promoting exploration of alternative topologies.
Generation terminates when the closing bracket token `"]"` is produced, ensuring that the output corresponds to a complete encoded structure. For example, given the prompt `"T1<2>["` with $T = 0.01$, the model generates `"T1<2>[1-12,10,D;1-12,8,T]"`.
The generated text is then converted back into an explicit truss configuration through a reconstruction module, and $U_{\max}$ is evaluated via finite element analysis.

3. Results

The proposed framework is evaluated on six benchmark tasks adopted from [2], characterized by different grid sizes, seed configurations, and loading conditions. These tasks provide a heterogeneous testbed to assess the generative capabilities of the fine-tuned model.
For each task, ten distinct candidate structures are generated. The generation process is controlled via temperature T scaling, starting from low values to favor admissible and consistent designs, and progressively increasing to promote structural diversity. For each temperature level, multiple sequences are generated until ten distinct admissible candidates are obtained.
The best-performing configuration among the generated candidates is then selected and compared against the known global optimum. The comparison is given in terms of an objective ratio, defined as the percentage ratio between the maximum displacement of the generated structure and that of the global optimum.

3.1. Task Results

The results for each task are summarized in Table 1. The achieved objective ratios range between 82% and 100%, demonstrating that the model is capable of generating mechanically effective structures, including configurations that are novel with respect to the training dataset.

Table 1: Objective ratio, temperature, and novelty label of the selected configuration for each task.

Task	Obj. ratio (%)	T	Novel/Seen
1	100	0.01	Seen
2	99.04	0.01	Seen
3	99.72	0.10	Novel
4	96.44	0.10	Seen
5	90.49	0.10	Novel
6	82.68	0.10	Novel

As an example, Fig. 2 shows the generation process for Task 1. This task is characterized by a small grid size and a low number of node additions, resulting in reduced combinatorial complexity that facilitates the generation of the optimal configuration. Its relative simplicity makes it particularly suitable for clearly illustrating the correspondence between the generated text string and the associated actions.

3.2. Analysis of Results

Across the 60 generated configurations (10 per task), 52 structures were successfully reconstructed and satisfied all checks, while 8 were discarded during the reconstruction phase. This result suggests that the model has learned the grammar rules governing admissible truss synthesis. In addition, the model demonstrated the ability to generate novel configurations not present in the training dataset, particularly in tasks characterized by larger grids and a greater number of required actions. This confirms that the learned representation extends beyond mere memorization.
However, performance trends reveal that deterministic decoding ($T = 0.01$) does not always yield the best structural solution. In four out of six tasks, the best-performing configuration among the ten generated candidates was not obtained at the lowest temperature.
Further inspection of token probability distributions at critical decision steps provides additional insight. In some cases, the globally optimal action sequence appears among the highest-probability continuations, but it is not assigned the highest probability. At very low temperature values, the probability distribution becomes sharply concentrated on the most likely token, drastically reducing the likelihood of selecting

