POLITECNICO
MILANO 1863
SCHOOL OF CIVIL, ENVIRONMENTAL
AND LAND MANAGEMENT ENGINEERING

EXECUTIVE SUMMARY OF THE THESIS

Truss Structure Optimization via Hierarchical Tree Search

MASTER'S THESIS IN CIVIL ENGINEERING FOR RISK MITIGATION

Author: Arvan Sedighzadeh

Advisors: Prof. Alberto Corigliano and Dr. Matteo Torzoni

Academic Year: 2024-25

1 Introduction

Computational Design Synthesis (CDS) is regarded as a means to automate the design of structural systems. It provides the conceptual foundation for truss optimization methods that integrate decision-making algorithms with physics-based models to generate adaptive and efficient designs. Classical approaches, including the ground-structure method and metaheuristics such as genetic algorithms, particle swarm optimization, and ant colony optimization, have provided valuable results; however, their scalability is limited due to high computational complexity.

Machine learning offers alternative pathways. Through Reinforcement Learning (RL), truss design synthesis is framed as a Markov Decision Process (MDP). In RL, methods such as Q-learning and deep Q-learning have been applied to generate feasible layouts [1]. These methods demonstrate flexibility but require extensive training and struggle to propagate performance information across long decision sequences.

Monte Carlo Tree Search (MCTS) has emerged as a competitive approach [2]. When combined with grammar rules, it ensures structural feasibility. This integration yields significant computational savings and enables the synthesis of robust designs, including those developed in progressive construction settings [2].

Remaining challenges concern dense design domains and structural performance under evolving conditions, such as moving loads and localized damage. To address these challenges, a Hierarchical Monte Carlo Tree Search (H-MCTS) framework is introduced, in which the search is organized across multiple refinement stages that progressively concentrate exploration within promising regions. In addition, an adaptive morphing strategy is developed, operating in two phases: in the offline phase, H-MCTS precomputes optimized configurations under discrete load and damage scenarios, while in the online phase, near-optimal layouts are obtained by interpolating between precomputed solutions to accommodate intermediate load positions and local degradation. Together, the hierarchical search and adaptive morphing strategies enable efficient, scalable, and resilient truss designs within a coherent CDS framework.

2 Core Methodology

In this section, the methodology underpinning the truss optimization framework is presented. It is divided into three components: the mathematical formulation of the optimal design problem (Section 2.1), the representation of structural synthesis as an MDP (Section 2.2), and the application of grammar rules to guide the design process (Section 2.3).

2.1 Optimal Design Problem for Truss Structures. The truss optimization problem is defined as finding a geometry that minimizes a structural objective under static loading. In this study, the objective is the minimization of the maximum absolute displacement, a formulation analogous to compliance minimization in topology optimization [2]. The problem is first expressed within continuum elasticity, where the design domain is divided into I subdomains $\{\Omega_1^s, \dots, \Omega_I^s\}$, each representing a portion of the structure, and their union defines the complete design. The optimization problem can be written as:

$$\overline{\Omega} = \bigcup_{i=1}^{I} \overline{\Omega}_i^s = \arg \min_{\Omega = \cup_{i=1}^{I} \Omega_i^s} ||\mathbf{u}(\mathbf{x})||_\infty, \qquad \text{with } \mathbf{x} \in \Omega. \tag{1}$$

Here, $\mathbf{u}$ denotes the displacement field, $\mathbf{x}$ the spatial coordinates, and $||\mathbf{u}||_\infty$ the infinity norm of the displacement, defined as the maximum absolute value among its components.

Problem (1) is governed by the following constraints:

$$\begin{cases} \nabla \cdot \boldsymbol{\sigma} + \mathbf{b} = 0, & \text{equilibrium,} \\ \boldsymbol{\sigma} = \mathbf{E}\boldsymbol{\epsilon}, & \text{linear elastic constitutive law,} \\ \boldsymbol{\epsilon} = \dfrac{\nabla \mathbf{u} + \nabla \mathbf{u}^{\mathrm{T}}}{2}, & \text{linear kinematic compatibility,} \end{cases} \tag{2}$$

which satisfy the conditions of static elasticity. In these equations, $\boldsymbol{\sigma}$ denotes the stress field, $\boldsymbol{\epsilon}$ the strain field, $\mathbf{b}$ the vector of body forces, $\mathbf{E}$ the elasticity tensor, $(\nabla \cdot)$ the divergence operator, and (∇) the gradient operator.

The Boundary Conditions (BCs) are defined as:

$$\begin{cases} \mathbf{u} = \mathbf{u}_{\mathrm{g}}, & \text{on } \partial\Omega_{\mathrm{g}} = \bigcup_{i=1}^{I} \partial\Omega_{\mathrm{g}\,i}^{s}, \quad \text{Dirichlet BC,} \\ \boldsymbol{\sigma} \cdot \mathbf{n} = \mathbf{f}, & \text{on } \partial\Omega_{\mathrm{h}} = \bigcup_{i=1}^{I} \partial\Omega_{\mathrm{h}\,i}^{s}, \quad \text{Neumann BC.} \end{cases} \tag{3}$$

Here, $\mathbf{u} = \mathbf{u}_{\mathrm{g}}$ prescribes displacements on the Dirichlet boundary $\partial\Omega_{\mathrm{g}}$, while $\boldsymbol{\sigma} \cdot \mathbf{n} = \mathbf{f}$ applies surface tractions on the Neumann boundary $\partial\Omega_{\mathrm{h}}$, with $\mathbf{n}$ denoting the unit outward normal vector.

