From Waste to Structure

A Deep Reinforcement Learning Approach to Circular Design

by

Karl-Johan I. Sørensen
B.Arch.
Royal Danish Academy of Fine Arts, 2018

Submitted to the Department of Architecture and the
Department of Civil and Environmental Engineering
in Partial Fulfillment of the Requirements for the Degrees of

Master of Science in Architecture Studies and
Master of Science in Civil and Environmental Engineering
at the
Massachusetts Institute of Technology
May 2024

Authored by: Karl-Johan I. Sørensen
Department of Architecture
April 29, 2024

Certified by: Lawrence Sass
Associate Professor of Computation and Design
Thesis Advisor

Certified by: Caitlin T. Mueller
Associate Professor in Civil and Environmental Engineering and Architecture
Thesis Advisor

Accepted by: Leslie K. Norford
Professor of Building Technology
Chair, Department Committee on Graduate Students

Accepted by: Heidi Nepf
Donald and Martha Harleman Professor of Civil and Environmental Engineering
Chair, Civil and Environmental Engineering Committee on Graduate Students

Thesis Committee

Caitlin T. Mueller, PhD
Associate Professor in Civil and Environmental Engineering and Architecture
Thesis Advisor

Lawrence Sass, PhD
Associate Professor of Computation and Design
Thesis Advisor

From Waste to Structure

A Deep Reinforcement Learning Approach to Circular Design

By
Karl-Johan I. Sørensen

Submitted to the Department of Architecture on April 29, 2024,
in Partial Fulfillment of the Requirements for the Degree of Master of Science in Architecture Studies and Master of Science in Civil and Environmental Engineering

Abstract

The design-to-construction process of buildings predominantly follows a top-down linear workflow, where a design is drawn and subsequently refined to determine the required materials and components. This approach assumes an infinite material supply or the capability to manufacture what is needed for the design. Constructing in this manner is resource-intensive and wasteful, making it incompatible with our global climate goals. One way to significantly reduce our material and environmental footprint is by extending the lifespan of building materials through circular design practices. In this approach, the available materials define the architecture, inverting the process from top-down to bottom-up. This method, known as Inventory-Constrained Design, enables the creation of new buildings using materials sourced from construction and demolition waste streams. These inventories, characterized by their non-standard and uniquely varied elements, are hard to design with due to the enormous quantity of possible combinations of even a few discrete elements. Identifying a feasible design that aligns with the designer's intent and meets functional requirements becomes an overwhelmingly time-consuming task, heavily reliant on manual trial and error. Computational optimization has been implemented to automate the process, but state-of-the-art algorithms still require manually pre-defining a parametric target design-space or take too long to compute when applied to larger problems.

This thesis proposes a new method for circular design utilizing Deep Reinforcement Learning (RL) to design structures, requiring only a design gesture and the inventory as input. It works by training an artificial neural network to sequentially assemble a structure from inventory elements, following the gesture while meeting a structural goal. Hence, the design layout directly arises from available inventory. After training, the neural net can be employed instantaneously to design new structures with new inventories without any significant computational expense. To evaluate the effectiveness of the RL method, it is applied to the specific problem of inventory-constrained design of planar roof trusses and demonstrated in a realistic example of assembling a long-span roof from a disassembled transmission tower.

Thesis Advisors:
Caitlin T. Mueller, PhD
Title: *Associate Professor in Civil and Environmental Engineering and Architecture*

Lawrence Sass, PhD, Associate Professor of Computation and Design
Title: *Associate Professor of Computation and Design*

Acknowledgments

First, I would like to thank my advisors, Caitlin and Larry, for their support and guidance during the conception and development of this thesis. Caitlin's enthusiasm and deep expertise opened completely new ways of thinking about computational structural design for me—ideas that planted the seed for this thesis. Larry's support through the Design Computation program, believing in my unrefined ideas early on, and always elevating our discussions, have been invaluable.

I would like to thank everyone in the Digital Structures Group. You are a rare group of cool, brilliant people whose lively discussions have shaped many aspects of this work.

Specifically, I would like to thank Keith J. Lee, Ariana Zilliacus, Rachel Blowes, Chloe Hong, Ioannis Mirtsopolous, Celia Chaussabel, Juliana Berglund-Brown, Demircan Tas, Natasha Hirt, Knud Sørensen, John Ochsendorf and Dimitrios Chatzinikolis, for helpful discussions, input, inspiration, and feedback on this work.

I would also like to thank the Danish funds that supported me, allowing me the freedom to focus on my studies during my time at MIT: Augustinus Fonden, Danmark-Amerika Fondet, Dreyers Fond, Friedrich Wilhelm Frank og hustru Angelina Franks Mindelegat, Direktør Ib Henriksens Fond, Knud Højgaards Fond, Tonni og Kaj Munksøs Fond, Louis-Hansen Fonden, Nordea Fonden, and William Demant Fonden.

I extend my gratitude to the architecture department and MIT for allowing me to walk among the giants whose shoulders I stand on.

Dedicated to Ari and Rio

Contents

1 Introduction 12
1.1 Organization of the thesis 12
1.2 Motivation 12
1.3 Circularity in the built environment 13
1.3.1 Case studies and built examples 14
1.3.2 Barriers to the adoption of circular construction 17
1.3.3 Digital circularity in architecture 19
1.3.4 Reinforcement Learning 22
1.3.5 Scope 23
1.4 Research Questions 25
1.5 Contributions 25

2 Literature review & background 26
2.1 Generative structural truss design 26
2.1.1 Evolutionary Algorithms 26
2.1.2 Heuristic and grammar-based methods 27
2.1.3 Topology optimization 28
2.2 Computational design with reuse 28
2.2.1 Combinatorial matching 29
2.2.2 Ground-structure topology optimization 30
2.2.3 Shape Grammar / rule-based methods 31
2.2.4 Evolutionary algorithms and simulated annealing 31
2.3 Reinforcement Learning 32
2.3.1 RL for physical reasoning 32
2.3.2 RL for design and optimization 32
2.3.3 RL for assembly and bottom-up design 32
2.3.4 RL for inventory-constrained optimization 33

3 Methodology 35
3.1 Conceptual Overview 35
3.1.1 Markov decision process 36
3.1.2 Policy gradient methods 37
3.1.3 Training 38
3.2 This thesis' implementation 38
3.3 Agent 39
3.3.1 Observation vector 39
3.3.2 Action tuple 40
3.3.3 Policy 40
3.4 Environment 41
3.4.1 Action interpretation for geometric assembly 42
3.4.2 Observation function 46
3.4.3 Reward function and reset flags 47
3.5 Training 49
3.5.1 Procedural inventories and targets 49
3.5.2 Curriculum learning 50
3.5.3 Hyperparameters 50
3.6 Summary of key method parameters 51
3.7 How to evaluate success 52

3.8 Inference ... 52
3.9 Implementation ... 54

4 Results ... 57
4.1 Trained models ... 57
4.2 Design Space sampling ... 58
4.3 Experimental setup ... 59
4.3.1 Max displacement (Structural Agent) ... 62
4.3.2 Geometric fit (Structural Agent) ... 62
4.3.3 Model reliability/robustness ... 63
4.3.4 Geometric Diversity Results ... 64
4.3.5 Discussion ... 67
4.4 Design case-study ... 67
4.4.1 Transmission tower definition ... 68
4.4.2 Inventory ... 70
4.4.3 Site & program ... 71
4.4.4 Design Gesture experiments ... 71
4.4.5 Assembling ... 74
4.4.6 Inventory usage ... 76
4.4.7 FEM analysis ... 78
4.4.8 Roof-truss interface ... 78
4.4.9 Construction Detail ... 80
4.4.10 Discussion ... 83
4.5 Discussion of Results ... 83

5 Conclusion ... 83
5.1 Summary of contributions ... 83
5.2 Potential Impact ... 84
5.3 Limitations and future work ... 84
5.4 Concluding remarks ... 85

6 Bibliography ... 86

7 Appendix ... 93
7.1 Roof sandwich plate ... 93
7.2 Example sequence ... 93
7.3 Training script ... 101
7.4 Evaluation script (inference) ... 103
7.5 Environmnent files ... 105

List of Figures

Figure 1: Top: La Cuisine building at the Winnipeg Folk Festival, bottom left: deconstruction of a steel-framed building nearby provided structural material for the new building, bottom middle: investigation of design alternatives using available steel, bottom right: the steel during erection – all images are courtesy of Tom Monteyne of Monteyne Architecture Works. 14
Figure 2: The City of Vancouver Materials Testing Laboratory uses materials from local demolitions. By Martin Tessler, courtesy: Perkins+Will. 15
Figure 3: EU Headquarters façade showing reused window frames, courtesy of Philippe Samyn and Partners. 15
Figure 4: Lexington Big Dig House. Left: the high-way bridge being disassembled, middle: constructing the house, right: the complete building, courtesy of Single Speed Design 16
Figure 5: Catalogue page from the Chicago House Wrecking Co., illustrating their inventory of reclaimed structural iron from the Louisiana Purchase Expedition (Chicago Wrecking Company, 1906). 18
Figure 6: General framework for irregular inventory assemblies from Cousin et al., (2023). 21
Figure 7: Sub-processes of Digital Circularity. 23
Figure 8: Trusses in different structures. Left: Roof truss from chapel in Otaniemi, Finland (© Ismael García Rios), right: space truss on the International Space Station (© NASA). 24
Figure 9: A few examples of randomly generated trusses from 20 unique bars. 25
Figure 10: Evolutionary optimization leading to truss design (Xie & Steven 1993). 26
Figure 11: discrete optimization of lattice structures using Genetic Algorithms (Rajeev & Krishnamoorthy, 1992). 27
Figure 12: Ground-structure-based truss optimization steps (Gilbert & Tyas, 2003). 28
Figure 13: Left: tree-fork pavilion (Amtsberg et al., 2022), right: Ski pavilion (© 2017 J.Brütting). 29
Figure 14: Computational matching using the Hungarian Algorithm (Huang et. Al 2021). 30
Figure 15: Conceptual diagram of the Reinforcement Learning loop, showing the action, observation, reward cycle. 35
Figure 16: Deep neural net policy architecture. 41
Figure 17: The RL environment flow chart 42
Figure 18: First element selection procedure. 44
Figure 19: Setting the local coordinate system to the current position on the gesture curve. 44
Figure 20: Selecting the inventory elements for case 1 (Decision point within inventory length range). 45
Figure 21: Selecting inventory elements for case 2 (Decision point outside inventory length range). 45
Figure 22: Observation construction in the environment. Top right shows the global coordinates of t, the target gesture, and the current triangle coordinates. Left, shows these mapped to the local coordinate system, and the middle shows a bar-chart of the binned inventory observation. 46
Figure 23: Structural analysis setup. Arrows indicate the applied point loads in kN. e is the region the truss has to reach to activate the structural analysis. The bars are color-coded to show stress-utilization. 48
Figure 24: Reset conditions. 1: intersecting avoid region, 2: veered off path, 3: reached the end, 4: self-intersection. 49
Figure 25: Inventory scanned by drone. 52

Figure 26: Example user input design gestures. 53
Figure 27: Example trussed dome design with suggested node detail. 54
Figure 28: Example truss RL generation (steps 1 - 4). The top left bar indicates the total accumulated reward. 55
Figure 29: Example truss RL generation (steps 17 - 20). The top left bar indicates the total accumulated reward. 56
Figure 30: Mean Episode Reward training history for the Non-structural Agent. 58
Figure 31: Mean Episode Reward training history for the Structural Agent. 58
Figure 32: Selected experiment cross section. 59
Figure 33: Experimental setup and independent variables. 60
Figure 34: Example generated truss. Note that the truss is always sequentially assembled from left to right. Used inventory bars are highlighted with dashed lines indicating their placement in the truss layout. The percentage indicates how much of the available inventory is used in the truss. 60
Figure 35: Top: Fit calculation, Bottom: max displacement calculation, Color scale indicates compressive (red) and tensile (blue) stress. The displacement visualization is scaled 20X. 61
Figure 36: Effect of different inventories on maximum displacement, truss layout, and inventory utilization. Percentage signs indicate inventory use per truss. Asterisks denote instances where V_{max} exceeds L/240 = 4.17cm. 62
Figure 37: Effect of different inventories on geometric fit. The percentage signs indicate inventory use per truss. 62
Figure 38 Variance of Maximum Displacement Across Different Inventories. The surface illustrates the average maximum displacement (V_{max}) as a function of inventory quantity and spread. For each combination of inventory parameters, 10 random inventories are generated, and their corresponding V_{max} values are depicted as points above and below the surface. 63
Figure 39: Variance of average fit to target curve across different inventories. The surface illustrates the average fit (d_{avg}) as a function of inventory quantity and spread. For each combination of inventory parameters, 10 random inventories are generated, and their corresponding d_{avg} values are depicted as points above and below the surface. 64
Figure 40: Diversity of truss-layouts for experimental agents sampled across different inventory categories (x axis), and design gestures (y-axis). The individual trusses are chosen manually by hand-picking Experimental Agents trained throughout the development of the method. 65
Figure 41: Trusses generated from different inventories and target gestures. Every column has the same inventory and every row the same gesture. Below each truss is its inventory, with highlighted elements used. There are also three numbers showing the (i) inventory percentage used, (ii) fit-score (iii) max displacement score. The top array is generated using the Structural Agent, the bottom array uses the Non-Structural Agent. 66
Figure 42: Transmission towers to be demolished in Roskilde, Denmark. (© Jens Dresling) 68
Figure 43: Transmission tower and assumed element cross section with dimensions in mm. 69
Figure 44: Disassembly of tower from top to bottom and the inventory it creates. The right array shows the lengths sorted digitally; the center array is the inventory indexed in order of disassembly. 70
Figure 45: Proposed tennis-hall site in Roskilde, source: Google Maps. 71
Figure 46: Design experiments with design gesture (top) selected for further development. Highlighted next to each design are the elements used to make it. 72
Figure 47: First design iteration with double curvature and crisscross typology. 1397 elements out of the 1413 elements available were used, but the agent failed to complete the truss. 73

Figure 48: By changing the gesture curves, the agent is able to complete the truss. The final design uses 1237 of the 1413 available elements. 73
Figure 49: Manually adjusting the truss ends to meet the hinge-point. Dashed lines indicate the raw truss generated by the agent, and the dark triangles indicate the manually adjusted elements. 74
Figure 50: Final Hall structural truss design, rendered in perspective (top) and front view (bottom). 75
Figure 51: Mapping of elements from transmission tower to roof truss arches. Bold elements are those used in the design. Left shows the inventory sorted by disassembly sequence, right shows the inventory sorted by length. 76
Figure 52: Mapping of elements from transmission tower to roof truss arches. The length-sorted inventory elements are color-coded by the arch they are used in. 76
Figure 53: Front, plan, and elevation drawings of the tennis court. 77
Figure 54: Simple structural analysis of the longest three-hinged arch truss in the design, subject to gravity load. Red arrows indicate the relative magnitude of w applied to the nodes 78
Figure 55: One of the trusses and the spacer blocks (black) added to accommodate the difference between the roof and the truss, with the height (22cm) of the tallest block. 79
Figure 56: Exploded axonometric of the developable roof panels. 79
Figure 57: Node detail of valence 4 node, showing angle profiles, gusset plate, pinned connections, spacer block, and roof sandwich panel 80
Figure 58: A single developable “roof strip” with three supporting trusses, and spacer blocks. Note that gusset plates are not drawn in this figure 81
Figure 59: Interior perspective render of the final design. 82
Figure 60: Interior perspective render of the final design. 82
Figure 61: Structural analysis of roof panels. 93
Figure 62: Example generated truss (step 1-3). 94
Figure 63: Example generated truss (step 4-6). 95
Figure 64: Example generated truss (step 7-9). 96
Figure 65: Example generated truss (step 10-12). 97
Figure 66: Example generated truss (step 13-15). 98
Figure 67: Example generated truss (step 16-18). 99
Figure 68: Example generated truss (step 19-21). 100
Figure 69: Overview of the evaluation environment definition 105
Figure 70: Overview of the training environment definition. 105

All images are created by the author unless otherwise noted.

1 Introduction

1.1 Organization of the thesis

This thesis introduces a novel computational method for circular design using Deep Reinforcement Learning (RL) to assemble structures from heterogeneous material inventories, requiring only a design gesture and the inventory as input. Section 1 discusses the environmental impact of the building sector and positions the argument for Circular Design. It also establishes the rationale for using Reinforcement Learning to navigate the complexities of designing with reused materials. Section 2 presents a revised review of the literature on generative design for reuse and provides an overview of how Reinforcement Learning has been applied to similar challenges, along with a brief theoretical foundation for RL. Section 3 introduces the main contribution of the thesis, the Reinforcement Learning framework, and describes how Circular Design can be reframed as an RL problem. Section 4 exposes the results, evaluating the approach's effectiveness quantitatively and qualitatively. The thesis concludes in Section 5 by summarizing its contributions and discussing the potential impact and future directions for research.

1.2 Motivation

Our built environment is a significant contributor to the climate crisis and uses an unsustainable amount of material. As Gorgolewski (2018) puts it:, “Today buildings are a graveyard for materials” (Chapter 1) – once used they rarely have a further life. Although there is a growing trend toward recycling Construction & Demolition (C&D) waste, this process often results in down-cycling, where materials are degraded to serve lower-value purposes such as road base (Zhang et al., 2022). Even in the best scenarios, recycling high-value structural elements involves remelting them, which consumes significant energy and diminishes the original value of the materials.

The building sector is responsible for approximately 30%-40% of global greenhouse gas emissions, with at least 10% of these emissions attributable to the embodied emissions in construction materials (Environment, 2022; IPCSS, 2023; United Nations Environment Programme, 2022). The sector also consumes 50% of all raw materials extracted globally, with these materials typically used in only one project before becoming waste (Herczeg et al., 2014). C&D waste constitutes over a third of the total waste production worldwide, with the United States alone generating 544 million tonnes in 2018—more than double its municipal solid waste output (Eurostat, 2018; US EPA, 2016)

To mitigate the embodied carbon in our structures and reintegrate waste into the building economy, a paradigm shift from a linear model (which extracts, uses, and disposes of building materials) to a circular model emphasizing repair, reuse, and recycling is imperative. The latest IPCC report advocates for the building sector to adopt measures such as optimizing and prioritizing the recycling and reuse of buildings and materials to reduce embodied carbon (IPCC, 2023).

The claim of this thesis is that new computational tools will play a crucial role in the adoption of circular measures, enabling designers to reuse C&D waste and other heterogeneous material inventories that would be considered waste or of low value in a linear economy for new projects. The research explores how new computational design tools can aid designers in making better use of reused building materials, focusing on the development of a Deep Reinforcement Learning-based design tool. This tool can suggest new, valid structural designs from the available inventory and address some of the challenges designers face when designing with reused materials.

1.3 Circularity in the built environment

Achieving circularity in building construction can be approached through various methods such as designing for disassembly, reusing salvaged materials, repairing, and recycling. While recycling is the most prevalent method for repurposing obsolete construction materials, it faces significant limitations. Despite the recyclability and reusability of many components found in C&D waste, only 20-30% of such materials are recycled or reused globally (Almeida et al., 2016). A significant issue with recycling is that materials often undergo downcycling. Even materials that are theoretically 100% recyclable require substantial energy for recycling processes like melting steel scrap. Reusing then has the advantage of reducing the carbon emissions that would otherwise be incurred in the melting process (Densley Tingley et al., 2017). Moreover, the purity of materials degrades with each recycling iteration due to contamination with impurities, such as copper in recycled steel (Daehn et al., 2017). In a theoretical case study, Brütting et al. (2019) found that structures composed of reused elements typically have greater mass and lower capacity utilization yet significantly reduce embodied energy and carbon emissions compared to structures built with new elements. Another study demonstrated that incorporating reusable steel elements can lead to a 30% reduction in both embodied energy and CO2 emissions compared to structures made entirely from new steel (Pongiglione & Calderini, 2014). Lastly, material reuse helps to enhance the value chain by maintaining the value of the original form, thus preserving their structural integrity. (Bakker et al., 2018).