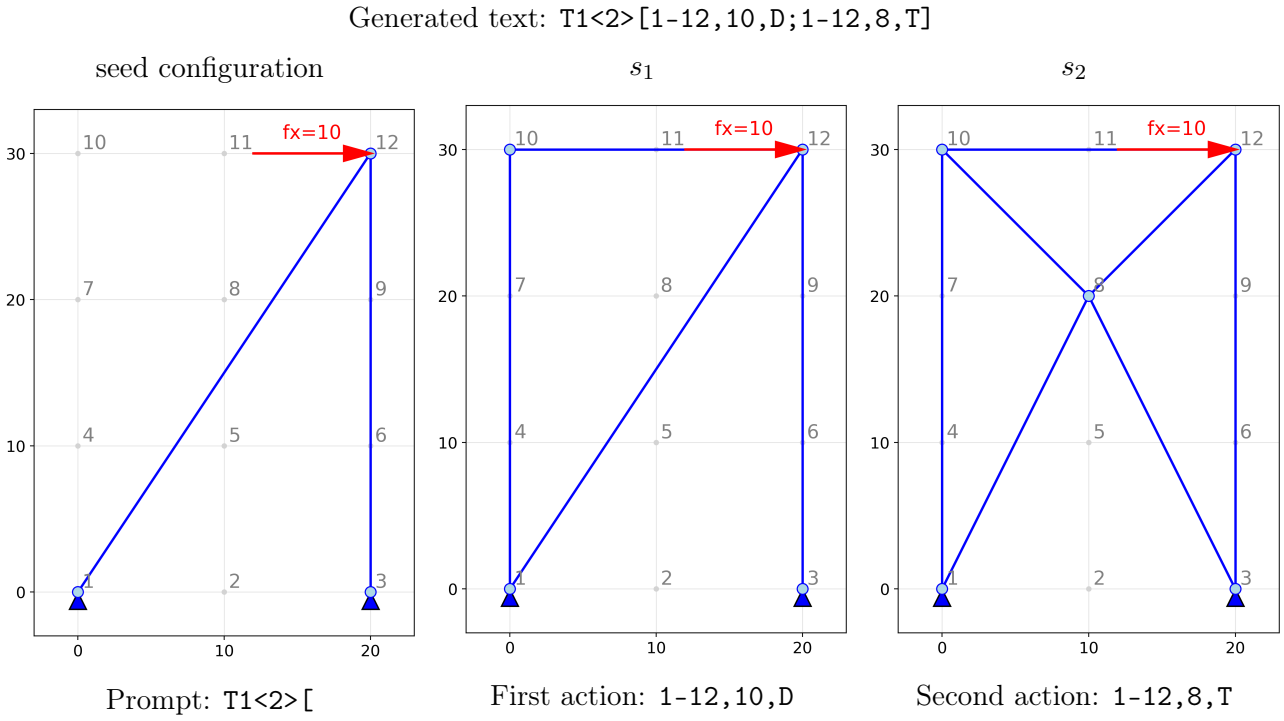

Figure 2: The prompt specifies the seed configuration and the required number of actions. The first action selects node 10 and connects it to the structure without removing any existing element, according to operator $\mathcal{D}$. The second action selects node 8, removes element 1-12, and later connects the node to the structure according to operator $\mathcal{T}$.

alternative tokens, even when they correspond to potentially optimal continuations.

Consequently, a trade-off emerges between adherence to grammar rules and exploration toward optimal solutions. Lower temperatures favor rule-consistent and admissible sequences, whereas moderately higher temperatures increase the likelihood of approaching the global optimal configuration.

4. Conclusions

Although LLMs were originally developed for language modeling, this thesis has demonstrated how they can be effectively adapted to truss structure optimization.

A key strength of the approach lies in the inference phase. Once fine-tuned, the model can generate multiple admissible candidate structures, allowing for subsequent selection of the best-performing configuration.

However, several limitations must be acknowledged. First, full fine-tuning remains computationally demanding, as it requires updating all model parameters. This motivates future investigation into parameter-efficient strategies.

Second, dataset construction represents a non-negligible effort. While this strategy guides the model toward efficient solutions, optimality is indirectly imposed through weighted supervision rather than autonomously learned. Consequently, the framework remains dependent on curated performance-labeled data.

A further critical aspect concerns the probabilistic nature of language models. The model does not explicitly learn the notion of optimization; instead, it samples from a learned probability distribution shaped by grammar rules and performance-weighted training. As shown in the analysis, the globally optimal configuration may not necessarily be selected, particularly when very low temperature values concentrate probability mass only on one token. This suggests that introducing a dynamic temperature strategy that adapts during each generation step could improve the likelihood of reaching the global optimum.

Despite these limitations, the primary objective of this thesis was to investigate whether an LLM-based framework could be applied to a problem of a physical nature. The results indicate that this framework has potential when

a problem can be reformulated as a structured sequence, allowing such models to be leveraged.

References

[1] Markus J. Buehler. MeLM, a generative pretrained language modeling framework that solves forward and inverse mechanics problems. *Journal of the Mechanics and Physics of Solids*, 181:105454, 2023.

[2] Gabriel Garayalde, Luca Rosafalco, Matteo Torzoni, and Alberto Corigliano. Mastering truss structure optimization with tree search. arXiv:2406.06145, 2024.

[3] Maximilian E. Ororbia and Gordon P. Warn. Design synthesis through a markov decision process and reinforcement learning framework. *Journal of Computing and Information Science in Engineering*, 22(2):021002, 2022.

[4] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners. Technical report, OpenAI, 2019.