To apply this framework to planar trusses, Problem (2) is solved through a Finite Element (FE) discretization. Each subdomain Ω_i^s corresponds to a truss element, and the union operator models the connections at the joints. The optimization problem is then reformulated as:

$$\min_{\Omega = \cup_{i=1}^{I} \Omega_i^s} \quad \|\mathbf{U}(\Omega)\|_{\infty}, \qquad \text{with } \Omega_i^s \text{ a truss FE,} \tag{4}$$

$$\text{Subject to:} \quad \mathbf{K}\mathbf{U} = \mathbf{F}, \qquad \text{in } \Omega = \bigcup_{i=1}^{I} \Omega_i, \tag{5}$$

$$\mathbf{U} = \mathbf{U}_0, \qquad \text{on } \partial\Omega_{\mathrm{g}} = \bigcup_{i=1}^{I} \partial\Omega_{\mathrm{g\,i}}^{s}, \tag{6}$$

$$V \leq V^{\max}, \qquad \text{with } V = \sum_{i=1}^{I} A_i\, L_i. \tag{7}$$

In this formulation, $\mathbf{U}$ is the vector of nodal displacements, $\mathbf{K}$ the global stiffness matrix, and $\mathbf{F}$ the vector of applied forces. The condition $\mathbf{U} = \mathbf{U}_0$ enforces prescribed displacements on boundary nodes. The constraint $V \leq V^{\max}$ limits the total structural volume, where V is the sum of element volumes given by the product of cross-sectional area A_i and length L_i for each element.

2.2 Markov Decision Process for Sequential Decision-Making. The truss optimization task can be represented as a sequential decision problem under uncertainty. This process is formalized using an MDP.

It provides a mathematical framework to describe how an *agent* interacts with its *environment* through states, actions, and rewards**.** At each step, the agent selects an action from a defined set, which leads to a new state and produces a numerical reward. The objective is to learn a policy that maps each state to an action to maximize the expected cumulative reward over a planning horizon.

An MDP is formally defined as a four-tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$. Here, $\mathcal{S}$ is the state space of possible truss configurations, $\mathcal{A}$ is the set of admissible actions, $\mathcal{P}$ is the Markov transition model, and $\mathcal{R}$ is the reward function. In this study, the planning horizon is discretized into steps $t = 0, \ldots, T$, with states denoted by $s_t \in \mathcal{S}$ and actions by $a_t \in \mathcal{A}$. A reward $r_t \in \mathcal{R}$ is assigned to each state-action pair, and the goal is to identify the *optimal* policy π^* that maximizes the total expected reward. The reward takes a value of zero when the structural volume exceeds the allowable limit or when the displacement is greater than that of the initial configuration; otherwise, it takes a value between 0 and 1, computed as the relative reduction in maximum nodal displacement with respect to the initial value.

For optimal truss design, $\mathcal{S}$ is derived from a grid-based environment, where nodes define possible FE connections. Accordingly, $\mathcal{S}$ denotes all structural layouts, while $\mathcal{A}$ represents all possible modifications of a given layout. Both spaces grow rapidly, even within relatively small design domains. As a result, explicit modeling of the transition model $\mathcal{P}$ becomes infeasible. Instead, transitions and rewards are evaluated through simulated experience using the FE model in Equation (5), which enables algorithms such as Q-learning [1] and MCTS [2] to approximate the *action-value* function $Q^{\pi}(s, a)$ and guide the search.

2.3 Generative Grammar Rules for Truss Design Synthesis. Generative grammar rules define how truss configurations are incrementally constructed and modified within a discrete design space. Each modification complies with mechanical principles to preserve stability by requiring new members to compose only triangular forms.

The procedure begins with a seed configuration s_0, which is progressively updated through allowed actions. At each step, an action applied to the current state s_t produces a new configuration s_{t+1}. The process continues until a terminal state s_T is reached, for instance when the maximum permitted volume $V^{\max}$ is attained.

Allowed actions are defined through three operations: selecting an *inactive* node, choosing an existing truss element, and applying a legal operator ($\mathcal{D}$ or $\mathcal{T}$). The $\mathcal{D}$ operator introduces a new node and connects it without removal, while the $\mathcal{T}$ operator replaces the chosen element with new connections. In both cases, intersections or overlaps with existing members are prevented, which guarantees geometric admissibility and preserves structural feasibility. In this way, the design space is constrained, which reduces computational effort by focusing only on geometrically valid truss configurations.

3 Computational Framework

This section presents the computational details underlying the original contributions of this work. It is organized into four parts: Section 3.1 outlines standard MCTS for optimal truss design; Section 3.2 introduces tree policies based on Upper Confidence Bounds for Trees (UCT); Section 3.3 details the hierarchical extension H-MCTS; and Section 3.4 describes the

offline–online morphing strategy for moving load and damage scenarios.

3.1 Optimal Truss Design via Standard MCTS. The truss topology problem is modeled as a sequential decision process in which a search tree is incrementally grown. As shown in Figure 1, nodes (or states) represent truss configurations, while edges correspond to admissible design actions defined by the grammar. The terminal node carries the final objective value, and intermediate feasible states are also evaluated and stored during tree growth. Each episode proceeds through selection, expansion, simulation, and backpropagation.

During selection, the current tree is traversed from the root node s_0 according to a tree policy that favors promising branches while retaining exploration. If the reached node is nonterminal and not fully expanded, a new child node is added during expansion by applying an allowed, unexplored action. If expansion occurs, the simulation starts from the newly added node; otherwise, it begins from the selected leaf node. From this node, a randomized simulation (rollout) is executed until a terminal state is reached to obtain a scalar reward r_t. Finally, the reward is backpropagated along the visited path to update visit counts and action-values, thereby refining the tree policy. With an increasing number of episodes, these updates lead to progressively more accurate value estimates and a more decisive search strategy.

This procedure is well suited to progressive construction, where intermediate stages typically sustain only self-load, and the benefits of design choices become evident only at later operational phases. Such settings illustrate the delayed reward nature of the problem, which MCTS handles effectively by propagating information from terminal states back to earlier decisions. Moreover, no global function approximation is needed in this algorithm because the algorithm stores partial value statistics only where the search visits.