Despite these environmental and economic benefits, the challenge persists in maintaining or increasing the value of reclaimed materials through direct reuse. Overcoming this challenge requires innovative design strategies and a shift in industry standards towards a more sustainable construction practice.

1.3.1 Case studies and built examples

Winnipeg Folk Festival La Cuisine

Figure 1: Top: La Cuisine building at the Winnipeg Folk Festival, bottom left: deconstruction of a steel-framed building nearby provided structural material for the new building, bottom middle: investigation of design alternatives using available steel, bottom right: the steel during erection – all images are courtesy of Tom Monteyne of Monteyne Architecture Works.

Designed by Monteyne Architecture Works for the Winnipeg Folk Festival and completed in 2011 at Birds Hill Park, Manitoba, Canada, the La Cuisine building extensively incorporates reused materials. These include structural steel and steel cladding from dismantled industrial buildings due to be demolished, as well as timber framing and cedar hydro poles. The steel was purchased for less than 10% of the equivalent new price and was adapted to new architectural demands through cutting, fitting, welding, and bolting, with the design evolving ad-hoc to accommodate the available materials. The cladding posed challenges due to the difficulty of removing thin steel without incurring damage, highlighting the need to design for disassembly (Gorgolewski, 2018, Chapter 3).

Vancouver Materials Testing Laboratory

Figure 2: The City of Vancouver Materials Testing Laboratory uses materials from local demolitions. By Martin Tessler, courtesy: Perkins+Will.

The City of Vancouver Materials Testing Laboratory employs materials from local demolitions. Approximately three-quarters of the building's structure and fabric are constructed from salvaged and recycled materials, resulting in significant cost savings of around $50,000. However, these savings were somewhat offset by increased construction management fees and labor costs. The primary salvaged materials include wood trusses, glulam beams, and wood decking (Gorgolewski, 2018, Chapter 3).

Brussels Council of the European Union Headquarters

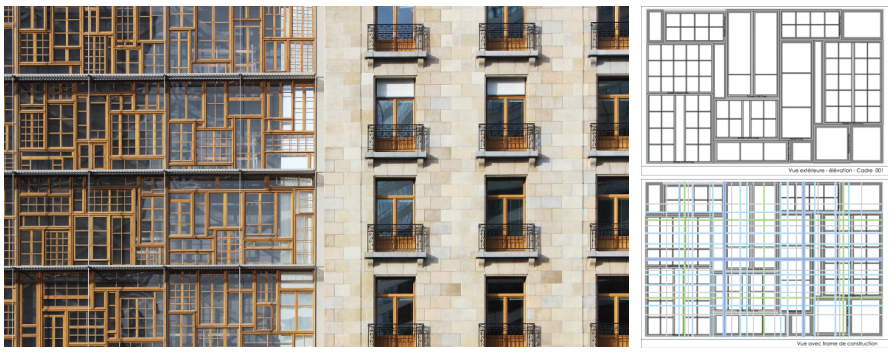


Figure 3: EU Headquarters façade showing reused window frames, courtesy of Philippe Samyn and Partners.

Designed by Philippe Samyn and Partners, along with Studio Valle Progettazioni architects and Buro Happold Ltd engineers in Brussels, Belgium, this office incorporates a double-skin façade formed by approximately 3,000 recycled oak window frames with single-glazing panels sourced from various locations across Europe. The integration of these reclaimed timber windows presented a significant challenge for the designers, who needed to assemble the diverse window dimensions into a coherent and visually appealing patchwork (Gorgolewski, 2018, Chapter 3). This is a common challenge

designers face when reusing heterogeneous inventories, one might speculate if the patchwork could be automated with computational tools such as automated packing algorithms.

Lexiton Big Dig House

Figure 4: Lexington Big Dig House. Left: the high-way bridge being disassembled, middle: constructing the house, right: the complete building, courtesy of Single Speed Design.

The Big Dig House, located in Lexington, MA, was designed by architect John Hong in 2006. This single-family home incorporates approximately 270 tons of concrete and steel salvaged from the dismantling of a temporary highway discarded during Boston's Big Dig. The structural framework of the house utilizes the highway bridge. The project required significant reprocessing efforts, including removing paint from the steel components. However, reusing these elements preserved the structural integrity of the original beams, which were designed for HS20-44 highway specifications and can withstand a loading of 12 kPa (250 psf)—significantly more than the 2 kPa (40 psf) required for residential construction. Consequently, this robust construction enabled unique features for a low-rise residential building, such as water-filled Trombe walls, playgrounds on upper levels, and the incorporation of full-scale landscapes on roofs and balconies ("Big Dig Building," 2014; Gorgolewski, 2018).

Other notable examples

- **Oasis Children's Venture**: Designed by architects Benjamin Marks and Matt Atkins in Stockwell, South London, this structure used structural framing from a deconstructed building. The framing featured bolted and screwed joints to facilitate future disassembly. Additionally, softwood flooring was reclaimed from nearby demolition sites and repurposed within the building.
- **Alliander Office**: Designed by Rau Architects and located in Duiven, the Netherlands, this project utilized 90% of materials from existing on-site buildings, including steel, waste timber, suspended ceilings, sanitary appliances, and blockwork.
- **Roy Stibbs Elementary School**: Constructed in 1994 in Coquitlam, BC, Canada, this school incorporated 270 tonnes of concrete and steel salvaged from a demolished school building, utilizing 75% reused steel.
- **BedZED Project**: A mixed-use development in South London that reused 98 tonnes of steel for its structural framework, resulting in a 4% cost reduction and a significant reduction in

environmental impact. The development reduced CO2 emissions by 81.5 tonnes (Schoon, 2016).

- **Posner Center for International Development**: This project transformed an old horse barn into a development center, reusing materials such as boxcar timber for new stairs, carpet tiles, plumbing fixtures, and kitchen appliances from a dismantled Hewlett Packard office building.
- **2012 London Olympic Stadium:** The design of the stadium incorporated 2,500 tons of re-used steel in the roof truss.

These case studies demonstrate that designing with reuse is not only possible but also brings significant benefits, such as a reduction in the embodied carbon footprint of a building by avoiding emissions related to the purchase of virgin materials, and sometimes even lowering the cost of a project. However, they also reveal that designers cannot rely on standard design patterns for such projects; instead, they must devote extra time and creativity. This underscores the need to integrate automated tools to allow designers to more quickly iterate ideas and adapt designs as the material inventory evolves. Additionally, achieving a reuse project requires extra time and creativity from the entire team involved. As Gorgolewski (2018) notes, "It is more difficult to predict the final outcome of such a process. Such buildings will tend to be more improvisational and, thus, require a client and design team with an open approach, and so may be less suited to some commercial clients." (Chapter 4). Despite these contemporary examples, material reuse is not the norm in today's design and construction practices. It is also notable that most of the large projects use a highly selected inventory of uniform salvage materials. Projects that demonstrate the use of heterogeneous inventories remain mostly at an experimental pavilion scale.

1.3.2 Barriers to the adoption of circular construction

Before industrialization, reusing building materials was more common than today due to the high costs and efforts associated with extracting virgin materials and demolishing buildings. However, the advent of mechanization and the demand for quicker construction processes have diminished the incentive for reuse. Presently, buildings are constructed using standardized components that can be assembled rapidly by unskilled labor. In the following section, some of the barriers perceived by today's industry are outlined.

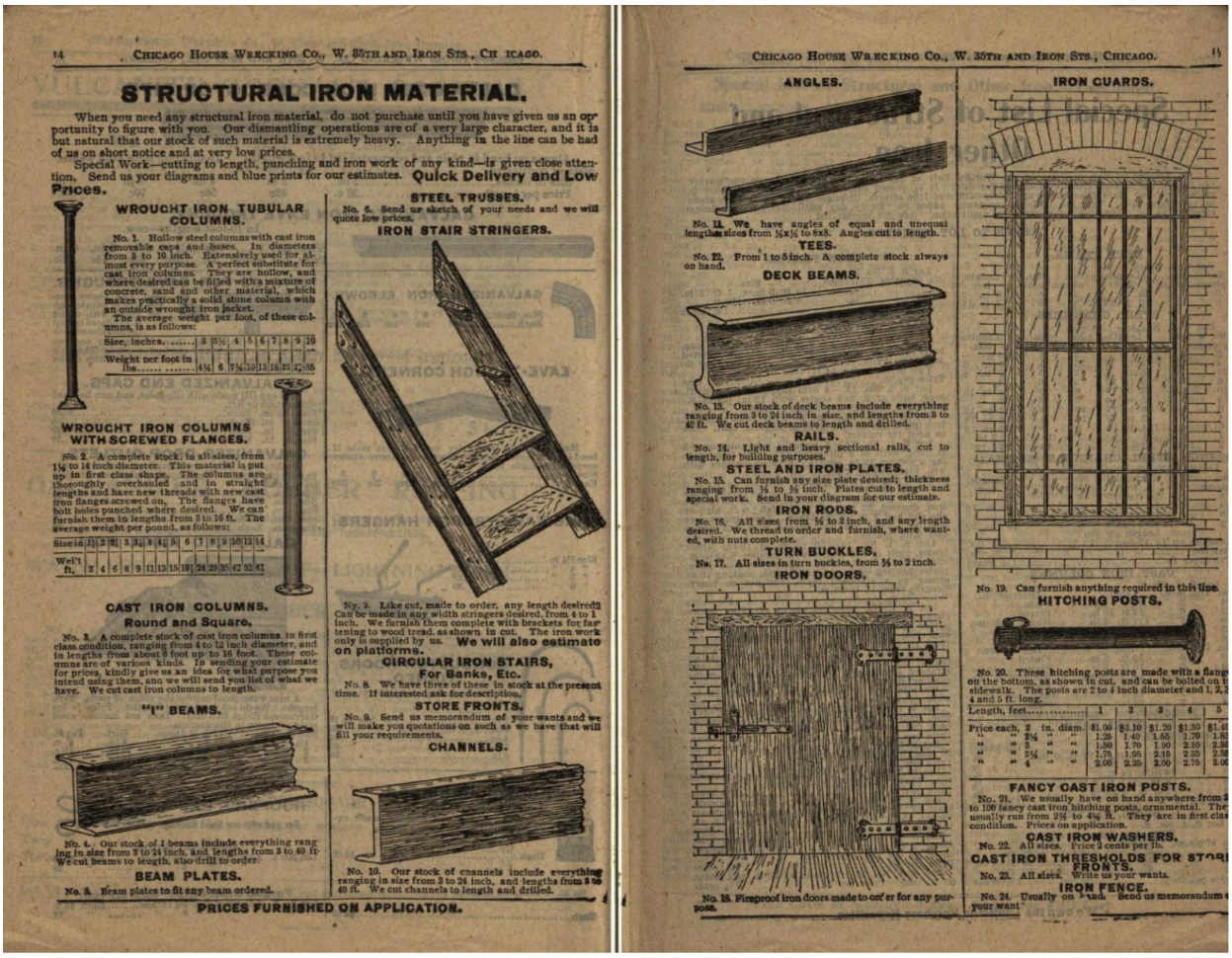


Figure 5: Catalogue page from the Chicago House Wrecking Co., illustrating their inventory of reclaimed structural iron from the Louisiana Purchase Expedition (Chicago Wrecking Company, 1906).

- **Perception of value:** There is a prevalent belief that second-hand materials are potentially inferior and carry greater risks, especially concerning the reuse of structural materials. (Gorgolewski, 2018).
- **Cost:** Research indicates that deconstruction can be 17–25% more costly and take 2–10 times longer than demolition. (Dantata et al., 2005) This increased time and expense might in part be attributed to traditional building designs not being conducive to deconstruction.
- **Availability** Dunant et al. found in their survey (Dunant et al., 2017) that: Existing structures were not designed for deconstruction, making extraction challenging. Local stock was also reported to be limited and, in some cases, inaccessible.
- **Time:** Structural engineers also reported feeling pressed for time (Dunant et al., 2017). Contemporary structural design, building construction, and demolition are optimized for speed, and current circular economy practices take longer than their wasteful counterpart. (Gorgolewski, 2018).

- **Design** In today's industry, working with upcycled construction waste and underutilized natural materials is challenging due to the uniqueness of the pieces, and the inventory they create does not conform to current construction standards (Y. Huang et al., 2021). In their survey, Dunant et al., found that structural engineers perceive the main barriers to the adoption of circular design to be that structural design principles used in existing buildings made it hard to adapt the materials to new systems that use different design principles (2017). This demands a significant amount of effort and skill from contractors as well. While these challenges may be surmountable in smaller projects, they are difficult to address at scale in everyday construction.

> *"By proposing a building made from materials at hand, the project introduces an entirely new paradigm for a project delivery process that has not changed substantially in the last fifty years. It radically alters the way a building is both conceived and made: form follows availability." - (Ruby et al., 2010)*

Despite these barriers, structural engineers viewed the reuse of steel as advantageous, especially when these obstacles can be removed (Dunant et al., 2017). All of these challenges associated with transitioning to a circular construction sector are very real and will require significant changes in our economic models, construction practices, and certification processes. However, they are not theoretically impossible to overcome, and given the urgent and pressing need, we must address them. This thesis proposes that the challenge of designing with reuse can be helped by outsourcing some of the time-consuming processing to advanced computational design algorithms.

1.3.3 Digital circularity in architecture

As we move from the industrial age to the digital age, does this provide an opportunity for a new way of thinking about materials? Products should not have a 'life' but should be part of an ongoing technical cycle. (Goodbun et al., 2014)

Recent advances in computational design and fabrication could aid designers and builders in working with reused materials. The emerging field of digital circularity is developing tools for architects and engineers to work with heterogeneous inventories of used building materials. A review of this field is provided in section 2.2, but for this section, the important information is that Digital Circularity has already established effective workflows to automate many of the steps involved in acquiring, digitizing, designing, and building with reused materials. The steps involved in a typical Digital Circularity workflow are defined in Cousin et al.'s (2023) general framework as:

1. **Digital inventory & characterization:** Working with irregular stock necessitates inspecting elements based on assembly criteria, then sorting, labelling, and organizing them into a manageable physical inventory. Once a physical inventory is established, it is digitized for computational use. Elements are 3D-scanned or captured in 2D. The geometric and case-specific features (angle, areas, length, coordinates, etc.) of digital elements are expressed as arrays of fixed length vectors.

2. **Target Typology:** Based on experience and intuition of a material's potentials, assembly strategies, and conceptual intents, a target typology is defined by human designers in a parametric model. Seeking adjustable parameters, the model is discretized to be geometrically comparable with that of the digital inventory.
3. **Allocation:** The arrays of fixed length vectors are cross referenced and compared by computing Euclidean distance or algorithms like Iterative Closest Point. The computed difference between each pair of inventory and target elements is designated as Cost, which is then compiled in a Cost Matrix. The Hungarian matching algorithm is used to find the best global match and assign elements for a complete design. This match has a Total Cost, which characterizes the overall quality of the assignment. The Total Cost and performance metrics define an objective function for an optimization loop with the target's parametric model and library permutations as parameter inputs. Users can adjust parameters to explore designs. Post design exploration, a digital model is output with all geometric information required for fabrication and assembly.
4. **Assembling**: Manufacturing practices based on material and assembly are required to process inventory elements. These are unique to each element and can be derived from the digital assembly model. The tolerance of each connection depends on the assembly system and element interfaces. Systems with high tolerance can accommodate irregularities at interfaces and require less processing. Assembly systems with low tolerance require tailored jigs and complex registration strategies on site. In these cases, researchers leverage precise digital fabrication methods. MR is an accessible tool for fabrication and construction that enables digital AR overlays on physical elements. The overlays translate fabrication instructions derived from the digital assembly model and allow manufacturing to be executed with accessible tools and infrastructure. (p 4.)

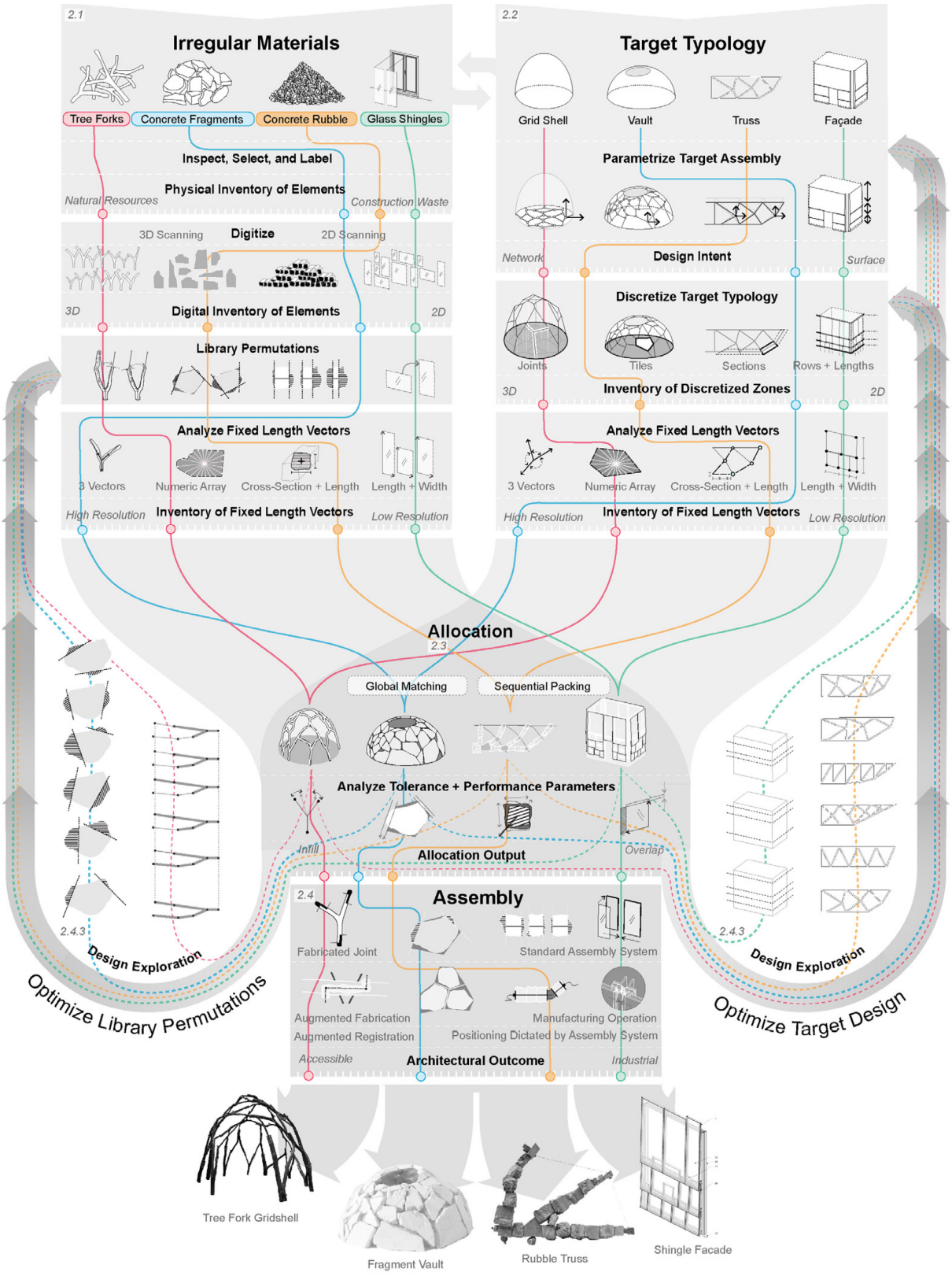


Figure 6: General framework for irregular inventory assemblies from Cousin et al., (2023).

1.3.4 Reinforcement Learning

The field of Digital Circularity is currently employing computational approaches such as combinatorial matching, branch search algorithms, and topology optimization to optimally solve assignment problems. While these conventional optimization methods are effective for tasks like matching an inventory to a predefined design typology or finding an optimal assembly for a small component or architecture pavilion, they become computationally intensive for larger problems and face other significant limitations. These limitations include the requirement for predefined parameters and adherence to rigid, deterministic frameworks. This is particularly challenging due to the variability and uncertainty inherent in reused materials, where the vast number of unique combinations of discrete elements quickly escalates the complexity of design problems. Additionally, these algorithms optimize one problem at a time and must restart when the input changes. Some of these problems can be expected to be solved as computer hardware improves. More importantly, conventional optimization fails to address the fundamental question of bottom-up design with reuse:

"What <u>can</u> be designed with this inventory?"

As Cousin et al. (2023) state in the current state-of-the-art framework for Digital Circularity, a target typology must be predefined by human designers in a parametric model based on experience and intuition of a material's potentials, assembly strategies, and conceptual intents (p. 4). Therefore, there is a need for a computational approach that keeps the design space open and itself generates a topology or layout based directly on the inventory at hand. This approach should require only a suggestive gesture of the design intent, such as a desired span and curvature of a roof or a floor area to encapsulate with walls, while being constrained only by the available inventory and required load case.

The problem with conventional optimization is that if we do not constrain the problem to a predefined layout, the dimensionality of the problem and the size of the search space skyrocket to virtually infinite proportions, as we must consider every possible combination of elements. When humans solve a bottom-up combinatorial design problem, such as tightly packing a trunk for a long trip, we use steps of observation (what is there, what is left) and intuitive rules to quickly decide on what order to pack the objects in and where they should be placed. Such intuitive rules, or heuristics, are practical problem-solving methods that simplify complex decisions or processes, making them more manageable. They are not guaranteed to be perfect or optimal, but they often provide a good enough solution within a reasonable timeframe. These heuristics feel intuitive to us and are learned through years of trial-and-error and accumulated knowledge, resulting in an implicit cognitive model for approaching such problems in general.

Our closest computational method to achieve a similar feat is in Machine Learning, where artificial neural networks are trained to enable computers to perform specific tasks without explicit instructions, instead relying on patterns and inference derived from data (Mitchell, 2017). Since a dataset of heterogeneous material inventories and what designs they can create does not currently exist, this thesis proposes using Deep Reinforcement Learning, which does not require such a dataset. Instead, it relies on agents learning optimal actions through trial and error within a simulated environment,

receiving feedback in the form of rewards or penalties (Sutton & Barto, 1998). Trained on diverse design scenarios involving reused materials, Deep RL agents could potentially learn to dynamically adjust to the unique characteristics of these materials, thus developing a deeper understanding of the opportunities different reused inventories present. This mimics the heuristic intuitive human design process described before in a formalized way encapsulated in the RL cycle of state observation and action.

In recent years, Deep Learning techniques have been increasingly applied to combinatorial optimization problems (Bello et al., 2017; Khalil et al., 2017), and initial explorations in applying RL to construction-related tasks have shown promise, particularly in sequential assembly problems (Bapst et al., 2019). Given these successes, this thesis proposes to extend these insights to Circular Design, a field with similar combinatorial challenges, by developing a Deep RL approach to inventory-constrained design—an unexplored area. This approach aims to significantly enhance the capabilities of architects and engineers in designing with reused materials by efficiently suggesting complete structural layouts from available inventories. This exploration presents an opportunity for substantial innovation and the potential for transformative changes in how materials are reused in architectural design.

1.3.5 Scope

As established, achieving circularity in design is a multifaceted problem, necessitating holistic changes in industry workflows. This thesis specifically addresses the design phase within the broader framework of a circular built environment and digital circularity, which encompasses sub-processes from disassembly to material testing and characterization (Figure 7).

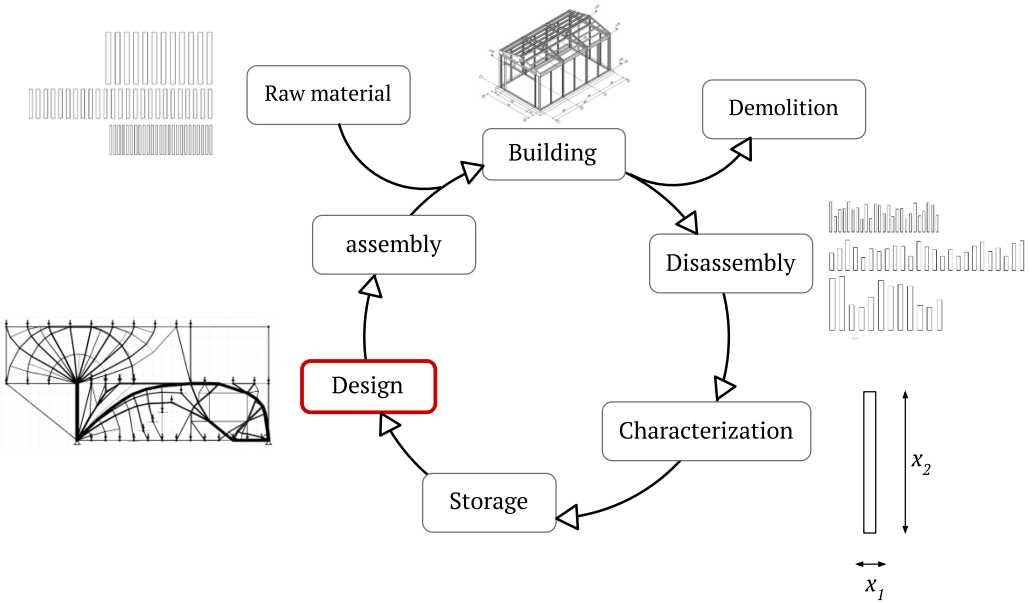


Figure 7: Sub-processes of Digital Circularity.

Reused materials are often relegated to non-critical architectural elements, such as facade panels or interior surfaces, and are seldom used in structural components. Given that load-bearing systems significantly impact the embodied carbon emissions of buildings (Allwood et al., 2012), and considering steel production accounted for 7-9% of global anthropogenic CO2 emissions in 2019 (World Steel Association, 2020), this thesis focuses on designing structures from reused materials, primarily steel. To further narrow the scope, this research concentrates on designing planar trusses—a ubiquitous structural element known for its efficiency and versatility in spanning roofs and floors. These are typically constructed from small linear elements, which constitute a large portion of C&D waste. These elements are not commonly reused due to them being short and their varying lengths not meeting requirements for conventional structural design.

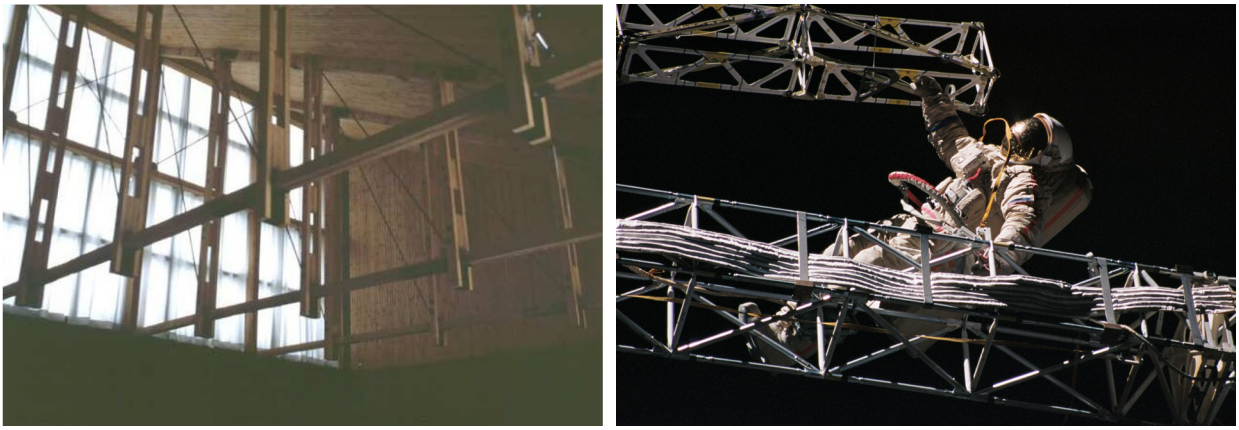


Figure 8: Trusses in different structures. Left: Roof truss from chapel in Otaniemi, Finland (© Ismael García Rios), right: space truss on the International Space Station (© NASA).

The combinatorial design challenge of creating trusses from an odd inventory of bars is surprisingly daunting. For example, just 20 bars of unique length can be assembled into approximately 2.4 quintillion (10^{18}) different trusses. This complexity makes bottom-up design from such inventories both non-trivial and a suitable test case for the current technological capabilities of Reinforcement Learning (RL) algorithms. The conceptual workflow developed for trusses could then be expanded to other structural typologies in future research.

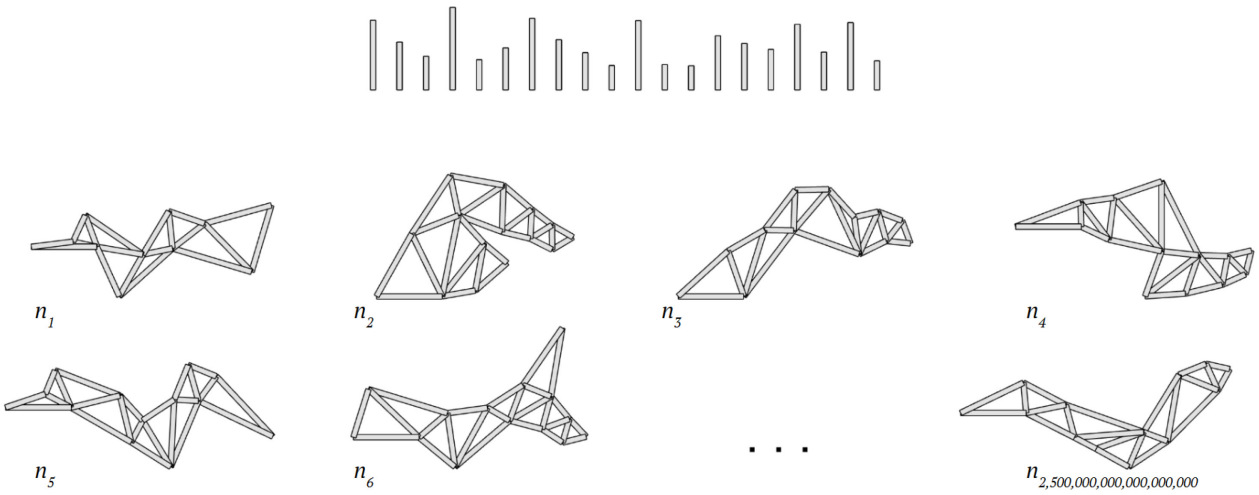


Figure 9: A few examples of randomly generated trusses from 20 unique bars.

1.4 Research Questions

Based on the background and scope of this investigation, the thesis poses the following research questions:

- How can we design structural trusses from heterogeneous inventories of reused linear elements?
- Can Circular Design be effectively formulated as a Deep RL problem, and to what extent can the input inventory directly influence the design layout?

1.5 Contributions

The contributions of this thesis include a computational method for designing structures with reuse, utilizing Deep Reinforcement Learning to sequentially assemble the structure from heterogeneous inventories. This contribution encompasses selecting an appropriate RL policy and training algorithm, defining the agent's observations and actions, and formulating a simulated geometric environment wherein an agent can learn to assemble trusses. Additionally, the thesis provides a practical implementation of this framework and demonstrates its effectiveness for planar truss design through analytical and qualitative analysis.

2 Literature review & background

2.1 Generative structural truss design

Generative Design refers to a group of design methodologies based on the generation and evaluation of outputs in a repetitive process (Art et al., 2002). Generative structural truss design falls into several categories: Evolutionary algorithms, Gradient-based optimization, heuristic methods, and Topology optimization. Computational generative design has been applied to the design of truss structures for a while. However, these do not consider constrained inventories.

2.1.1 Evolutionary Algorithms

Inspired by natural selection principles, these algorithms iteratively modify a population of design solutions, selecting the best performers for reproduction and mutation to generate new design iterations.

(Xie & Steven, 1993) introduced a straightforward evolutionary procedure for structural optimization, which adds or removes material from a discrete design domain after calculating the internal stress distribution. (Rajeev & Krishnamoorthy, 1992) explored discrete optimization of structures using Genetic Algorithms (GA), marking a significant early adoption of GAs in structural engineering. These algorithms optimize structural systems with discrete design variables. (Mueller & Ochsendorf, 2015) further expanded the application of evolutionary algorithms in their work on StructureFit, where an initial truss is defined, and design variables subsequently changed to explore the design space by directly selecting preferred designs. This integration facilitates a more user-interactive exploration of the design space, aligning the optimization process more closely with specific design intents and objectives.

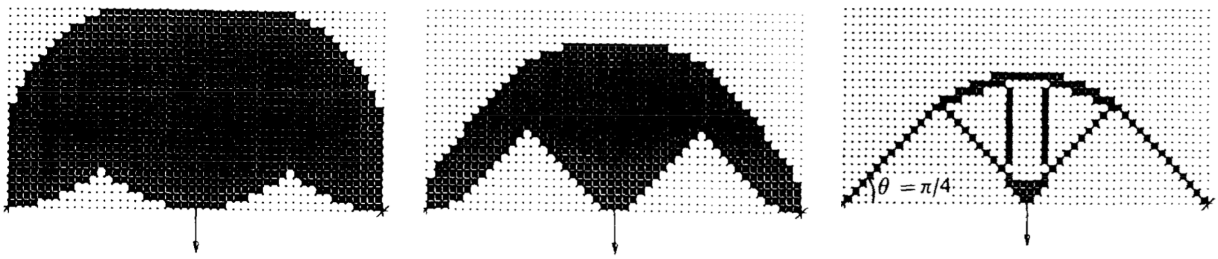


Figure 10: Evolutionary optimization leading to truss design (Xie & Steven 1993).

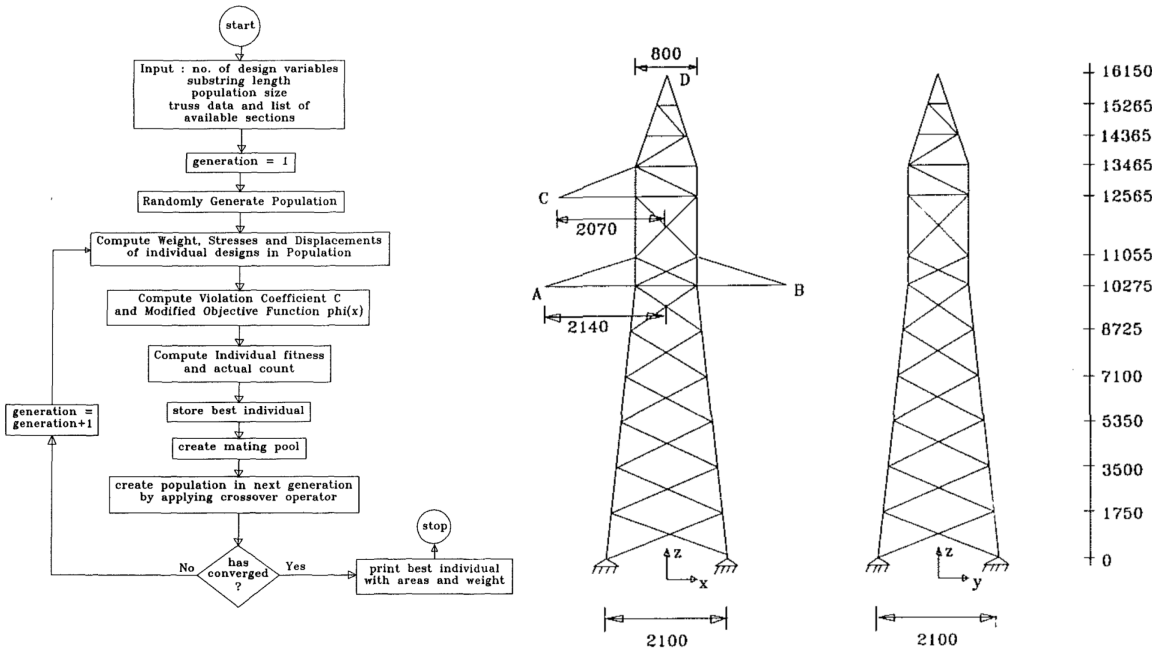


Figure 11: discrete optimization of lattice structures using Genetic Algorithms (Rajeev & Krishnamoorthy, 1992).

2.1.2 Heuristic and grammar-based methods

This approach utilizes predefined grammars to guide the creation of truss designs, specifying how elements can be combined or added to meet design criteria. These grammars are often paired with algorithms like simulated annealing and particle swarm optimization to enhance the generative process. Rule-based and heuristic methods search for solutions stochastically, relying on specific rules rather than direct gradient calculations.

(Shea et al., 2005; Shea & Cagan, 1999) introduced eiform, which employs shape annealing to generate novel roof trusses. This method combines grammatical parametric shape generation to add flexibility to the design process. While using simple grammatical rules can create complex designs, controlling these grammars can be challenging and may lead to unpredictable outcomes.

(J. Lee et al., 2015, 2016) developed a computational design methodology that integrates shape grammars with graphic statics to rapidly generate diverse, discrete structures in static equilibrium. However, this method's reliance on randomness makes it difficult to control. (Cloesen, 2018), extended this work by incorporating evolutionary algorithms to optimize the generation process. Similarly (Mirtsopoulos & Fivet, 2023) enhanced the methodology with policy-based exploration, adding more geometrical and topological control by introducing constraints to the rules.

2.1.3 Topology optimization

Topology optimization is typically applied to continuous material distribution problems but can also be adapted for truss design by treating truss connectivity and nodal placement as variables. This method iteratively removes inefficient material from a structure through analytical gradient-based optimization, to find an optimal material layout within a given design space. Truss structures are usually optimized starting from a ground structure, which comprises all possible member positions between truss nodes (Dorn et al., 1964). To simplify the optimization process, cross-sectional areas are often treated as continuous variables, which are adjusted to find a unique, single optimal solution.

Gilbert and Tyas (2003) explored the layout optimization of large pin-jointed frames, demonstrating the applicability of modern computational techniques to large-scale problems. However, the ground structure approach is generally feasible only for modestly sized problems due to the complexity of containing every conceivable member connecting the nodes. In layout optimization, members are added iteratively, which allows the handling of larger frames effectively.

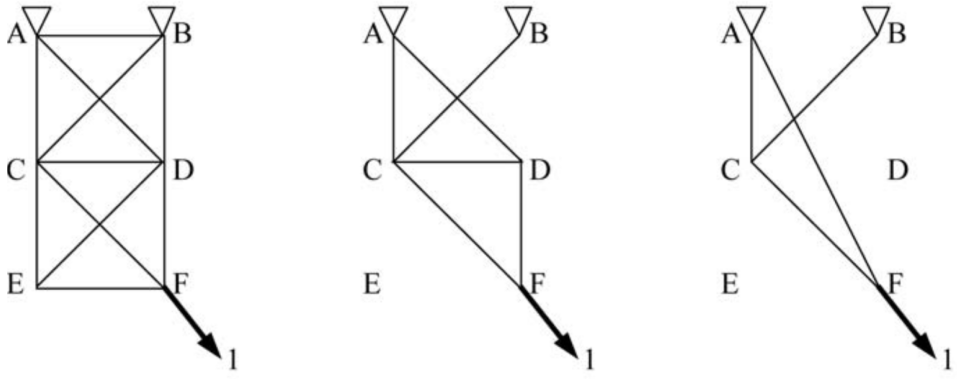


Figure 12: Ground-structure-based truss optimization steps (Gilbert & Tyas, 2003).