3.2 UCT-Based Tree Policies. The tree policy applied during selection is driven by an upper-confidence bound criterion. In this work, the classical UCT family is adopted with suitable adaptations. Two formulations are employed for comparative and parametric studies in the experiments:

$$\mathrm{UCT}_j^{\alpha} = (1-\alpha)\frac{v_j^{\Sigma}}{n_j} + \alpha\sqrt{\frac{2\log\sum_l n_l}{n_j}}, \tag{8}$$

where v_j^{Σ} is the accumulated reward obtained after choosing child node $j = 1, \ldots, J$; n_j is the visit count of that child; $\sum_l n_l$ is the visit count of its parent node; and $\alpha \in [0,1]$ is a weighting hyperparameter that balances the average-return term $\frac{v_j^{\Sigma}}{n_j}$ against the exploration term $\sqrt{\frac{2\log\sum_l n_l}{n_j}}$. This latter term encourages the selection of less-explored children.

By slightly adjusting Equation (8), another version of UCT, known as the MixMax modification [3], is obtained as follows:

$$\mathrm{UCT}_j^{\alpha\beta} = (1-\alpha)\left[\frac{(1-\beta)v_j^{\Sigma}}{n_j} + \beta v_j^{\mathrm{best}}\right] + \alpha\sqrt{\frac{2\log\sum_l n_l}{n_j}}, \tag{9}$$

where v_j^{best} is the best reward recorded after choosing child node j; and the hyperparameter $\beta \in [0,1]$ balances the exploitation emphasis between the average return and the best-seen return. In contrast to Equation (8), which is based solely on average returns, this formulation integrates peak rewards, mitigating bias toward high-average branches and enabling the discovery of hidden global optima among suboptimal solutions in expansive decision trees.

3.3 H-MCTS: A Scalable Search Strategy. Although grammar-guided MCTS performs well [1,2], its scalability degrades as higher node density increases the branching factor of the search tree. To restore tractability while preserving accuracy, a hierarchical search is proposed. The key idea is to stage the search over a sequence of grids with increasing resolution.

As illustrated in Figure 2, each k-th stage executes an independent MCTS on its grid G_k. From the terminal design $s_T^{(k)}$, which is obtained through the learned policy π_k, the set of

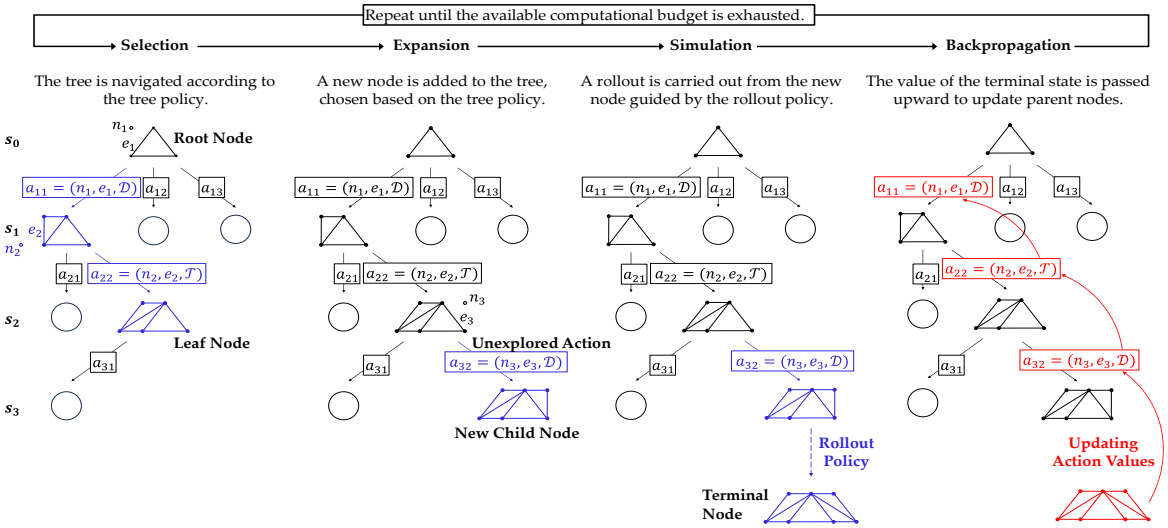

Figure 1: Schematic of the MDP-based truss design process solved by MCTS, illustrating the four algorithmic phases and the use of grammar rules to ensure the generation of admissible designs.

active nodes $n_h^{(k)}$ is extracted, and circular refinement regions $\mathcal{C}$ are defined around these nodes. At stage $k+1$, only the inactive nodes located within the refinement regions, together with the fixed support nodes (i.e., boundary nodes where displacements are constrained), are retained, and the process is repeated on the finer grid G_{k+1}. In this way, the search is progressively concentrated on promising regions of the design domain while avoiding the computational cost of exploring the entire dense grid.

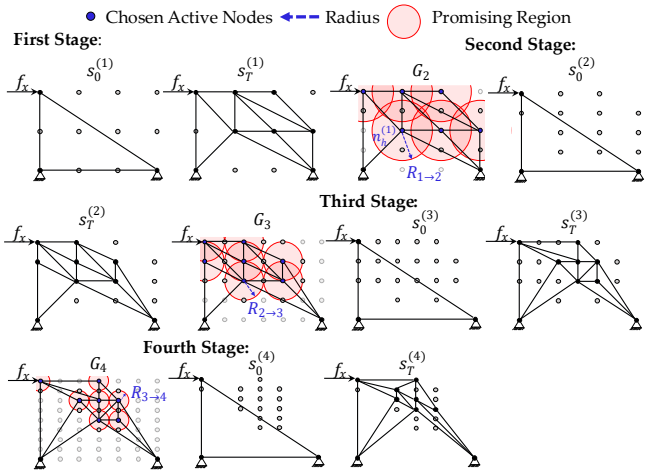

Figure 2: Multi-stage refinement in H-MCTS.

Across stages, the same physics-based numerical model, BCs, and volume constraint are enforced to ensure comparability of designs. The global best design, defined from the configuration with the lowest displacement objective across all stages, is tracked and updated whenever an improvement is obtained. The number of stages and refinement radii are problem-dependent and can be set to meet the available computational budget. In practice, using at most four stages has been found to balance design performance and computing time.

Figure 3 illustrates the full H-MCTS process, which strategically prunes the search tree branches at dense scales and channels simulations to the most valuable zones of the design space. As a result, the search progresses more efficiently and the final performance is improved, while the number of required FE evaluations is kept within practical limits.

3.4 Adaptive Truss Morphing. To handle load movement and localized degradation in the progressively built optimized configuration, an offline–online morphing strategy is employed. In the offline phase, H-MCTS is executed for a discrete set of load positions $p \in \{1, \dots, P\}$. The optimized layouts for each load position and damage case are stored in a database $\mathcal{DB}$.

A truss element $e_{(m=1,\dots,M)}$ is considered damaged (denoted e_d) if any of its mechanical or geometrical properties q_m decrease, including the Young's modulus E_m, density ρ_m, yield strength $\sigma_{y,m}$, cross-sectional area A_m, second moment of area I_m, and radius of gyration $r_{g,m}$. A damaged element becomes critical when it exceeds its yield limit $\sigma_{y,d}$ or the buckling threshold $\sigma_{\mathrm{cr},d}$. Accordingly, D is defined as the set of critical damaged elements, i.e., $D = \{e_d \in \Omega^s |\ e_d$ exceeds yield or buckling limits$\}$.

Figure 4 illustrates how the offline run for load applied at hinge $n_h^{(p=2)}$ explores a search region $\mathcal{E}(p)$, given by the union of the circular load region $\mathcal{L}(p)$ and the critical regions $\mathcal{D}_{\mathrm{crit}}(d)$ activated around the critical elements in D. The outcome of the offline phase is therefore a set of high-quality designs that integrate local reinforcements where needed.

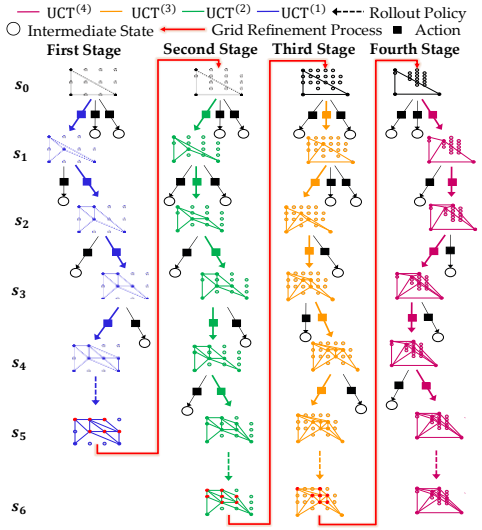

Figure 3: H-MCTS framework overview.

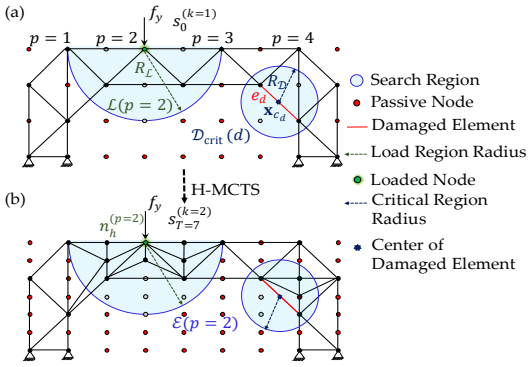

Figure 4: (a) Search region; (b) Optimized truss layout.

The algorithmic description of the offline procedure is provided in Algorithm 1. For each load position p, at each stage k, the candidate layout π_k is verified against yield thresholds for all elements and buckling thresholds for elements under compression. The global best is updated only if this verification is satisfied and the objective improves ($f_k < f^*(p)$). Upon completion of all stages, the best policy $\pi^*(p)$ and its objective $f^*(p)$ are stored in the database $\mathcal{DB}$.

In the online phase, as shown in Figure 5, an arbitrary load position is addressed without executing a full H-MCTS. Neighboring offline designs are retrieved from the database and merged under rule-based constraints that prevent member intersections and preserve the initial configuration (see Figure 5(c)). The combined layout is then mapped to a finer grid, and an approximation region $\mathcal{A}(p)$ is defined as the union of a small online load circle $\mathcal{L}^{\mathrm{on}}(p)$ and any activated online critical regions $\mathcal{D}_{\mathrm{crit}}{}^{\mathrm{on}}(p)$ (see Figure 5(d)). Nodes outside this region are treated as *passive* and removed from evaluation, while only members inside or intersecting the boundary are retained;

external members from restored layouts are discarded. Local FE connections are added where necessary to maintain structural integrity, linking active nodes to nearby ones to form triangular substructures, ensuring a mechanically feasible layout. Finally, grammar-compliant actions are applied within the approximation region under the constraints on volume $V^{\max}$ and member length $L^{\max}$, producing a near-optimal layout that is obtained instantaneously, without the cost of a full offline search.

Algorithm 1 Offline H-MCTS

Input: grids $\{G_k\}_{k=1}^{K}$, episodes $\{N_k\}_{k=1}^{K}$, refinement radii $\{R_{k\to k+1}\}$, grammar rules, $\mathrm{UCT}^{(k)}$ parameters (α_k, β_k), physics-based model, $V^{\max}$, load positions $p \in \{1, \dots, P\}$, load radius $R_{\mathcal{L}}$, critical radius $R_{\mathcal{D}}$.

Output: database $\mathcal{DB} = \{(\pi^*(p), f^*(p))\}_{p=1}^{P}$.