All these traditional methods for computational generative designs have been effectively utilized in industry for designing optimal and novel structural trusses. However, none of the approaches account for a constrained inventory and, therefore, cannot be used for Circular Design. In fact, the optimal designs generated with these methods typically require custom fabrication due to the numerous unique elements involved. Next, we will explore how generative design and optimization have been applied in the context of circular design.

2.2 Computational design with reuse

In recent research, computational methods have been utilized to reuse and upcycle irregular materials in design and fabrication processes. Many of these methods have been demonstrated in experimental pavilions across academic institutions. The "Elastic Gridshell Pavilion," illustrated in Figure 13 and designed by the Structural Xploration Lab at EPFL, is one such example. This structure, crafted from reclaimed skis, showcases an innovative approach to repurposing technologies—specifically skis that might otherwise be discarded when they no longer meet their original functional

requirements. Through computational form-finding techniques, the team analyzed the impact of variations in ski bending stiffness on the pavilion's form. The "Mine the Scrap" project by Certain Measures (Nolte et al., 2015) is an artistic research project that explores how artificial intelligence and computer vision can convert scrap materials into architectural forms. The Tree-Fork Pavilion (Figure 13: right) utilizes discarded tree forks as structural, load-bearing joints. This initiative developed a method to reuse tree forks in new structures, taking advantage of their inherent properties as moment-resisting valence-three nodes. Typically viewed as low-value by the industry, these irregular tree forks were computationally matched to Y-shaped nodes within a space truss structure.

These projects highlight the transformative potential of digital workflows in enhancing upcycling practices. However, their applications remain specific to each case, and the computational workflows require a user-defined design target typology, which requires time, resources, and expertise to develop. This thesis proposes a method that requires a less specific target-design input and instead generates the structural topology directly from the inventory, thereby broadening the scope and impact of computational circular design.

Figure 13: Left: tree-fork pavilion (Amtsberg et al., 2022), right: Ski pavilion (© 2017 J.Brütting).

2.2.1 Combinatorial matching

Various algorithms have been applied to the combinatorial optimization problem of matching an inventory to a target design. Simple heuristic algorithms, such as the Greedy Search, are straightforward to implement but tend to be slow. Huang et al. (2021) employed a faster algorithm, known as the Hungarian algorithm, to perform matching for circular designs that incorporate reused structural elements, adopting a top-down philosophy for design with reuse. This method originates from conventional parametric design and optimization, beginning with a parameterized target design concept model. Subsequently, an inventory is algorithmically searched to match the target, ideally aligning both geometric and structural capacities. However, this typically does not result in a perfect fit, necessitating further processing of the inventory.

This methodology was later applied to the design of structures utilizing irregular tree forks (Amtsberg et al., 2022) and rubble trusses (Al Khayat, 2023), which used matching to create a truss consisting of post-tensioned concrete rubble. Cousin et al. (2023) expanded upon this matching framework by generalizing it into a modular framework. They demonstrated how various circular design problems can be described using feature vectors that capture essential properties of the

inventory's capacity and the design's demand. The Hungarian algorithm is then used to find the optimal assignment matrix with the smallest global distance between a set of inventory feature vectors and a set of design feature vectors. This approach is currently the fastest combinatorial optimization method available and provides a useful modular implementation in a design workflow.

A significant limitation of Hungarian matching and other combinatorial matching, search, and optimization approaches is their reliance on a target topology developed based on human designers' experience and intuition within a parametric model. Consequently, these methods do not generate a structural layout solely using the inventory. Furthermore, their performance degrades exponentially as the size of the inventory and design target elements increases.

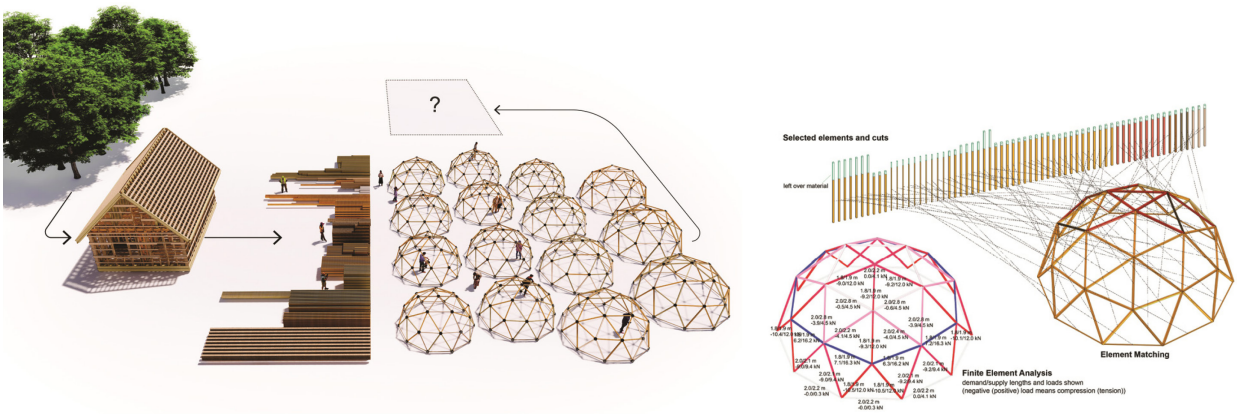


Figure 14: Computational matching using the Hungarian Algorithm (Huang et. Al 2021).

2.2.2 Ground-structure topology optimization

Brütting, Senatore, and Fivet (2019,2022) employed topology optimization in a comprehensive discrete structural optimization formulation for designing trusses using constrained inventories. Their method uses a mixed-integer linear programming framework, which can be solved for global optimality using a branch-and-cut tree search. The process is iterative, involving element assignment and topology optimization, followed by shape optimization to ascertain the optimal fit for length and cross-sections. The method considers stock properties such as structural capacity, length, and availability. However, a significant limitation of their methodology is the requirement for a manually defined ground structure, which must remain of relatively modest size to be feasible. Additionally, while their approach allows for structural optimization, it is slow, especially for larger problems, and still necessitates dealing with off-cuts.

2.2.3 Shape Grammar / rule-based methods

Shape grammar and rule-based methods employ a bottom-up philosophy, originating from the computational method of Shape Grammars (Stiny, 1980), with further advancements in Making Grammars (Knight & Stiny, 2015). These methods utilize predefined, explicit rules to automate and control the process of aggregating materials, ensuring a guaranteed geometric fit of existing materials into the design and aligning more closely with non-computational physical workflows. Recent developments include Rossi (2023), who developed a Grasshopper plugin, WASP, which integrates rule-based aggregations into a user-friendly modular CAD workflow. Steelman (2022) applied WASP in the inventory-constrained structural bridge design, while Seats (2020) explored form-finding for trusses with reuse and graphic statics. However, challenges arise in controlling and integrating additional design intentions. These methods are better suited to inventories of self-similar parts rather than truly diverse material stocks, which is critical in reuse applications. The main drawbacks include difficulty in control, which, when achieved, tends to restrict the design space. Additionally, each new design challenge requires the formulation of new rules, and these methods struggle with solving the design of edge cases present in complex structural systems.

2.2.4 Evolutionary algorithms and simulated annealing

Several researchers have employed Evolutionary and Genetic Algorithms (GA) to optimize component placement and orientation. An early application comes from Fujitani and Jujii (2000) who optimized the structural design of steel frames using a fixed topology from a limited stock of members, employing evolutionary algorithms for weight optimization. More recently, Van Marcke, Laghi, and Carstensen (2024) introduced a method for aggregating partially disassembled triangular elements from reused trusses to design new planar truss structures, incorporating a GA optimization module to select among generated trial designs. Zirek (2023) applied an Evolutionary Algorithm for orienting tree forks, while Marshall et al. (2020) focused on the computational arrangement of demolition debris. Mollica, Zachary, and Self (2016) used evolutionary optimization for dynamically placing tree forks within the target curves of the Architectural Association Wood Chip Barn. Furthermore, Xu et al. (2019) creatively employed evolutionary optimization with genetic algorithms for generative LEGO Technic designs.

These methods are more effective than brute-force search but are still too slow for larger design problems. Some of these methods begin to truly aggregate elements bottom-up, but they are slow due to the enormous search space when using larger inventories. They also fail to learn anything in general after an optimization that can be applied to new, similar problems, which this thesis' method addresses.

2.3 Reinforcement Learning

Reinforcement learning has evolved significantly since its formal study began, particularly with the pivotal contributions of Richard S. Sutton and Andrew G. Barto, whose 1998 book, "Reinforcement Learning: An Introduction," synthesized ideas from AI, psychology, and optimal control to establish a comprehensive framework for RL. RL "is the branch of machine learning concerned with learning how an *agent* should behave in an *environment* to maximize a *reward*." (Charniak, 2018, p. 113). This field has seen numerous successful applications, including the mastering of Chess and Shogi by Silver et al. (2017), the game of Go (Silver et al., 2016), and Atari gaming strategies by Mnih et al. (2013). Recent advancements include the development of AlphaFold by Jumper et al. (2021), solving the Rubik's cube through the efforts of Agostinelli et al. (2019), and improvements in dense 2D packing by Chu and Lin (2019), each highlighting the diverse potential of RL in solving complex, real-world problems.

2.3.1 RL for physical reasoning

Physical reasoning has long been a focus within the field of artificial intelligence. Initial research, such as Winston's work in the study of geometric logical reasoning (1980) expanded in the early 1990s through research in geometric constraint-solving (Bouma et al., 1995). Historically, this area emphasized rule-based and structured representations over learning approaches, mainly due to hardware limitations of the time not being able to handle the large training data required to train machine learning models.

2.3.2 RL for design and optimization

Reinforcement learning has recently become a focus in architecture for design optimization and physical construction, as evidenced by the diverse applications in recent studies. Mandow et al. (2020) and Apolinarska et al. (2021) explore RL in architectural design and construction, respectively. Our research method aligns with Mandow et al. (2020) in using RL for design optimization but diverges as we apply RL to generate designs directly rather than evaluating designs from learned shape grammars. Other notable works include Ruiz Montiel et al. (2013) and Mirra and Pugnale (2022), who explore RL in shape grammars and structural frames, Seo and Kapania (2021) and Hayashi (2021) for structural design concepts and optimization-focused studies by Sun and Ma (2020), (Brown et al., 2022), (Du et al., 2023), and Hayashi and Ohsaki (2021) on topology optimization using RL.

2.3.3 RL for assembly and bottom-up design

Recent studies in reinforcement learning for physical construction and assembly have made significant progress. Li et al. (2017) examined learning to stack blocks at predicted stable points, while Hamrick et al. (2018) focused on attaching blocks to stabilize unstable stacks. (Khalil et al., 2017) explored basic block-stacking by predicting the shortest paths through a transition model, and Janner et al. (2021) developed object representations and coarse-grained physics models for black stacking.

Zhou et al. (2022) developed an intelligent machine that automatically transforms captured portraits into LEGO assemblies. Vallat et al. (2023) focused on scaffold-free construction using robotic arms to assemble hexagonal blocks, while Akizuki et al. (2020) explored generative modeling of voxels to fill a target design volume. Lee and Mueller (2020) investigated protein folding for constructing space trusses, illustrating the breadth of RL applications in physical reasoning and design-by-assembly. Despite these advancements, the complexity of the structures constructed by agents and the degree of design autonomy remain limited, highlighting a crucial challenge in learning to assemble intricate structures with functional objectives.

The field has also seen a growing application of deep learning to combinatorial optimization problems, which are closely related to sequential assembly challenges (Bello et al., 2017; Khalil et al., 2017). For instance, Wietschorke et al. (2020) explored RL for architectural optimization. Additional contributions include robotic assembly of timber joints (Apolinarska et al., 2021), dense 3D packing (Hu et al., 2020), and innovative stacking research by (Chung et al., 2021; Ghasemipour et al., 2022; Yang et al., 2021).

Most of these methods automate construction tasks end-to-end, thereby limiting user interaction. Bapst et al. (2019) used structured agents and graph neural networks for simulated stacking of complete block designs, demonstrating complex construction behaviors in settings with up to 50 objects. This thesis introduces a more interactive approach. It allows users to directly influence the assembly process by specifying a target design gesture, which aligns with the adaptive design strategies proposed by Janner et al., (2019) and Wibranek et al. (2021).

Despite these examples using discrete inventories, few constrain the inventory use. If they do, they use very small kit-of-parts type inventories, such as LEGO blocks, with pre-designed interfaces allowing them to connect and fit together easily. None use heterogeneous inventories or consider their agents for Circular Design practices.

2.3.4 RL for inventory-constrained optimization

The literature presents only two preliminary studies on applying RL for inventory-constrained optimization. Apellaniz et al. (2023) developed an RL framework to address diverse optimization challenges in architecture, focusing on cradle-to-cradle issues. This framework primarily uses RL for optimal assignment, serving more as a matching tool than a fully generative method. (C. Huang, 2021) utilized an RL-based search system that designs new shell structures composed of existing waste materials by aggregating irregular plate fragments within a 3D design domain, optimizing stability and symmetry. Although conceptually similar to the approach of this thesis, Huang focused on plate elements, and the resulting design did not converge to a structurally closed shell suitable for use in architecture. While not specifically using RL, Sterrenberg (2023) introduces a deep learning framework for design optimization with reuse, employing a Variational Autoencoder combined with a surrogate model to optimize space frame structures for structural performance and stock utilization. However, this method is limited to shape optimization and cannot optimize the topology. Both approaches demonstrate significant limitations in their scope and application, highlighting a rich

opportunity for developing open-ended, bottom-up, inventory-constrained generative design tools for Circular Design.

As shown, current implementations of reinforcement learning for design and construction primarily rely on top-down, pre-defined design typologies and use simple inventories of blocks, making them limited in their real-world application to circular structural design. This presents an opportunity to apply Deep RL to generate structural designs in a truly bottom-up manner, starting from an inventory and sequentially assembling the elements guided only by non-negotiable functional constraints and a suggestive design gesture. The proposed method aims to support designers in generating designs without the constraints of predetermined typologies. Deep RL enables the proposed method to learn from each design iteration, improving its ability to generate better designs over time, employing strategies that a human might not be able to identify manually. This approach challenges the traditional top-down design paradigm and represents a novel development in applying reinforcement learning for circular design. The next section will detail the proposed method, applying Deep RL to planar truss design, utilizing a heterogeneous inventory. This method involves training RL agents and generating assembly sequences, where the inventory dictates the design possibilities.

3 Methodology

3.1 Conceptual Overview

In any Reinforcement Learning problem, the Agent and the Environment are the two main components to define. At every timestep, the agent receives an observation vector O_t as input and outputs an action tuple A_t. This action is received by the environment, which interprets it and produces the next environment state S_t. The state provides a complete description of the environment. The environment then outputs a new O_t along with a reward R_t, which is received by the agent and used to calculate the next action. This cycle repeats until the environment outputs a reset signal. During training, the reward updates the agent's policy parameters. Afterwards, the policy is used for inference, and the policy parameters remain static. The following section describes how this general RL framework has been applied to the problem of designing trusses from reused elements. Figure 15 provides a conceptual diagram of the Reinforcement Learning loop, showing the action, observation, and reward cycle for this method.

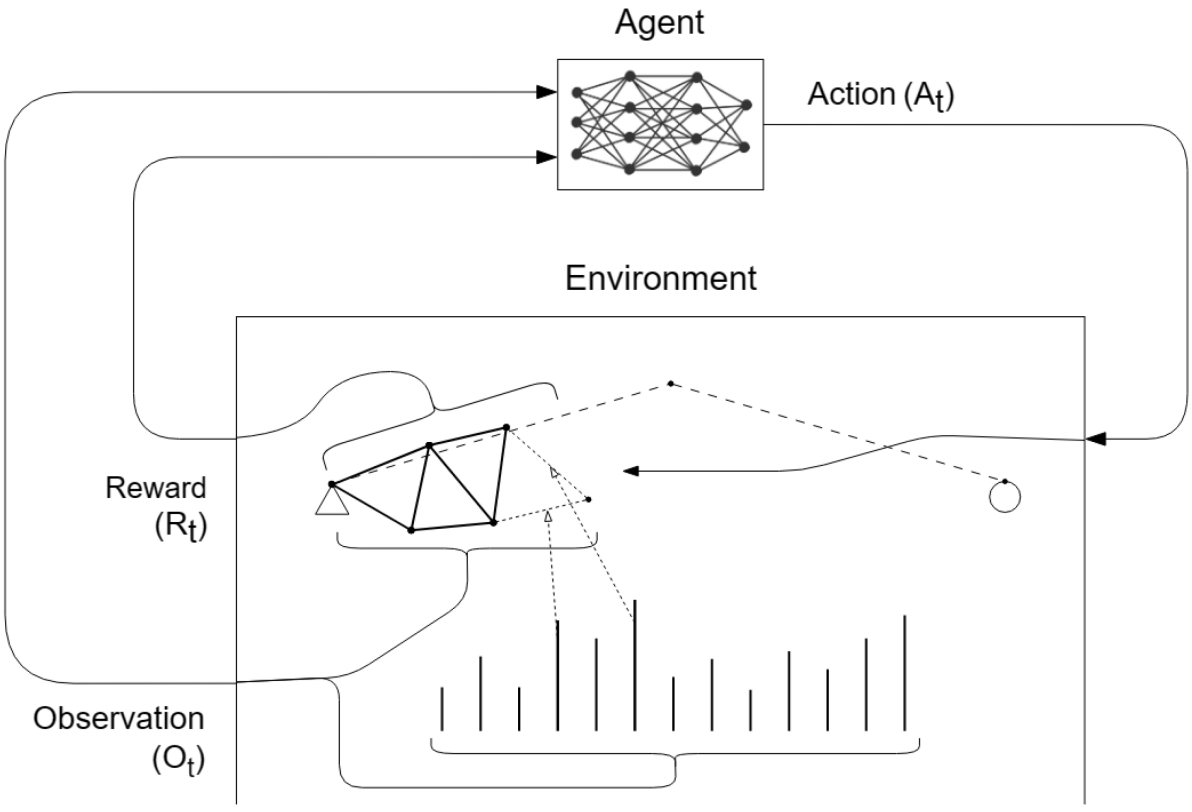


Figure 15: Conceptual diagram of the Reinforcement Learning loop, showing the action, observation, reward cycle.

The method implements a Proximal Policy Optimization (PPO) algorithm within an RL framework structured around a custom environment. Below is a detailed mathematical explanation of the Markov Decision Process (MDP) and PPO as they relate to the RL model used in the implementation.

3.1.1 Markov decision process

In Reinforcement Learning, an agent does not perceive its environment as a human does; instead, it receives a list of numbers: a vector. These numbers can represent various elements, but if they are structured and provide sufficient information about the environment in which the agent is acting, the agent will be able to learn what actions to take. It learns through trial and error, balancing exploration and exploitation. This means sufficiently exploring the state space by testing novel strategies for choosing bars, thus avoiding premature convergence to a local minimum while leveraging previously successful actions. With the right balance of exploration and exploitation, the agent can efficiently navigate the vast combinatorial space of bottom-up truss design without converging too early or wandering randomly without improvement. This trial-and-error approach is formalized in what is known as a Markov Decision Process (MDP) (Bellman & Dreyfus, 2010; Puterman, 1994; Sutton & Barto, 1998).

MDP is a formal framework used to model decision-making scenarios where outcomes are partly stochastic and partly under the control of a decision-maker. The standard components of an MDP are expressed as a 4-tuple:

$$< S, A, P, R, \Upsilon >$$

Where:

- S is the state space: a set of all possible states in the environment $s \in S$.
- A is the action space: a set of all possible actions the agent can take $a \in A$.
- P is the state transition probability function: $P(s'|s, a)$ defines the probability of transitioning to state s' from state s by taking action a.
- R is the reward function: $R(\mathrm{s}, \mathrm{a})$ gives the immediate reward received after transitioning from state s via action a.
- Υ is the discount factor: a value between 0 and 1 that discounts future rewards, reflecting the value of immediate rewards over distant ones.