Initialize: set $\mathcal{DB} \leftarrow \emptyset$.
For each load position $p \in \{1, \dots, P\}$ **Do**
  Define load region: $\mathcal{L}(p)$.
  Define damage set $D = \{e_m : q_m < q_{m,0}\}$, where subscript 0 denotes undamaged values. A damaged element e_d becomes *critical* **If** it exceeds its yield strength $\sigma_{y,d}$ or buckling limit $\sigma_{cr,d}$; **Then** activate critical region $\mathcal{D}_{\mathrm{crit}}(d)$. **End If**
  Set search region: $\mathcal{E}(p) = \mathcal{L}(p) \cup \cup_{e_d \in D} \mathcal{D}_{\mathrm{crit}}(d)$.
  Initialize best: $\pi^* \leftarrow \emptyset$, $f^* \leftarrow +\infty$.
  For stage $k = 1, \dots, K$ of H-MCTS **Do**
  Restrict grid: **If** $k = 1$ **Then** use G_1; **Else** build G_k. **End If**
  Run MCTS on grid G_k:
    ▷ Apply **Selection**, **Expansion**, **Simulation**, and **Backpropagation** steps.
  Record stage result: obtain π_k and objective f_k.
  If $f_k < f^*$ and $(\forall e_m \in s_T^{(k)}: |\sigma_m| \le \sigma_{y,m}$, and $(\sigma_m \ge 0$ or $|\sigma_m| \le \sigma_{cr,m}))$ **Then**
    update (π^*, f^*).
  End If
  End For
  Store result: append $(\pi^*(p), f^*(p))$ to $\mathcal{DB}$.
End For
Return $\mathcal{DB}$.

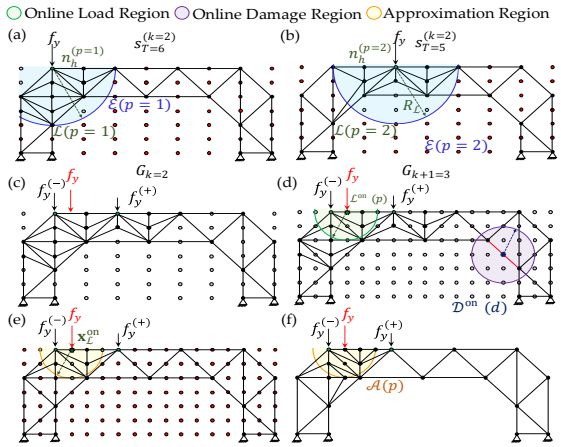

Figure 5: Online damage region $\mathcal{D}^{\mathrm{on}}(d)$ not activated. (a–b) Retrieval of offline layouts; (c) Merging under rule-based constraints; (d) Grid refinement; (e) Approximation region; (f) Near-optimal truss under moving load and damage.

4 Results

This section reports the obtained computational results: Section 4.1 compares the two UCT-based selection strategies, Section 4.2 evaluates the proposed H-MCTS framework, and Sections 4.3 and 4.4 examine the same bridge-like truss, first through progressive construction with load-induced morphing, and then through structural adaptation under damage. All numerical experiments were conducted in `Python` using the `Spyder` development environment on a personal computer with an `Intel® Core™ i7-1165G7 CPU @ 2.80 GHz` and `16 GB RAM`.

4.1 Comparative Performance of Tree Policies. Each truss element has Young's modulus $E = 10^4$ and cross-sectional area $A = 1$, while static loads of magnitude $f_x = f_y = 10$ are applied and self-weight is neglected. However, in Case Study A, which is directly adopted from [2], only a vertical load f_y is considered on a 5×5 coarse grid. In this case, two selection policies, over five training runs, are compared: the α-only UCT_j^{α} (8) and the MixMax $\mathrm{UCT}_j^{\alpha\beta}$ (9). The planning horizon is $T = 4$, and the volume constraint is $V^{\max} = 480$. All parameters are considered dimensionless, following [1] and [2].

The MCTS result with $N = 800$ (the number of episodes), $\alpha = 0.1$, and $\beta = 0.75$, shown in Figure 6, presents the sequence of structural layouts together with their displacement objectives $\|\mathbf{U}(\Omega)\|_\infty$ and structural volume V. Exhaustive search identified the global optimum among 1,433,800 possible configurations. MCTS, with MixMax policy, successfully synthesized that optimal solution.

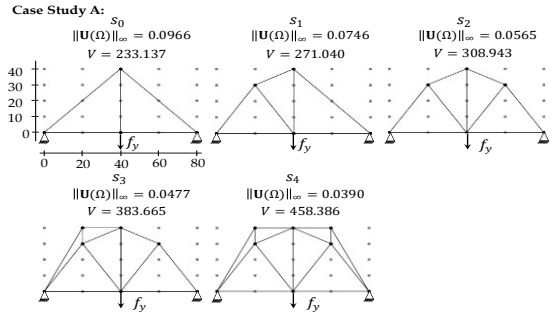

Figure 6: Sequence of layouts generated by the MCTS.

The displacement trends in Figure 7 show that the MixMax strategy converged to the optimum within about 400 episodes, while the α-only policy did not, even after extended training relative to Reference [2].

Unlike [2], where FE evaluations were performed only at terminal states, here evaluations are performed at every feasible state, providing stronger feedback, though at the expense of a higher number of FE evaluations. Figure 7 further confirms that the MixMax strategy reached the global optimum with significantly lower computational cost, requiring on average 780 FE evaluations and 23 seconds, compared to 19,570 evaluations and 320 seconds for the α-only policy.

Overall, the MixMax policy outperformed the α-only policy, offering faster convergence, lower computational cost, and identification of optimal solutions.

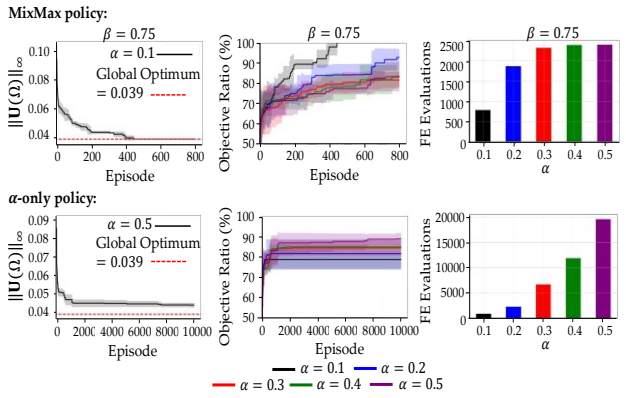

Figure 7: Results of MCTS for two selection strategies.

4.2 Efficiency and Solution Quality of H-MCTS. The performance of the hierarchical strategy is investigated for Case Study E [2]. The seed configuration and the final optimized trusses at each stage are presented in Figure 8.

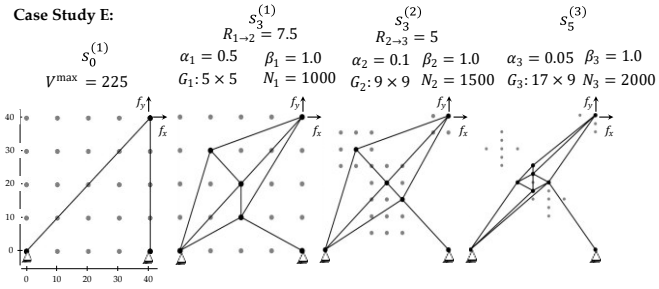

Figure 8: Seed and optimized layouts across three stages, each illustrated by its principal parameters.

Figure 9(a) shows that, as in other case studies, maximum displacements decreased steadily with each refinement stage. Figure 9(b) highlights that, unlike the general trend of rising costs at higher stages, the computational effort was reduced in this case. This improvement resulted from retaining only a small number of inactive nodes after refinement, which concentrated the search on compact promising regions.

In all experiments, the simulation phase of MCTS was the most time-consuming part, since every new state required repeated geometric feasibility checks [2].

In Case Study B, H-MCTS, as shown in Figure 10, recorded superior performance compared with standard MCTS. It achieved a 19.10% reduction in maximum displacement and reduced computational time by nearly two hours, as depicted in Figure 11.

Together, these results demonstrate that H-MCTS improves solution quality stage by stage, while also providing computational efficiency in design spaces where standard MCTS becomes impractical.

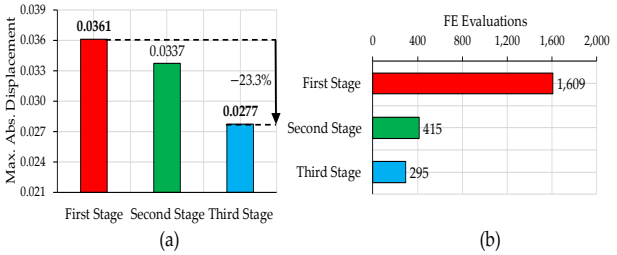

Figure 9: (a) Progressive displacement reduction and (b) lower computational cost over H-MCTS stages.

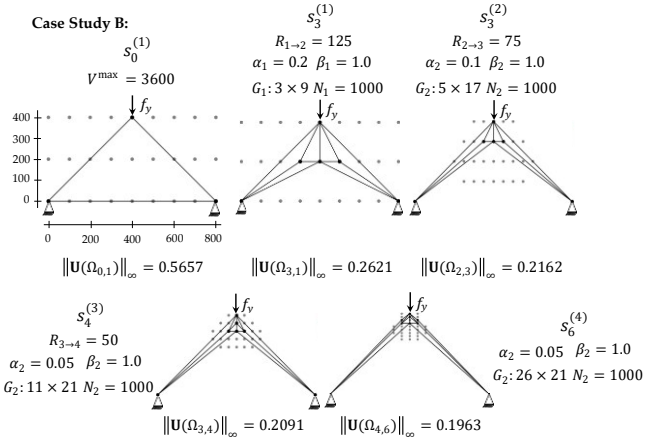

Figure 10: Optimized layouts across successive H-MCTS stages.

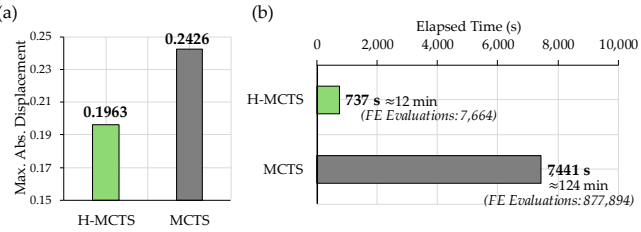

Figure 11: Comparative performance of H-MCTS and MCTS in terms of displacement reduction and computational time.

4.3 Progressive Construction and Load-Induced Structural Morphing. The MCTS-based strategy was applied to guide the progressive construction of a bridge-like truss. The members were modeled as IPE 80 steel sections ($A = 7.64\,cm^2$, $I = 8.489\,cm^4$, $r_g = 1.05\,cm$) with steel S235 ($\rho = 7850\,\frac{kg}{m^3}$, $\sigma_y =$ 235 MPa, E = 210,000 MPa). Unlike previous cases, the seed layout was allowed to grow progressively toward the target support node under self-weight. At each stage, stiffness gains had to be balanced against increased weight, with the objective of minimizing maximum displacement. Practical constraints included a volume limit of 0.046 m^3, a mass limit of 361.1 kg with a 47.1 kg extra allowance, and a maximum element length of 2.5 m, which prevents trivial long-member designs and promotes finer, more efficient topologies. These constraints reflect fabrication, transportation, and assembly restrictions.

As depicted in Figure 12, the design domain measured 10 m × 3.75 m, discretized on a 4×9 grid with a central passive region. The progressive sequence of configurations was obtained using the UCT_j^{α} policy ($\alpha = 0.3$) and 1000 episodes.