It should be noted that in the RL for Circularity framework, the state is represented as a vector (observation), which is a filtered representation of the state that encapsulates sufficient information about the environment to allow effective decision-making by the agent. This is further explained in sections 3.3.1 and 3.4.2.

The MDP provides a formal framework for modeling the interaction between an agent and its environment through states, actions, and rewards. The goal in an MDP is to find an optimal policy π^* that maximizes the expected sum of discounted rewards, known as the return, defined by

$$\sum_{t=0}^{t=\infty} \gamma^t R\ (\mathrm{s_t}, a_t, s_{t+1})$$

Where $\mathrm{s_t}, a_t, s_{t+1}$ is the state, action, and future state at any given time-step, respectively (Sutton & Barto, 1998).

3.1.2 Policy gradient methods

In the RL for circularity method, we use a Policy Gradient method to optimize the policy directly π by adjusting the policy parameters θ in the direction that maximizes the expected return (a good truss). This is achieved by using gradient ascent on the expected return, typically expressed as:

$$\nabla_\theta J(\theta) = E_t[log\pi_\theta(a_t|s_t) * A_t]$$

Where $log\pi_\theta(a_t|s_t)$ is the log probability of taking that action at that state, and A_t is the advantage. If $A > 0$, this action is better than the other actions possible at that state (*Advantage Actor Critic (A2C)*, 2024). However, to address two key challenges with normal policy gradient methods, the inefficiency of data usage and excessive policy updates that can lead to performance degradation, the RL for Circularity method uses an enhanced policy gradient method, known as Proximal Policy Optimization (Schulman et al., 2017).

Proximal Policy Optimization (PPO) (Schulman et al., 2017)

PPO modifies the policy gradient objective to include a clipping mechanism that limits the update size, making the learning process more stable. The objective prevents the updated policy from deviating too far from the current policy, which is crucial for maintaining stable learning dynamics. It uses a clipped surrogate objective, defined as:

$$L^{CLIP}(\theta) = \mathbb{E}_t[\min(r_t(\theta)A_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t)]$$

Where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{old}(a_t|s_t)}$ is the probability ratio of the new policy to the old policy, A_t is the advantage estimate and ϵ is a hyperparameter typically set around 0.1 to 0.3. PPO has two advantages:

1. Unlike standard policy gradient methods, which typically update the policy from each batch of data only once, PPO reuses each batch of data for multiple epochs of minibatch updates. This approach significantly improves sample efficiency.
2. Actor-Critic Architecture: PPO employs an actor-critic architecture where the 'actor' updates the policy based on the clipped objective, and the 'critic' estimates the value function, helping

to stabilize policy evaluation. The actor controls how our agent behaves (policy-based method), and the critic measures how effective the action taken is (value-based method)(*Advantage Actor Critic (A2C)*, 2024).

These advantages are particularly beneficial for our problem because they allow the agent to handle the complex action and state spaces involved in the generative design of trusses. Using MDP and PPO for inventory-constrained generative truss design enables a robust generative design approach that optimizes the decision-making process in a complex environment characterized by the variety and unpredictable nature of working with heterogeneous inventories from the bottom up. As established in the introduction, there are nearly infinite potential states that the agent can encounter.

3.1.3 Training

During training, PPO alternates between sampling data through interaction with the environment and optimizing the clipped surrogate objective by stochastic gradient ascent, updating both the policy (actor) and the value function (critic). In this thesis, the state is calculated in our custom-implemented environment. Here, each state-action-reward sequence is collected and used to update the policy and value function via PPO. The actor and critic are modeled as Deep Neural Networks, which learn to approximate the optimal policy and value functions, respectively. Deep learning, a subset of neural networks, consists of multilayered neural nets. The distinguishing feature of a deep learning algorithm is that the number of layers between the input and output layer in its neural network must be at least three (Kavlakoglu, 2023).

3.2 This thesis' implementation

For this thesis, RL solves the problem of finding a sequence of bars that assembles a truss whose top chord approximates a design gesture curve, given a curve and a list of inventory lengths. To formalize this problem in the language of reinforcement learning, we need to define the action, observation, and reward.

In our RL loop (Figure 15), the agent takes an action (e.g., placing the next two bars on this bar), and the environment returns an observation (e.g., the position of the new bars) and a reward (a number indicating how closely the new triangle approximates the gesture). The agent learns to improve its actions based on the rewards and observations with the objective of maximizing the expected cumulative reward.

At each timestep of an episode, the agent's action is to choose two available bars and attach them to the truss, selecting an edge of the previously constructed triangle to attach to. An episode terminates when: (1) the closest node to the gesture curve is further than 0.5m; (2) the constructed triangle intersects with the rest of the truss; (3) the constructed triangle intersects a defined avoid-region above the curve; or (4) the termination criterion, further described in section 0 is achieved.

The methodology extends the work carried out by Wibranek et al., 2021 and Bapst et al., 2019, Implementing Wibranek's agent-environment socket communication and Bapst's object-centric alternative to absolute actions, to support learning compositional behaviors, which are more invariant of the location in the scene, which we term relative actions. With relative actions, the agent takes actions in a reference frame relative to the previously constructed triangle in the scene.

3.3 Agent

The agent is the active component in the framework, performing actions within the truss environment, observing its progress, and receiving rewards for actions that contribute to the goal of assembling a truss. Through trial and error, the agent learns an optimal policy, enabling the truss design. The efficacy of the agent in solving the task is determined by several key elements: the observation space, the policy, the learning algorithm, and the action space, all of which have been tailored specifically for the task of assembling trusses from reused materials. The exact definitions of these parameters are outlined in subsequent sections.

3.3.1 Observation vector

The observation received at every timestep is a 26-dimensional vector with continuous values ranging between -1 and 1. These values provide structured information crucial for understanding the current state of the inventory and the ongoing design, such as the available lengths and the placement of the last bars relative to the target curve. The observation space is segmented into four parts:

1. Parameter Observation (1D): Represents the current position along the gesture curve.
2. Target Gesture Observation (6D): Coordinates of three leading points on the curve ahead of the agent, indicating the future direction.
3. Current Triangle Observation (8D): Coordinates of the ends of the two edges forming the current triangle.
4. Inventory Observation (11D): Values representing the quantity of inventory elements sorted into 11 bins, each bin corresponding to a range of lengths.

5. $$O_t = \begin{bmatrix} o_1 \\ o_2 \\ o_3 \\ \vdots \\ o_{26} \end{bmatrix}, where\ o_i \in [-1,1]\ for\ all\ i = 1,2,\dots 26$$

These observations are computed relative to the previously placed triangle, integrating local information about the curve's future direction by including three leading points. This structure make the agent able to calculate its value function, without overfitting to the current gesture curve. Conversely, limiting observations to a single point limits the agent's ability to look ahead. Experiments conducted by (Wibranek et al., 2021) have shown that including three points strikes an optimal balance for

solving the curve-guided sequential assembly. Adjustments to this parameter may be considered in other scenarios to enhance performance.

3.3.2 Action tuple

In this project, a three-dimensional continuous action tuple was defined, allowing the agent to decide at each timestep which two elements to select from the inventory and their attachment points in the design. The action tuple is represented as:

$$A_t = < x, y, k >$$

Where:

- x and y are continuous coordinates denoting the location of the cursor within the local reference frame, each ranging from -1 to 1.
- k is a binary value quantized from a continuous range, effectively $k \in \{0,1\}$, specifies which edge of the previously constructed triangle to build upon.

Although the problem is combinatorial, involving ultimately discrete actions, the choice was made to utilize continuous actions to better manage the changing inventory size. These actions are interpreted as discrete decisions, selecting the placement of the triangle vertex and deciding on the attachment edge for the two selected elements.

3.3.3 Policy

The policy computes the actions At, given an input observation Ot. A Deep Machine Learning policy based on an artificial neural network with fully connected hidden layers, known as a Multi-layer Perceptron (MLP), is utilized. This network comprises three hidden layers, each containing 128 neurons. Many variations of the MLP architecture were experimentally tested, varying in width and depth. The configuration described was found to perform the best, balancing effective learning and reasonable training time. It is sufficiently complex to grasp the intricacies of the combinatorial task without overfitting or excessive training durations.

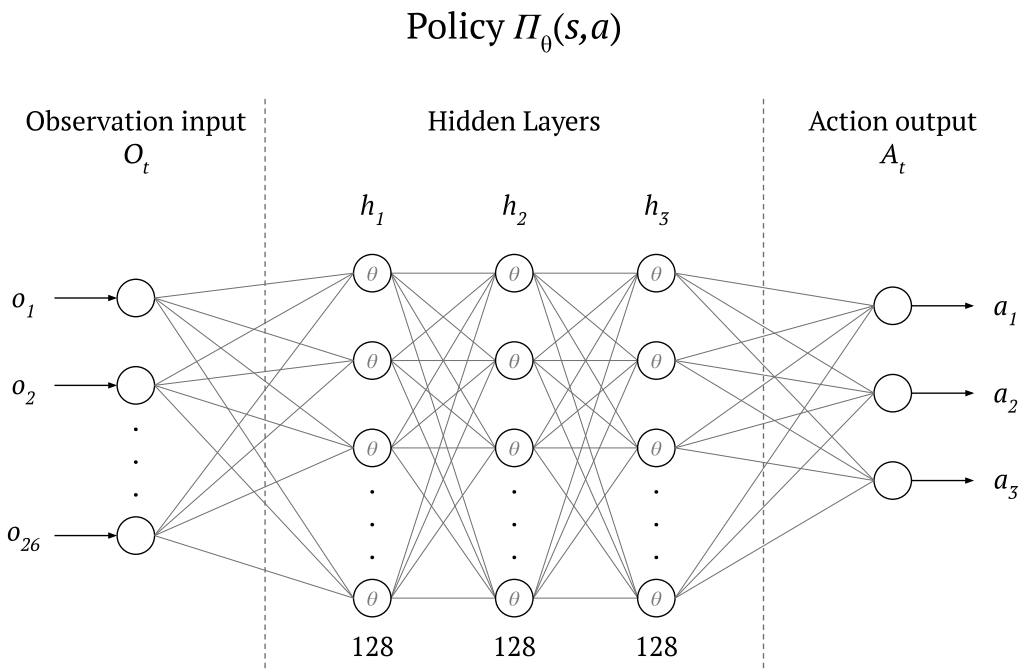


Figure 16: Deep neural net policy architecture.

Proximal Policy Optimization (PPO)

In this research, PPO on is employed to balance exploration and exploitation and manage the learning process effectively. PPO, one of the many algorithms used in RL, was chosen for its robustness and versatility across diverse problems. It is particularly effective in handling continuous actions and observations, making it well-suited for research purposes. This algorithm's strength lies in its ability to maintain a stable learning trajectory without substantial fluctuations in performance, a common issue in other policy gradient methods (Schulman et al., 2017).

3.4 Environment

The environment is the geometric engine where the agent's actions are interpreted at each step. In the environment, the next inventory elements are selected and attached sequentially to the truss the agent is designing. Essentially, it operates as a geometric script that performs operations and outputs the consequences in terms of updated observations, reset flags, and rewards. While the RL agent is a general structure that is largely task-agnostic (receiving an observation vector and outputting action tuples), the environment is task-specific and tailored to interpret the agent's actions in the context of truss construction, providing structured observations. In applied RL research, setting up the problem and especially developing the environment is crucial. Thus, formulating the environment is a key contribution of this research. The environment maintains an inventory of the available elements for the agent and tracks where used elements are incorporated into the truss. It also keeps track of the geometric layout of the truss. In this way, it oversees training, updates procedural inventories and gesture curves, and calculates rewards, resets, and observations.

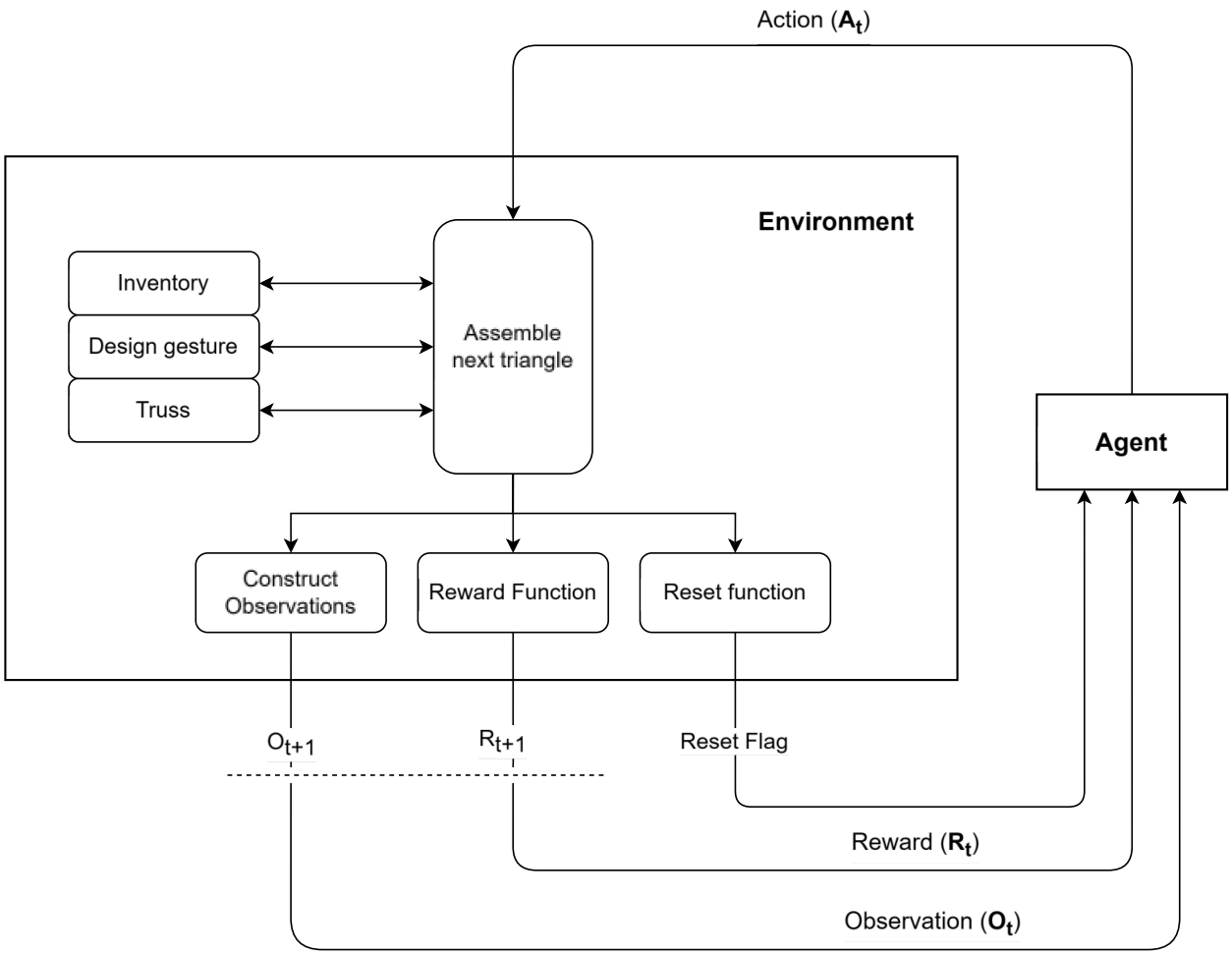


Figure 17: The RL environment flow chart.

3.4.1 Action interpretation for geometric assembly

Designing a planar truss can, in most cases, be broken down to the act of constructing a series of triangles. The geometric assembly function is responsible for interpreting A_t into a triangle to be added to the truss, using two new inventory elements for each triangle added. This function has been crucial to the model's success. Through experiments, it was established that to enable effective learning, as much processing as possible should be outsourced to the environment. This allows the agent to focus on the critical task of determining which triangle can be built next, given the current state of the truss, the gesture curve, and the inventory. To achieve this, the processing has been isolated to deciding which edge of the previous triangle to build upon and determining the location of a single decision point. This point marks the desired location for the vertex of the next triangle relative to the previous triangle in the truss. This implicit action mimics how a human might describe their design process in a similar situation: “It looks like I can build a triangle roughly this size given what is left in the stock. That would move me in the right direction.” Explicitly choosing the exact two elements from the inventory, coordinates in space for their placement, and the correct angles to form a valid

triangle involves unnecessary processing for solving a relatively trivial geometric problem that can be easily calculated using Euclidean geometry and simple trigonometry.

The system takes as input the action tuple, the previously added triangle, the target curve, the currently available inventory lengths, and outputs the next added triangle and an updated inventory. Below is the pseudo-code for the truss assembly, with references to Figure 18, Figure 19, Figure 20, and Figure 21 and symbols within these.

```
Algorithm: Truss Assembly
Input: CurrentTriangle, ActionTuple <x, y, k>, TargetCurve, InventoryLengths
Output: UpdatedInventory, NewTriangle

1: Map (x, y) from ActionTuple from [-1, 1] to [-2, 2]
2: Draw decision point (x, y) // P
3: if i = 0 then: // refer to Figure 18
4:     Map TargetCurve to local coordinate system // A
5:     Find closest point on local TargetCurve // T
6:     Calculate distance from T to O // d
7:     Find the element in the inventory with length closest to d // m
8:     Orient inventory member to align with TargetCurve // M
9:     Remove selected element m from Inventory
10: else: // Refer to Figure 20 and Figure 21
11:    Move local coordinate system to current closest point on gesture curve to area
       centroid of last constructed triangle // Figure 19
12:    Use Action[k] to determine which previous element to continue from // e
13:    if len(P, O) < maximum inventory length then: // refer to Figure 20
14:        Use the distance from P to the two ends on the selected truss element to
           find the closest length matches in the Inventory // m1 and m2
15:        Orient m1 and m2 to selected truss bar e, using trigonometric function 1
           // M1 and M2
16:    else: // refer to Figure 21
17:        Construct a triangle pointing in the direction of P, using the longest
           available inventory element for its largest edge
18:        Use the lengths d1 and d2 in the constructed triangle to find the two in
           -ventory elements m1 and m2 with the closest lengths
19:        Orient m1 and m2 to selected truss bar e, using trigonometric function 1
           // M1 and M2
20: Output the updated Inventory without selected elements
21: Output the new Triangle
```

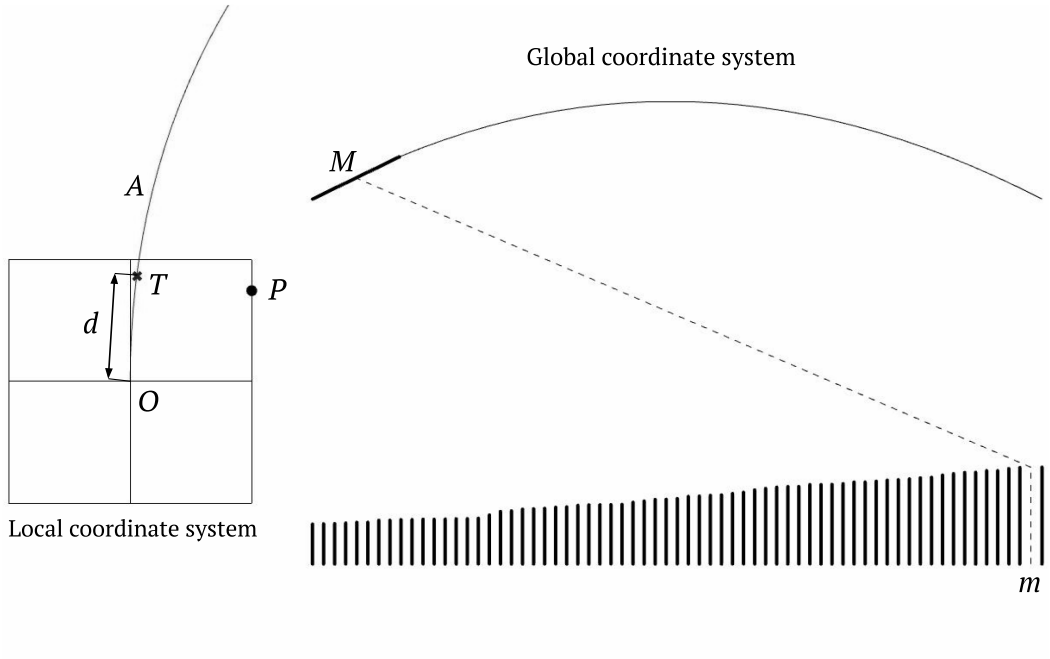


Figure 18: First element selection procedure.