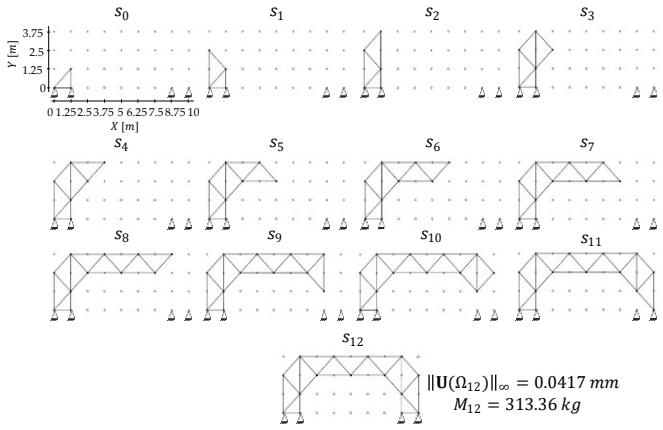

Figure 12: Sequence of truss layouts under self-weight.

The final configuration served as the basis for morphing under a 5000 kg moving load. In the offline phase, configurations were precomputed for discrete load positions, using the remaining mass capacity of 47.74 kg. The online phase interpolated between stored solutions, producing immediate results and achieving an average accuracy (compared to the H-MCTS outcome) of 88.83%, with the best value of 90.87% at symmetric load positions.

Figure 13 shows that the structural responses were found to be symmetric for mirrored load positions, and the algorithm explored a consistent number of decision nodes across these cases.

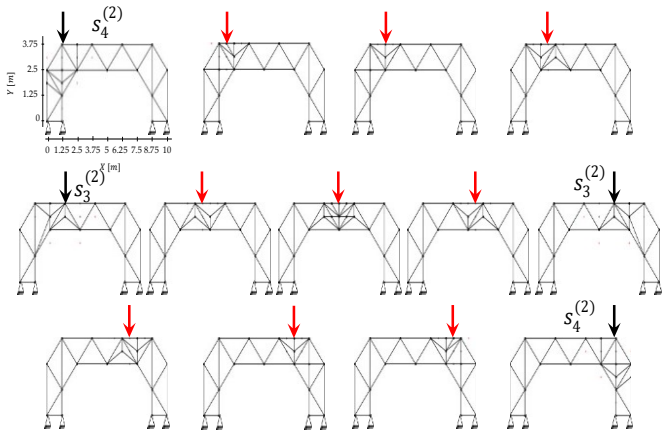

Figure 13: Structural adaptation results for each loading position, showing morphing of the truss topology with H-MCTS (stage k = 2; black arrows) and approximated layouts (red arrows). Structural symmetry about the central axis is evident.

To provide a comparative assessment, the maximum displacements recorded for each loading position are reported in Figure 14. The light-colored bars denote the initial seed configuration, whereas the dark bars correspond to the optimized layouts (either H-MCTS or interpolation), highlighting reduced displacements and confirming the expected structural symmetry.

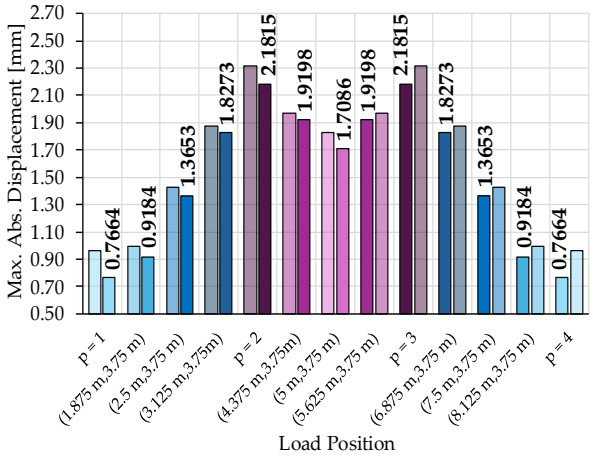

Figure 14: Maximum displacement reduction across load positions for optimized versus initial layouts.

4.4 Damage-Aware Structural Adaptation. The same bridge-like truss system was examined under a damage scenario. Damage was represented as a reduction in the material properties of a specific member $e_{d=1}$. High-temperature exposure was considered the primary mechanism, leading to thermal degradation. In this case, the Young's modulus E_1 was reduced from 210,000 MPa to 84,000 MPa, and the yield strength $\sigma_{y,1}$ from 235 MPa to 141 MPa.

The algorithm identified the damaged element e_1 and verified whether it was critical. A region was defined as critical if that member exceeded its yield strength $\sigma_{y,1}$ or its buckling capacity $\sigma_{\mathrm{cr},1}$. When no critical region was present, the allowable structural volume and mass were 0.046 m^3 and 361.1 kg. When a critical region was activated, these limits increased to 0.052 m^3 and 408.2 kg. During online approximation, further allowances were required to ensure adequate reinforcement and allow intermediate adjustments, raising the maximum volume to 0.060 m^3 and the mass to 471 kg.

Figure 15 shows the axial stress evaluation for the load case at $n_h^{(p=2)}$, highlighting the mitigation of stress not only in the damaged member e_1 but also in the slender member e_2.

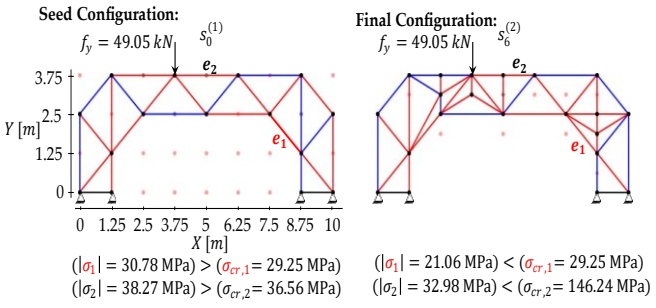

Figure 15: Stress reduction in the offline phase; blue members show tension, red members show compression.

In load cases between $p = 2$ and $p = 3$, the buckling threshold was exceeded in the seed configuration. Optimization through both H-MCTS and interpolation redistributed the stresses through added reinforcements, reducing stress levels in the

critical element. The maximum reduction relative to the initial values was 72.83% in the offline phase and 68.80% in the online phase.

Figure 16 reports the displacement response for all load conditions. Compared to the undamaged case, structural symmetry was lost due to the weakened member. Larger displacements were observed for loads applied between $p = 2$ and $p = 3$, while the lowest displacement occurred at $p = 4$.

Figure 17 presents the optimized layouts for all load positions, illustrating the effect of critical region activation and the resulting reinforcement strategies. For load positions where no critical region was activated, the layouts remain symmetric.

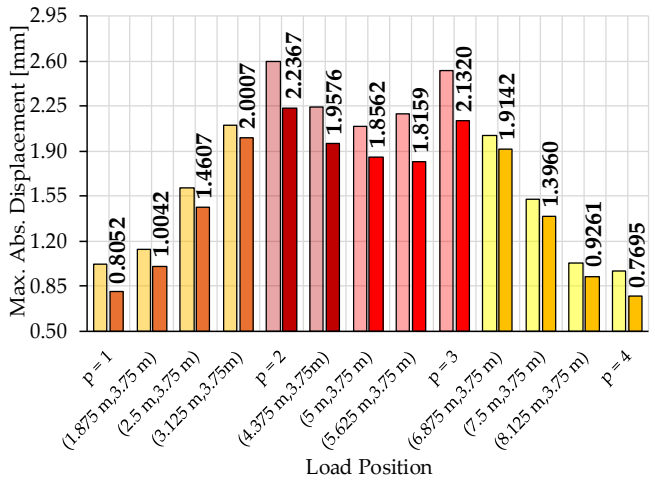

Figure 16: Maximum displacements under the damage scenario, showing asymmetry and performance variation.

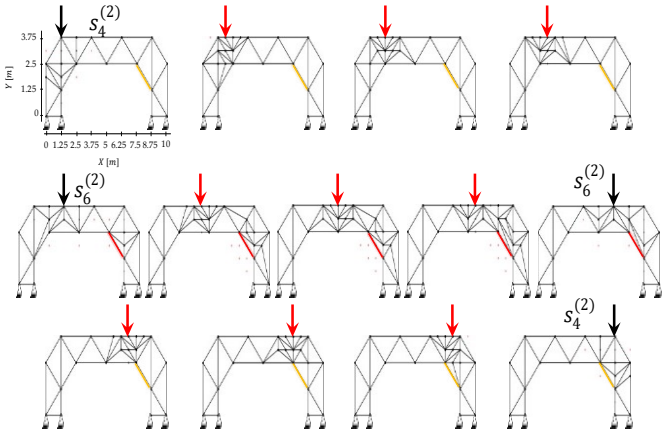

Figure 17: Final structural designs with and without critical region activation, with red showing activation and yellow showing no activation.

5 Conclusions and Future Perspectives

This research has introduced and evaluated the H-MCTS framework for truss optimization under moving loads and material degradation. The method was designed to address adaptability and computational efficiency in truss optimization compared to baseline MCTS [2] and other learning-based strategies [1].

The balance between exploration and exploitation was improved through the MixMax policy $\mathrm{UCT}_j^{\alpha\beta}$, which consistently enhanced convergence and solution quality.

The robustness of the proposed H-MCTS framework was confirmed on the benchmark Case Study E from [2]. Curse of dimensionality, a critical limitation of standard MCTS in dense grids, was effectively addressed by the hierarchical refinement strategy. Meanwhile, substantial computational savings were achieved through this approach.

A comprehensive analysis of a bridge-like structure was performed by progressively constructing it and then adapting it to moving loads through offline precomputation and online approximation. The online phase provided immediate results, eliminating the computational effort spent on learning while retaining an accuracy of up to 90.9%. Furthermore, a damage-aware extension was introduced in this bridge-like case study by reproducing thermal degradation effects in a truss member. The critical region was identified when yield or buckling thresholds were exceeded, and localized reinforcement was applied. Both offline and online procedures significantly reduced stresses and displacements, with stress reductions of up to 72.8% in the damaged element.

Collectively, these findings offer a pathway toward intelligent structures bridging material-level mechano-intelligence [4] and structural-scale adaptivity.

Future work may include the integration of variable cross-sectional areas, circular design strategies through the reuse of structural members, and deep learning for improved search guidance. Extensions to continuous design domains, three-dimensional structures, and dynamic loading conditions would further broaden the scope and practical applicability of the framework.

6 References

[1] M. E. Ororbia and G. P. Warn, "Design synthesis of structural systems as a Markov decision process solved with deep reinforcement learning," *Journal of Mechanical Design*, vol. 145, no. 6, p. 061701, June 2023, doi: 10.1115/1.4056693.

[2] G. Garayalde, L. Rosafalco, M. Torzoni and A. Corigliano, "Mastering truss structure optimization with tree search," *Journal of Mechanical Design*, vol. 147, no. 10, p. 101702, October 2025, doi: 10.1115/1.4068300.

[3] S. Ariyurek, A. Betin-Can and E. Surer, "Enhancing the Monte Carlo Tree Search algorithm for video game testing," in *IEEE Conference on Games (CoG)*, Osaka, Japan, 2020, doi: 10.1109/CoG47356.2020.9231706.

[4] Y. Zhang, A. Deshmukh and K.-W. Wang, "Embodying multifunctional mechano-intelligence in and through phononic metastructures harnessing physical reservoir computing," *Advanced Science*, vol. 10, no. 34, p. 2305074, 23 October 2023, doi: 10.1002/advs.202305074.