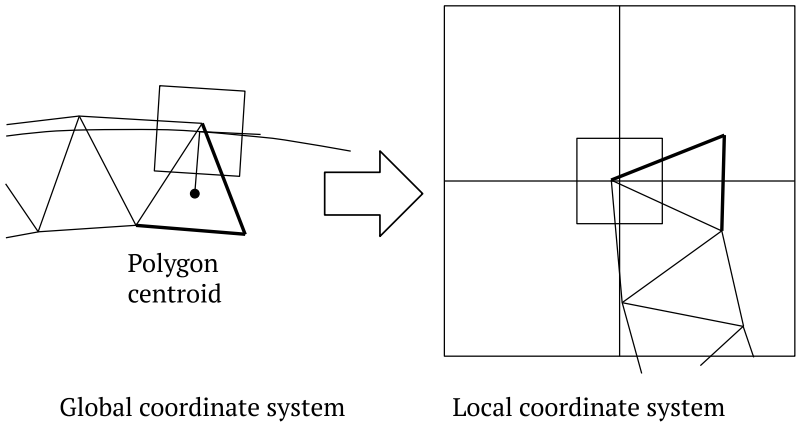


Figure 19: Setting the local coordinate system to the current position on the gesture curve.

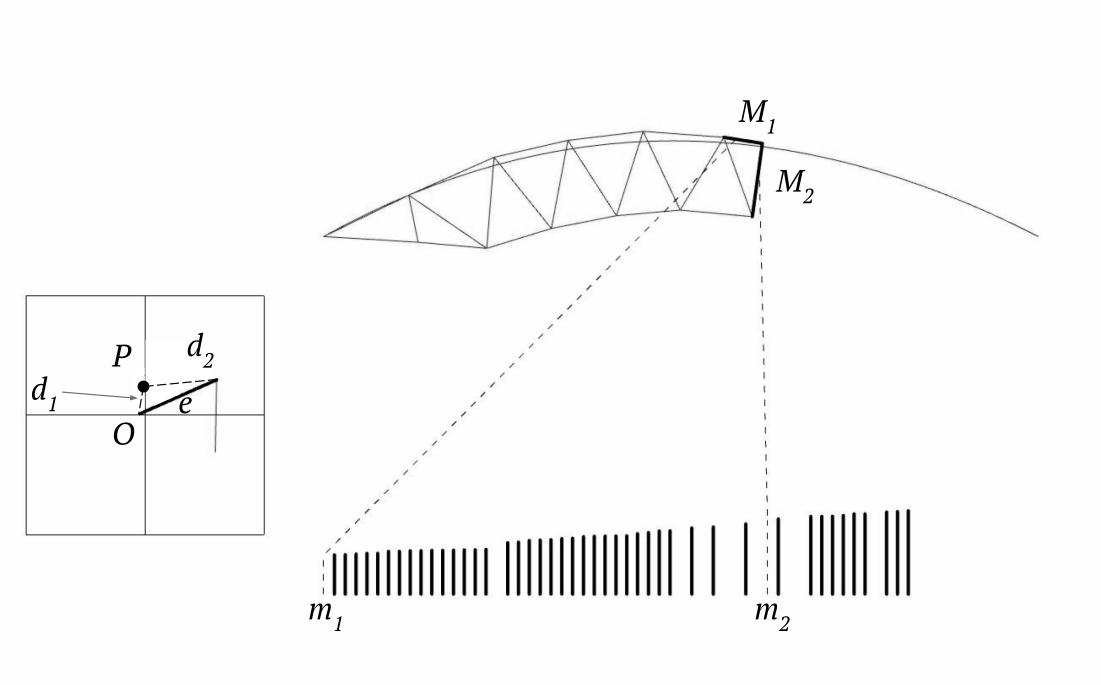
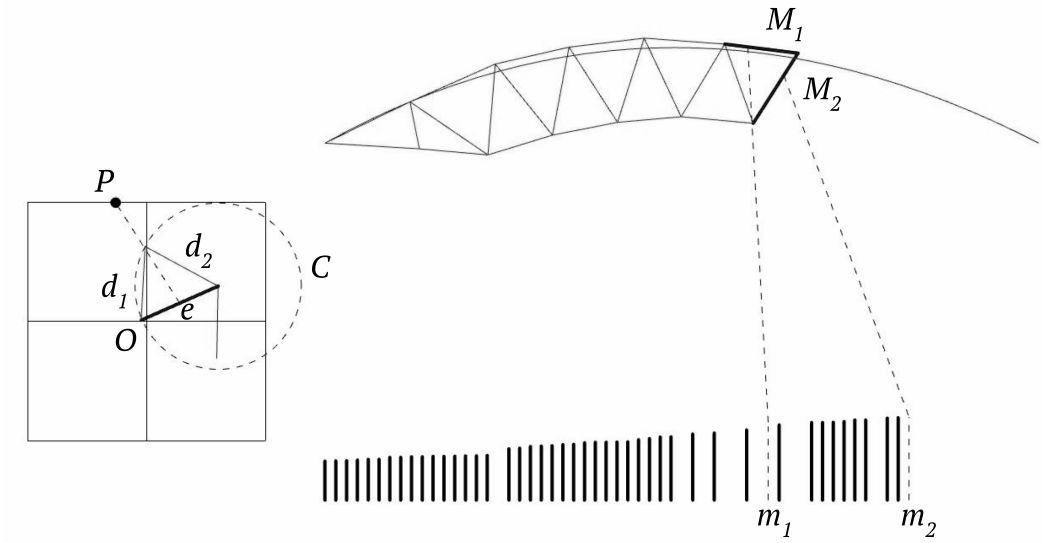


Figure 20: Selecting the inventory elements for case 1 (Decision point within inventory length range).



-

Figure 21: Selecting inventory elements for case 2 (Decision point outside inventory length range).

3.4.2 Observation function

As mentioned in 3.3.1, the observation function sends a tuple of values between -1 and 1 to the agent, conveying structured crucial information about the inventory, the previously added triangle, and the gesture curve. These are constructed in the environment as follows, with reference to Figure 22:

1. Parameter Observation (1D): The target gesture curve is reparametrized to a domain [-1 to 1] domain. The area centroid of the newly added triangle is then used to find the parameter value (t) closest to the area centroid. This represents how far the agent has progressed along the gesture.
2. Target Gesture Observation (6D): x, y coordinates of three leading points (a, b, c, d) on the curve ahead of the agent indicates the future design direction.
3. Current Triangle Observation (8D): x, y coordinates of the endpoints (d,e,f,g) of the two edges forming the current triangle.
4. Inventory Observation (11D): Values representing the quantity of inventory elements sorted into 11 bins, each bin corresponding to a range of lengths as shown on Figure 22.

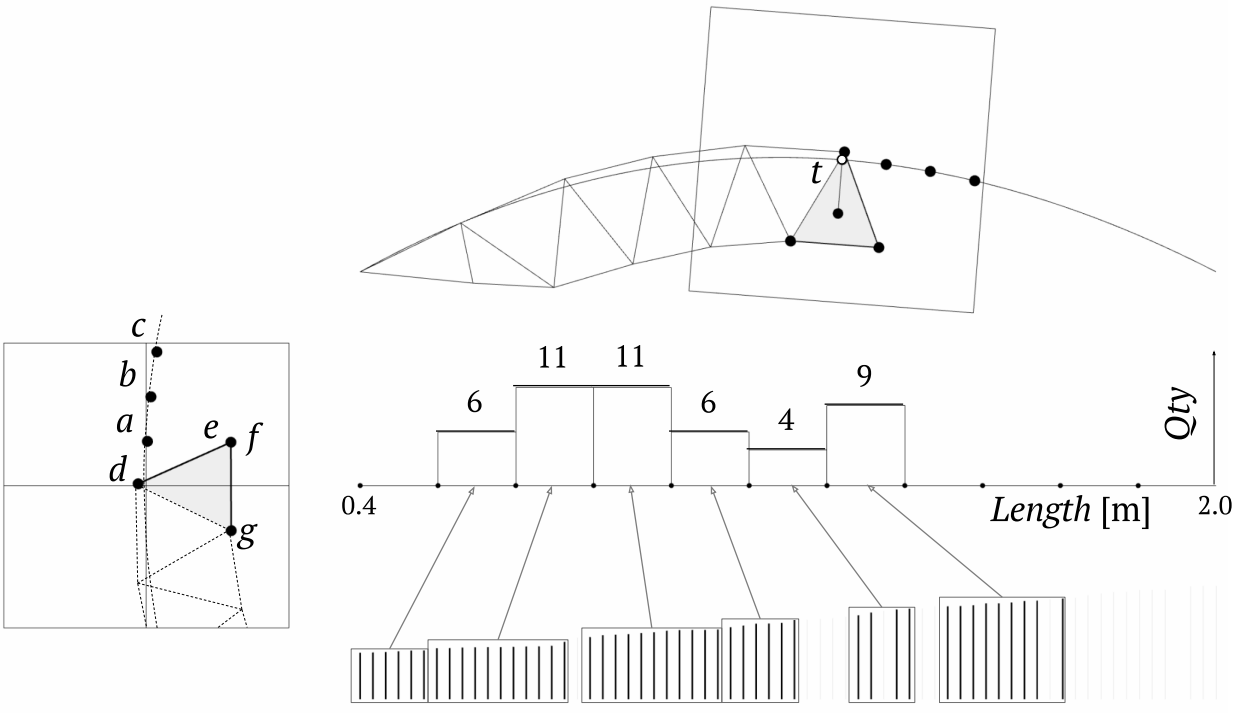


Figure 22: Observation construction in the environment. Top right shows the global coordinates of t, the target gesture, and the current triangle coordinates. Left, shows these mapped to the local coordinate system, and the middle shows a bar-chart of the binned inventory observation.

3.4.3 Reward function and reset flags

The reward function and reset flag are crucial to the agent's operation, as the agent learns a policy that maximizes the accumulated reward of each episode. Many attempts were made with different formulations, but ultimately, the presented formulation allowed the agent to learn in the fewest training timesteps how to sequentially design the best trusses in terms of fit and stiffness while adhering to the inventory.

The reward function encourages the agent to assemble a truss with a top chord as close as possible to the target gesture and to construct a complete truss that minimizes maximum displacement. In particular, the reward is maximized when the top chord perfectly follows the gesture curve and completes a stiff truss. These two properties are essential to the function of a roof truss. If the agent deviates too much from the curve, the episode ends, and the agent must start anew in a reset environment. It was found that rather than rewarding the agent only at the end of a successfully completed truss—a method known as a sparse reward scheme—the agent trained more effectively when rewarded at each step that contributed to a projected better truss. Additionally, it was found that instead of punishing the agent with negative rewards for actions that did not contribute to a better truss, such as building too far away from the target gesture, the agent learned better if such conditions simply reset the environment to the initial state s_0. The reason is that the agent would start terminating the truss early as a sub-optimal but reliable way to avoid any potential negative reward down the line. One exception is the case where the agent chooses two short inventory elements that cannot result in a triangle. We express Reward at each step as:

$$R_{step} = R_{bead} + R_{end} + R_{struct}$$

Where:

1. R_{bead} : Is calculated by first distributing circles (beads) along the target with a radius r, at δ intervals. For every intersection between the newly added triangle and a bead, a step reward of +1 is added unless the added triangle intersects with the avoid region (reset condition 1)
2. R_{end} : If a vertex of the added triangle intersects with a circle placed at the end of the target gesture c, a step reward of +2 is added.
3. R_{struct} : At the end of an episode, if a vertex on the final triangle added is within the end region e, a Finite Element Analysis if calculated to find the maximum displacement in centimeters v_{max} of one of the nodes or element mid-points of the truss for a distributed load case. The displacement is then scaled according to the average length of the available inventory λ, to assign a fair relative structural reward to trusses built with inventories of smaller lengths, which naturally result in greater displacement. The scaled displacement is then

subtracted from 5. Hence, the maximum structural reward for the stiffest theoretical truss is 5. The full structural reward can be expressed as:

$$R_{struct} = max\ (0, 5 - C\lambda v_{max})$$

Where λ is the average length of the available inventory, v_{max} is the maximum displacement as calculated by FEM, and C is a scaling factor chosen to scale $C\lambda v_{max}$ to a [0,5] range for the expected inventories inventories the agent will work with.

Parameters for the Finite Element Method (FEM)

The FEM analysis assumes a simply supported boundary condition with a distributed load w applied only at the top chord nodes, except for the end nodes which are reserved for supports. The main criterion for the FEM formulation is that the range of displacements is relatively stable for the problem setup, meaning that the trusses we roughly expect, as defined by the gesture lengths and inventory quantities, will result in reliable and stable displacement calculations. In other words, calculated internal stresses should be within the chosen material's yield stress σ_y. For this research, these values were chosen manually through trial and error:

- w = 10kN/m
- $CroSec$ = ASTM A500-03a 60x3.2 (Cold-Formed Carbon Steel Structural Round Tubing)
- Pin connections at all nodes

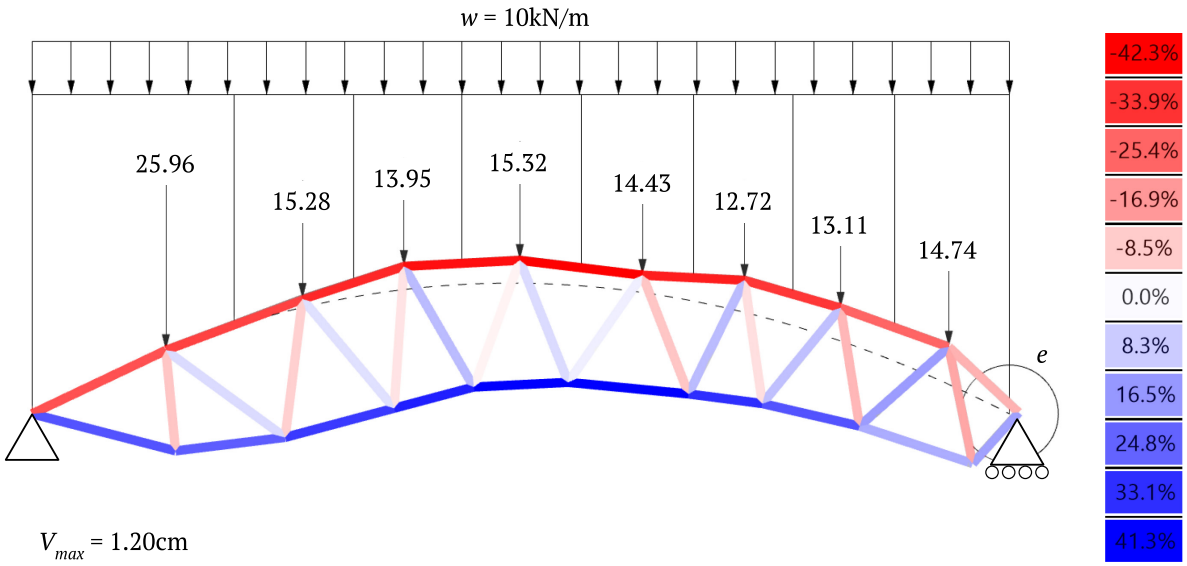


Figure 23: Structural analysis setup. Arrows indicate the applied point loads in kN. e is the region the truss has to reach to activate the structural analysis. The bars are color-coded to show stress-utilization.

Reset conditions:

Reset if:

1. Added triangle intersects with avoid region as defined by the area above and to the sides of the gesture curve, offset by distance η
2. The closest vertex of the added triangle is more than a defined distance ε from the target gesture.
3. If a vertex of the added triangle intersects a circle with radius = η placed at the end support
4. If the added triangle intersects with the truss

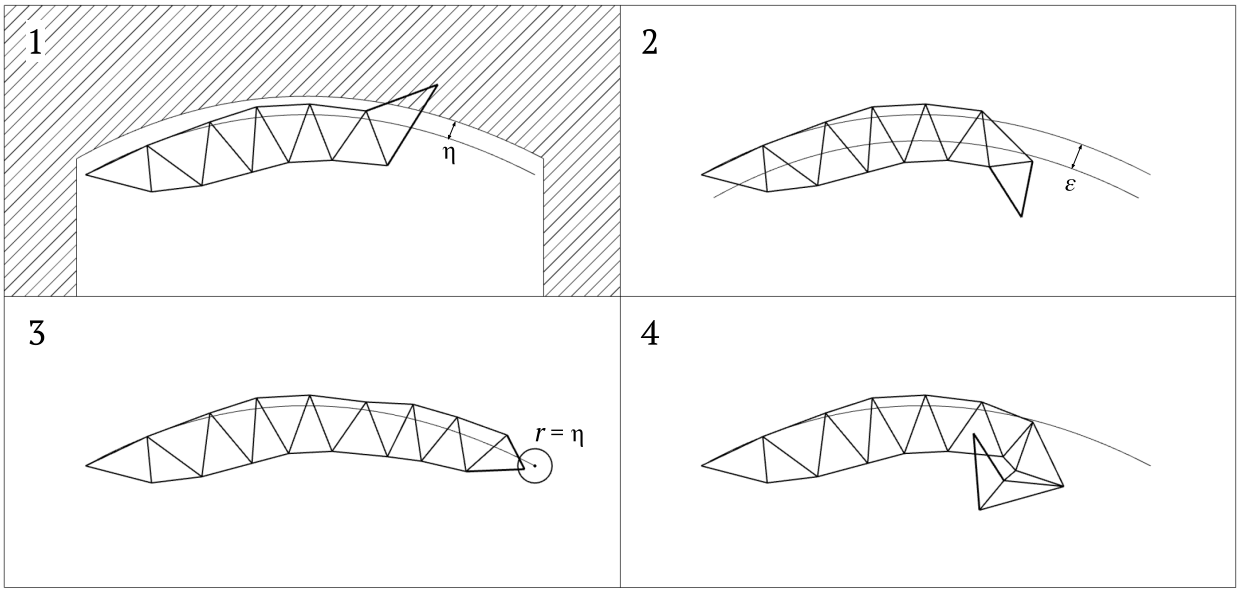


Figure 24: Reset conditions. 1: intersecting avoid region, 2: veered off path, 3: reached the end, 4: self-intersection.

3.5 Training

Here are the parameters related to training the agent.

3.5.1 Procedural inventories and targets

To train the Multi-Layer Perceptron PPO agent, the environment is set to procedurally generate new random inventories and target curves. A new random inventory is generated every 150 steps, using the current step as the input seed. This seed is used to generate three random numbers: (1) range start [0.2m – 3m], (2) range end [0.2m – 3m], (3) inventory quantity [20 – 300]. For the target curves, a new seed is generated every 50 steps. This seed is used to generate two random numbers: one number chooses between three gesture categories: straight, curved, and kinked; the second random

number [0 – 1] adjusts a variable on the gesture curve to determine the curvature or kink angle depending on the category. Updating the seed every 50 and 150 steps allows the agent multiple attempts for each unique inventory and gesture combination, which provides more stable gradients for the policy updates.

3.5.2 Curriculum learning

To improve training, it was found that progressively increasing the complexity of the task during the entire training sessions is crucial. This approach is referred to as curriculum learning. The curriculum steps implemented to train the agent were:

1. 0 – 100k steps: Static straight gesture with the same inventory
2. 50k – 100k: Static straight gesture with procedural inventory
3. 100k - 400k: Procedural gestures and procedural inventories

3.5.3 Hyperparameters

Several hyperparameters are used in the provided training configuration for an agent, each with a specific purpose to optimize learning.

- n_steps = 1024: This parameter defines the number of steps the agent takes before updating the model. A larger value like 1024 provides more experiences per update.
- n_epochs = 10: Specifies how many times the learning algorithm will reuse the data collected during the n_steps. Revisiting this data multiple times allows for more thorough learning from each set of experiences. In this thesis setup, the environment takes relatively long to compute compared to the policy update, so it makes sense to use the collected data multiple times.
- learning_rate = 0.003: The rate at which the model's weights are updated during training. 0.003 is a standard moderate rate.
- gamma = 0.99: This is the discount factor for future rewards. A high value is chosen to prioritize long-term gains, encouraging the agent to focus on actions that lead to a better completed truss at the end of the episode.
- batch_size = 256: This parameter determines how many experiences are processed together. A batch size of 256 is large enough to provide a good estimate of the gradient.

3.6 Summary of key method parameters

Through experiments, the author determined that these parameters in the implementation are highly influential to the result:

- Local Observations and Actions: Initially, an "absolute action and observation" approach was implemented, where, for example, the agent might choose to place the next triangle at world coordinates (7.43, 3.54). However, learning absolute actions scales poorly as the size of the environment grows because the agent must effectively re-learn its construction policy at every location (Bapst, 2019). Similar to Bapst et al. (2019), the results show that agents that use object-centric actions outperform setups that use less structured representations and generalize better beyond their training when asked to reason about larger scenes.
- Decision Isolation: Initially, the truss assembly problem was posed broadly, granting the agent complete freedom to position elements on the canvas. This setup aimed to enable learning how bars connect to form triangles and, subsequently, trusses. However, the agent could not learn past the simple construction of a triangle using three bars. After trial and error, it became apparent that it was inefficient to use training resources for the agent to learn basic geometric operations, which could be easily handled through conventional mathematical geometry. Consequently, the problem was reformulated to automate all deterministic operations, like calculating angles between bars to ensure they form a triangle and sum to π radians. This reformulation focuses the agent's learning on decisions requiring RL "intelligence" that are not easily solved through analytical or numerical methods. Such decisions include choosing which edge of a previously added triangle to extend and determining the approximate direction of the new triangle based on an observation of the state. Directions are marked by “decision points” on the canvas, as detailed in Section 3.4.1. This strategy conserves training time and computational resources by bypassing the need to learn elementary geometric operations.
- Structural Reward: A structural reward enabled the agent to assemble better structural trusses and also converged better by constraining the design-space, essentially excluding actions that lead to a good geometric fit but produce stress points, very high aspect-ratio triangles, or small kinks in the design.

3.7 How to evaluate success

With the goal of generating viable roof trusses from a constrained heterogeneous material inventory, the method sets the following qualitative and quantitative objectives:

1. **Good**: The model aims to generate 'good' roof trusses, as defined by:
    a. Fit: The top chord of the generated truss should closely fit the target gesture.
    b. Displacement: The model should minimize the displacement of the truss under a uniform distributed load.
2. **Reliability/robustness**: The method should consistently generate good trusses across different inventories and design gestures.
3. **Speed**: The model should generate trusses quickly, enabling users to rapidly receive results and interactively adjust inputs.

In summary, the objective is to achieve good results quickly, most of the time.

3.8 Inference

Once trained, the model is loaded into a design environment and presented with the user's design problem. The use case envisioned is as follows:

1. The user defines a list of all the available inventory lengths. This can be extracted from a database by a demolition/disassembly company, manually measured and input, or automatically generated through computer-vision scanning, such as flying a drone over beams laid on the ground. An inventory can be any quantity as long as the range of lengths of the individual elements is reflected in the procedurally generated training inventories.

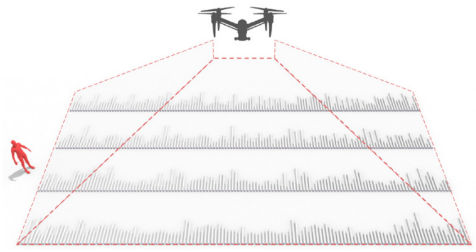


Figure 25: Inventory scanned by drone.

2. The user draws one or more planar target gestures in the design environment, indicating the overall intent of the roof to be assembled. These curves can take any form or length as long as they are planar respectively.

Figure 26: Example user input design gestures.

3. The inventory and gestures are sent to the model, which live-aggregates inventory elements along the gesture, attempting to create a structural truss following the gesture.
4. The inventory and gestures are sent to the model, which live-aggregates inventory elements along the gesture, attempting to create a structural truss following the gesture.
5. If the design is unsatisfactory, the user has multiple options:
    a. Changing the gesture curves. This can be done manually or parametrically to optimize the design to fit the inventory better automatically.
    b. Changing the inventory: The agent's attempt at assembling trusses to meet the design goals might reveal that adding a few more elements of a certain size could improve the design. The user can input these and immediately see how the truss layout changes in response. These new elements can be bought, manufactured, or found if needed.
    c. Rerun the Model: A final option is to rerun the model non-deterministically. PPO operates by sampling from a probability distribution to decide on actions, allowing for non-deterministic behavior and the production of varied outcomes. Users can leverage this characteristic by conducting multiple runs of the model, thereby generating a range of potentially effective trusses from which they can select based on personal preferences.
    d. The design is detailed and fabricated manually or through automatic detailing digital fabrication. A helpful guide to locating and placing the physical inventory in the structure is the implementation of Augmented Reality to aid the fabricator in merging the digital inventory-to-design match with the physical pieces.

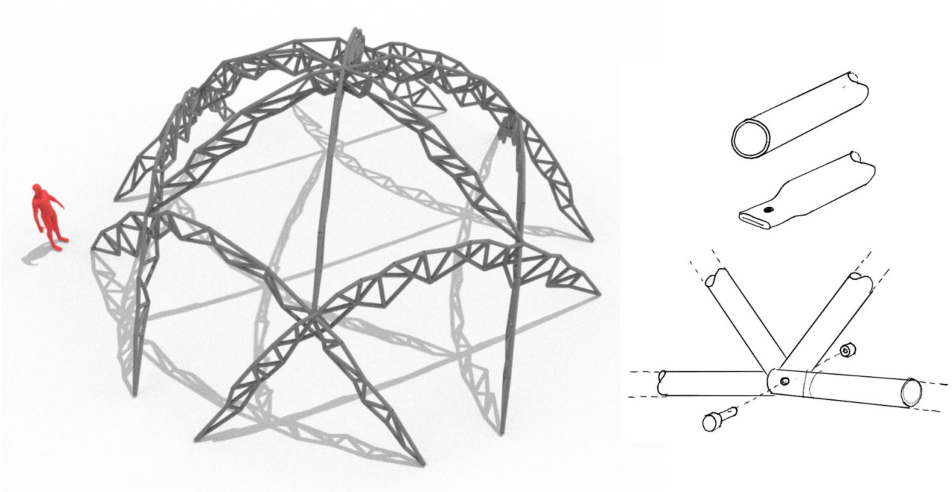


Figure 27: Example trussed dome design with suggested node detail.

3.9 Implementation

To solve the above-defined RL problem, the stable-baselines3 library (Raffin et al., 2021), which is the open-source standard in the RL field, is employed. A socket-based interface integrates Grasshopper as an RL environment with stable-baselines3, named gh_gym, developed by Wibranek et al. (2021). The environment is set up in the Grasshopper3d plugin for Rhino7 with the following custom add-ins: Anemone for loop support and Karamba3d for Finite Element Method (FEM) analysis.

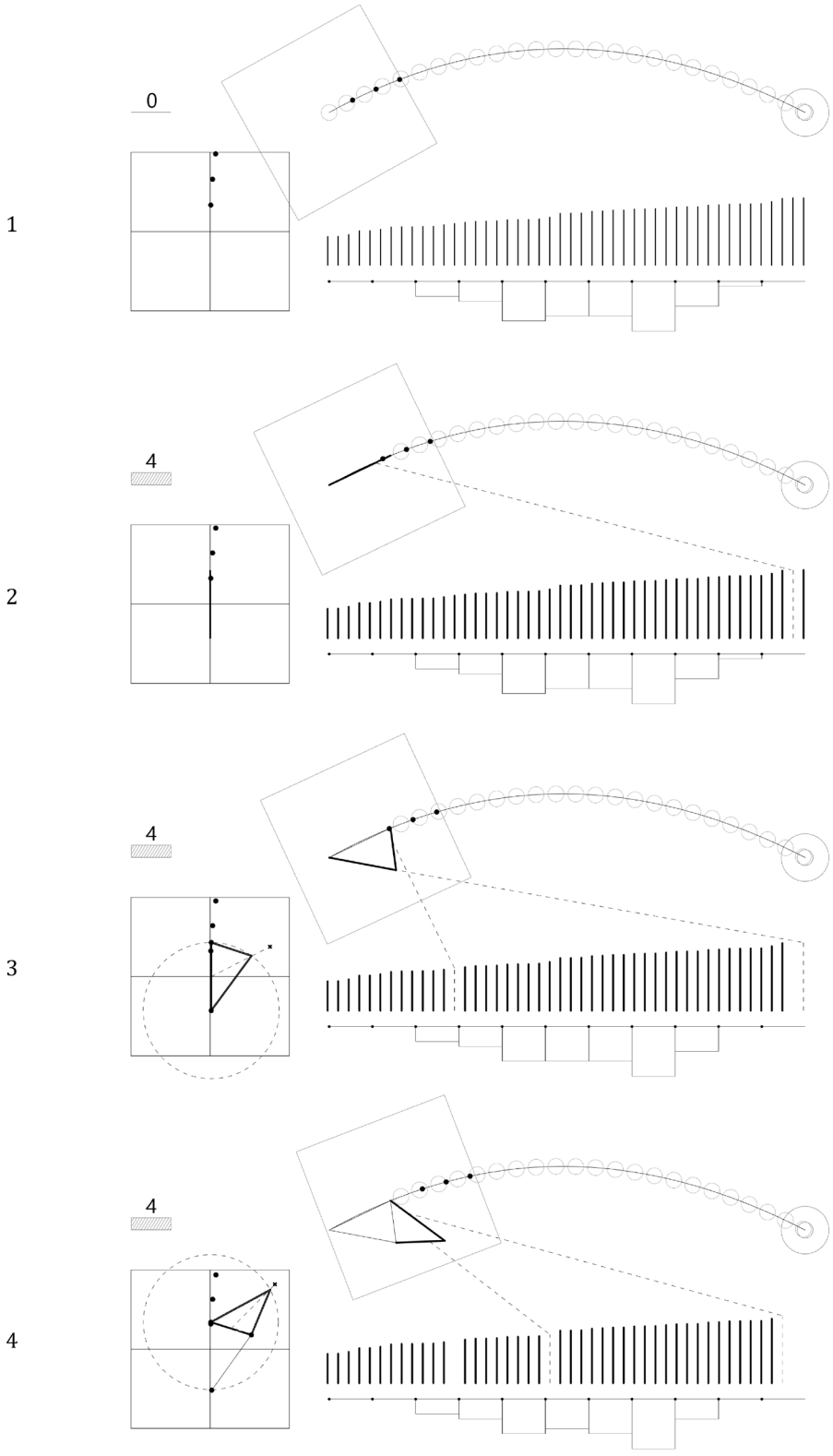


Figure 28: Example truss RL generation (steps 1 - 4). The top left bar indicates the total accumulated reward.

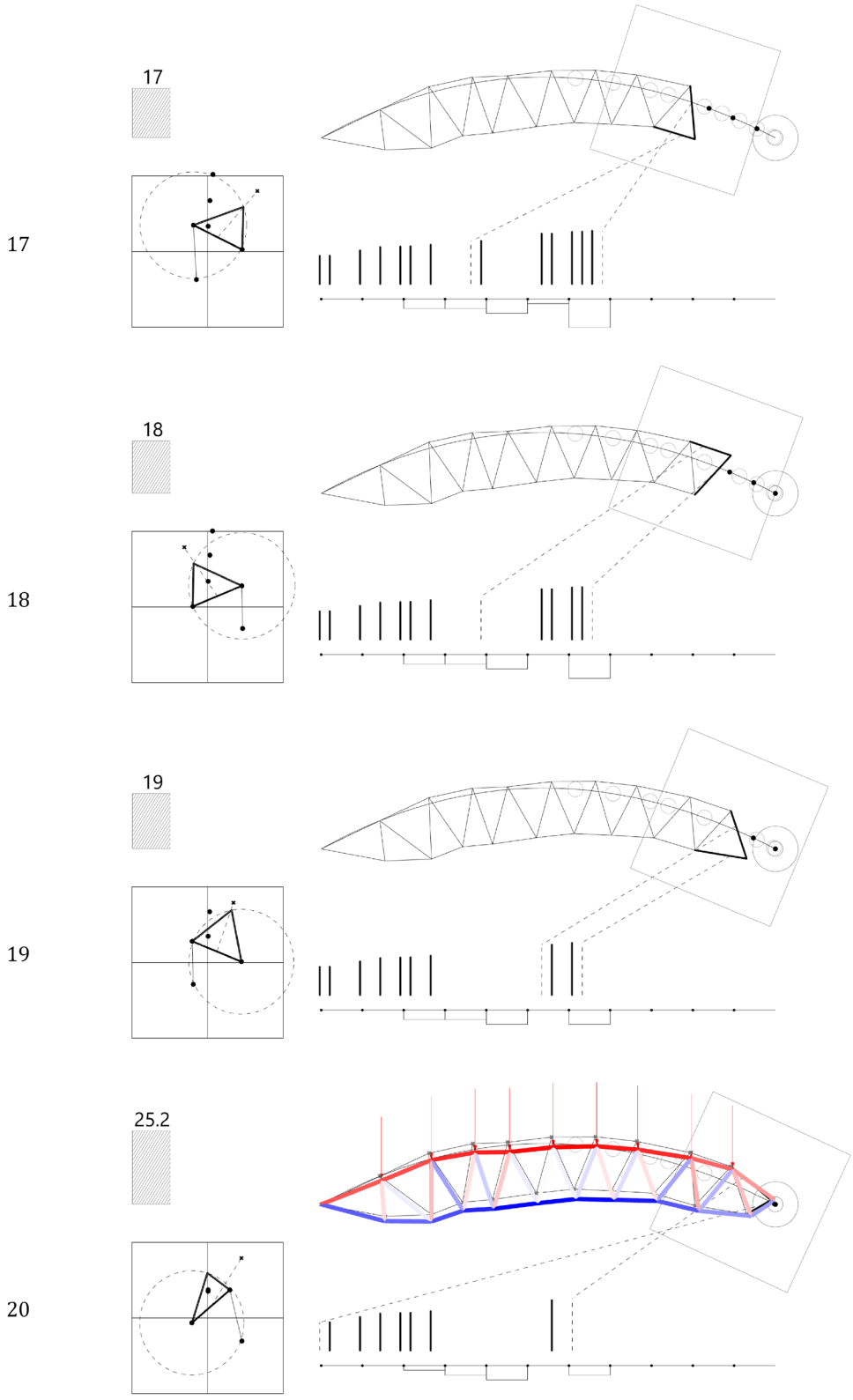


Figure 29: Example truss RL generation (steps 17 - 20). The top left bar indicates the total accumulated reward.

4 Results

This section evaluates the method, using the formulated RL agent to construct structural planar trusses following different gesture curves with varying inventories. Furthermore, a design case study is presented on a realistic design problem involving an inventory from a deconstructed transmission tower and assembling it into the roof truss of a tennis-hall to demonstrate the practical design usage of the RL approach.

4.1 Trained models

Two final agents were trained for these results. Both agents used an identical setup, except for the reward function: one agent was trained with a structural reward, and the other with the structural reward turned off. In the following sections, these are referred to as the Structural Agent and the Non-Structural Agent, respectively. For some of the results, other agents are selected from the database of test agents trained throughout the development of the approach. These are referred to as Experimental Agents and are sometimes used to show the diversity of results that can be achieved when experimenting with different neural architectures, hyperparameters, and environment setups. The final two agents were trained using the training setup as detailed in Section 3.5 (Training) on 10-meter-long procedurally generated design gesture curves and procedurally generated material inventories with a size range of [0.2m, 3m] and a quantity range of [20, 300], with a distributed load of 10kN/m.

All models were trained on a laptop with a 3.30GHz AMD Ryzen 9 5900HS CPU, an NVIDIA GTX 1650 4GB GPU, and 16 GB ram, for 400,000 steps (14hr training time). The mean episode reward plots show the full reward history, as the model learns an optimal policy. Note that the Non-Structural Agent converges faster, while the Structural Agent continues to learn for more time-steps. This indicates that more training is needed for an agent to learn the complex decision-making required to meet structural goals.

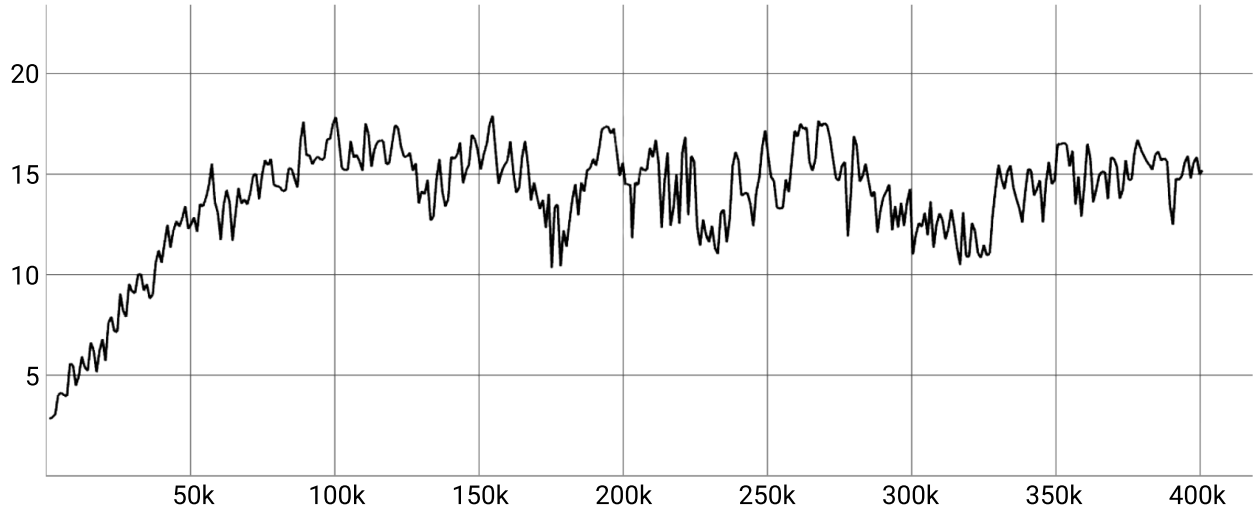


Figure 30: Mean Episode Reward training history for the Non-structural Agent.

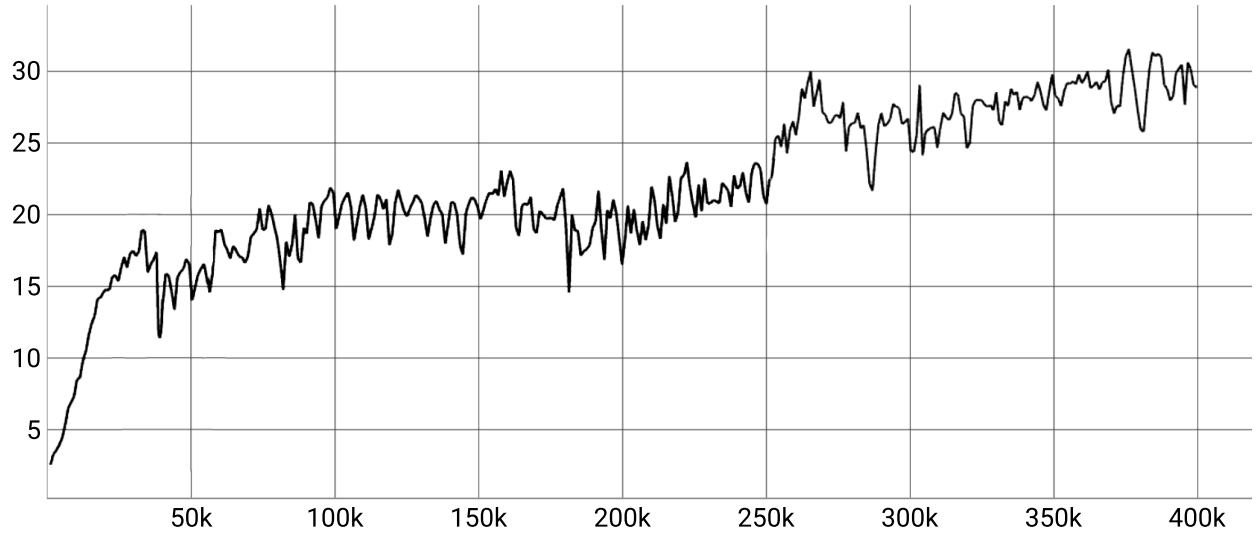


Figure 31: Mean Episode Reward training history for the Structural Agent.

4.2 Design Space sampling

As established in (3.7 - How to evaluate success), the properties of the method we are interested in are fit, displacement, reliability/robustness, and speed. To evaluate these properties, we define them as dependent variables in a standardized experiment setup.

4.3 Experimental setup

Setup

1. A linear span with length L = 12m
2. Distributed load w = 7.5 kN/m
   This is based on a design 50psf design load-case as suggested by Professor John Ochsendorf as a general rule-of-thumb for the gravity load-case of a roof (DL = 20psf, LL = 30psf). If we assume that every 12m truss is spaced 3m (10ft) apart we get:
   $$50psf * 10ft = 500lb/ft = 7.28kN/m \approx 7.5kN/m$$
3. Cross-section C*roSec* = ASTM A500-03a, Round HSS 60x3.2 with the following properties:

Table 1 Cross-sectional properties used in results.

Diameter [mm]	Thickness [mm]	Area [cm²]	Tensile strength [MPa]	Yield strength [MPa]
60.0	3.1	5.16	310	228

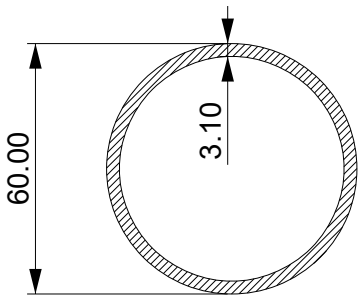


Figure 32: Selected experiment cross section.

Independent variables

1. Q - Inventory Quantity, where $Q \in [25, 250]$
2. S - Inventory Spread, centered around 1.2 m, with a spread range $S \in [0.1, 1.9]$

For a given Q (quantity) and S (spread), a random inventory is generated such that the lengths of the items, L, are defined by:

$$L_i \in [1.2 - \frac{S}{2}, 1.2 + \frac{S}{2}]$$

Where i ranges from 1 to Q, representing the index of each item in the inventory and each L_i is randomly selected within the specified range.

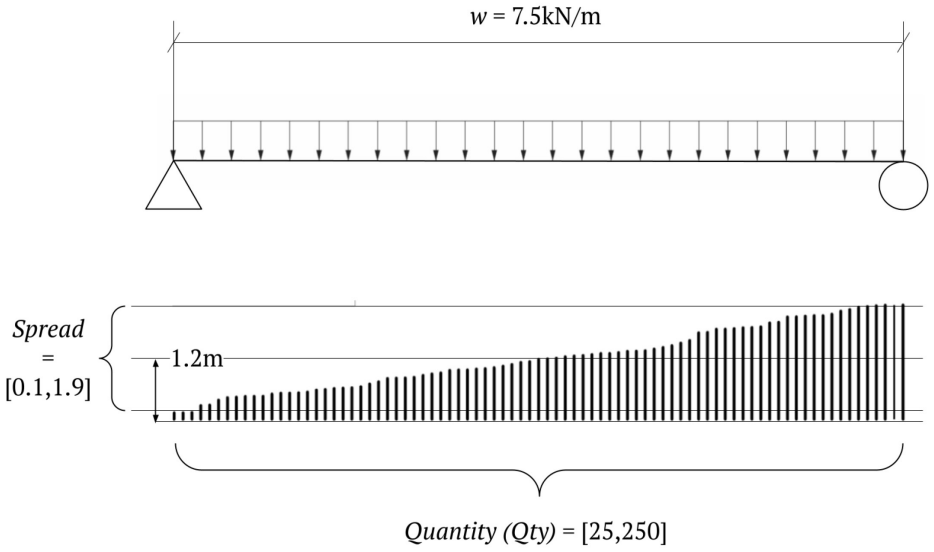


Figure 33: Experimental setup and independent variables.

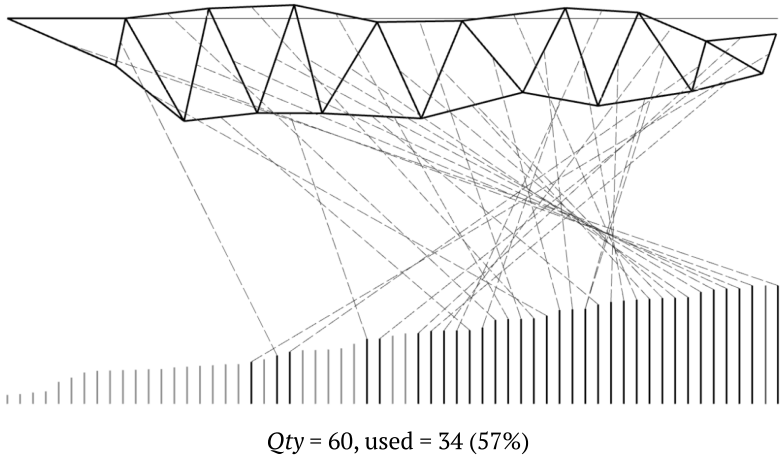


Figure 34: Example generated truss. Note that the truss is always sequentially assembled from left to right. Used inventory bars are highlighted with dashed lines indicating their placement in the truss layout. The percentage indicates how much of the available inventory is used in the truss.

Dependent variables

Once a truss is generated, such as the one in Figure 34, it is analyzed for geometric fit and structural performance. The dependent variables are:

1. Geometric fit: The geometric fit is assessed by defining 50 points evenly spaced along the target curve. P_i for $i = 1,2, \dots ,50$ for each point P_i, measure the vertical distance d_i from P_i to the truss' top chord. The fit score is calculated as the average of these distances:

$$\text{Fit} = \frac{1}{50}\sum_{i=1}^{50} d_i$$

2. Maximum displacement: Is calculated with FEM in Karamba3D, as specified the section 3.4.3 (Parameters for the Finite Element Method (FEM)).

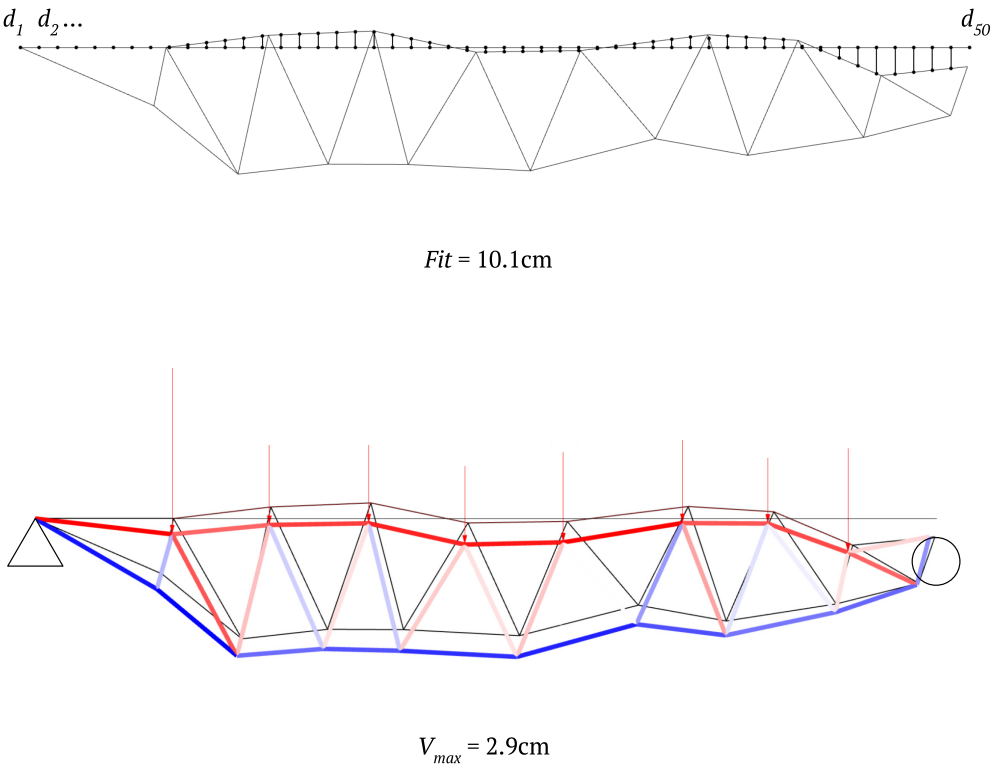


Figure 35: Top: Fit calculation, Bottom: max displacement calculation, Color scale indicates compressive (red) and tensile (blue) stress. The displacement visualization is scaled 20X.

4.3.1 Max displacement (Structural Agent)

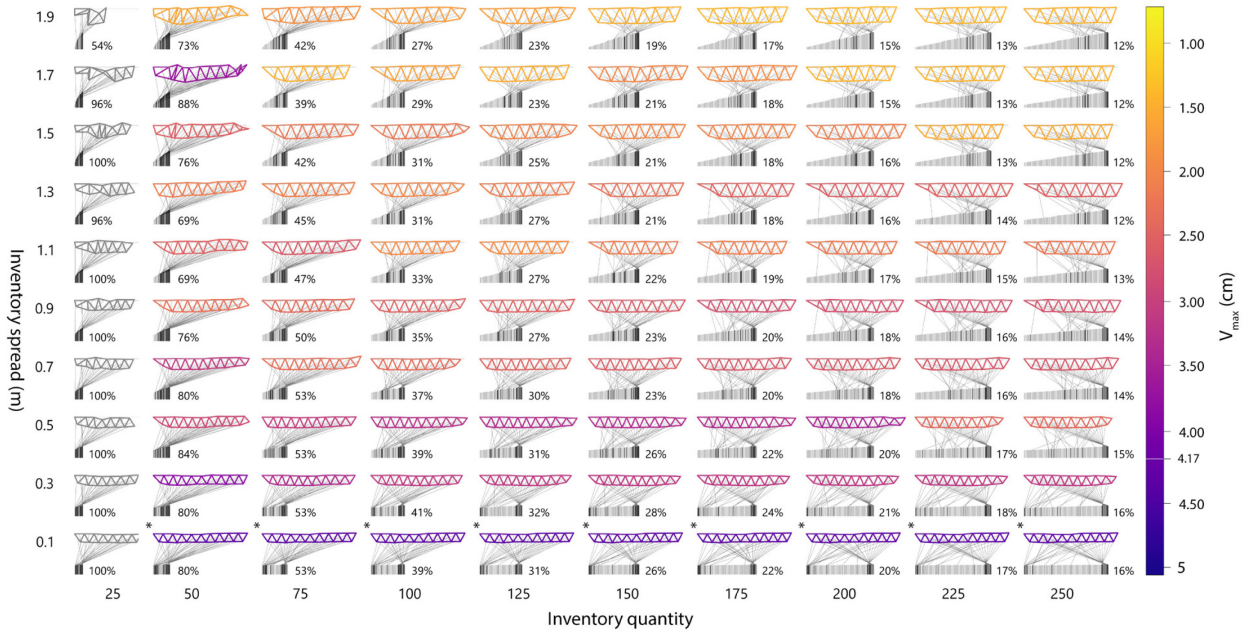


Figure 36: Effect of different inventories on maximum displacement, truss layout, and inventory utilization. Percentage signs indicate inventory use per truss. Asterisks denote instances where V_{max} exceeds L/240 = 4.17cm.

4.3.2 Geometric fit (Structural Agent)

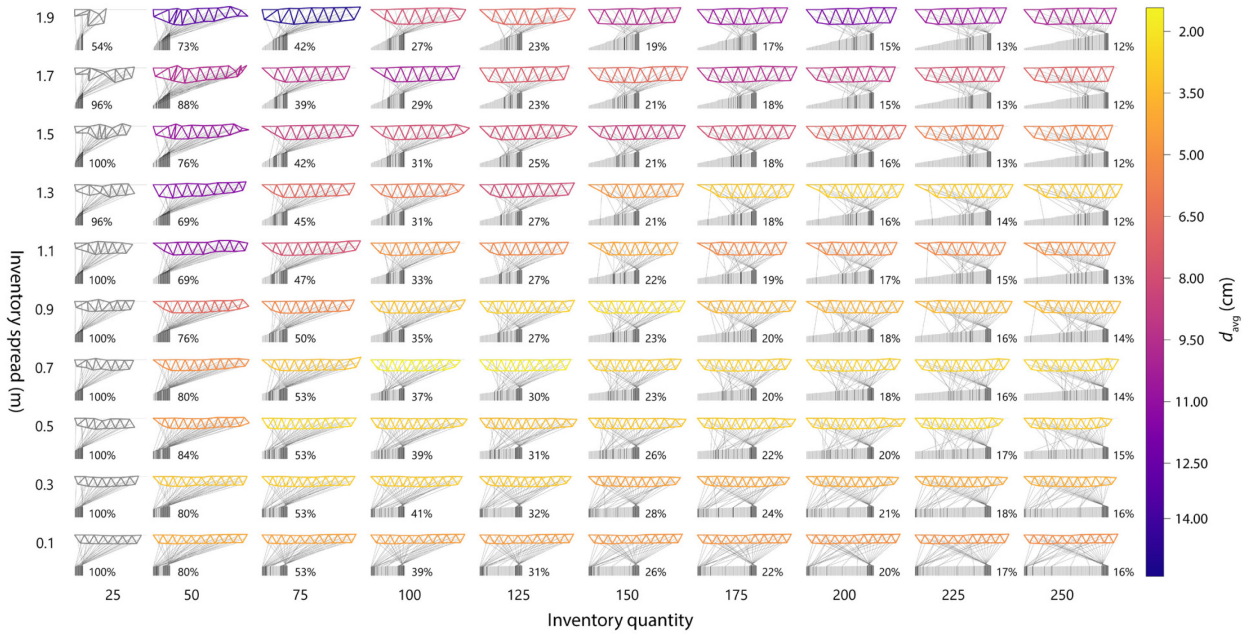


Figure 37: Effect of different inventories on geometric fit. The percentage signs indicate inventory use per truss.

Discussion

The results for maximum displacement and geometric fit show that the structural agent can generate trusses across various inventory spans and quantities. It can also be observed that, in general, the agent can reliably assemble good trusses using 50%-75% of an inventory. This suggests that for any inventory-constrained planar truss design with a heterogeneous inventory, a rule of thumb is that you need twice the inventory elements required for your design. Additionally, the data reveal that smaller spreads generally yield better geometric fits, while the best inventories for stiffness combine both a large spread and a large quantity. Furthermore, there is a tendency in the Structural Agent to preferentially use the longest elements available in the inventory, likely as a strategy to maximize the effective depth of the truss.

4.3.3 Model reliability/robustness

Using the same setup, we now sample the design space with 10 seeds per inventory to investigate how reliably the agent generates trusses when faced with slight variations of the same inventory type.

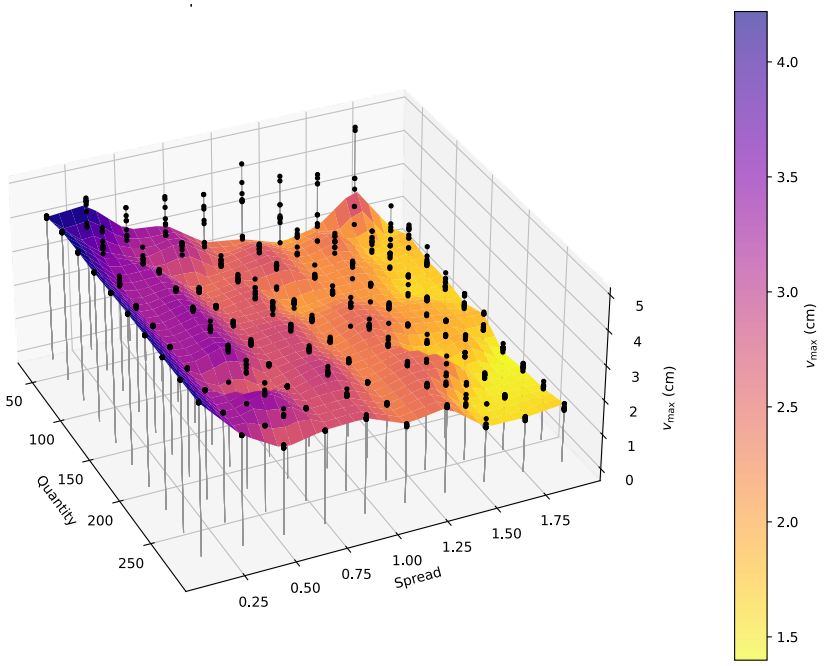


Figure 38 Variance of Maximum Displacement Across Different Inventories. The surface illustrates the average maximum displacement (V_{max}) as a function of inventory quantity and spread. For each combination of inventory parameters, 10 random inventories are generated, and their corresponding V_{max} values are depicted as points above and below the surface.

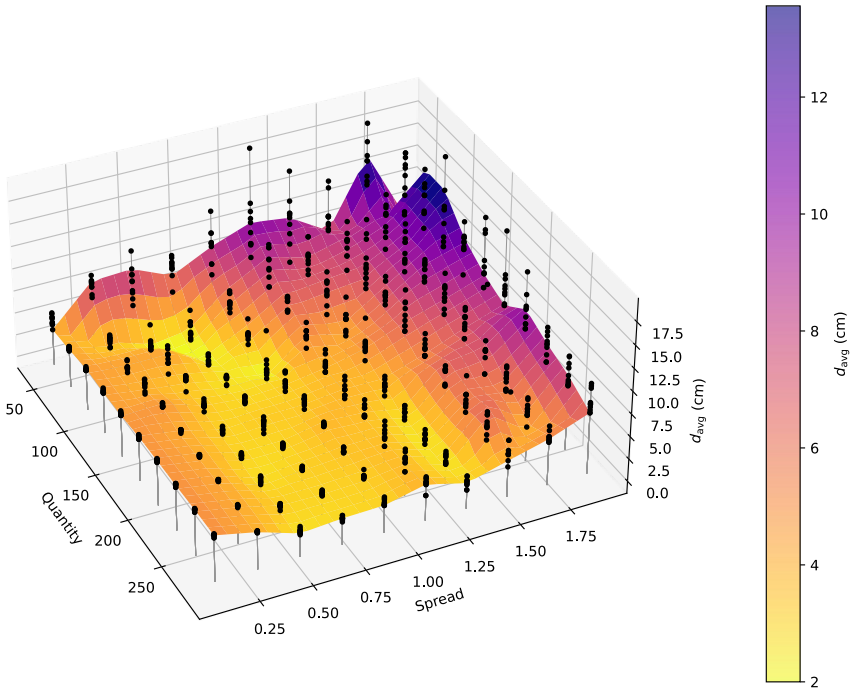


Figure 39: Variance of average fit to target curve across different inventories. The surface illustrates the average fit (d_{avg}) as a function of inventory quantity and spread. For each combination of inventory parameters, 10 random inventories are generated, and their corresponding d_{avg} values are depicted as points above and below the surface.

Discussion

The reliability/robustness plots, in addition to visualizing the fit and displacement results from earlier as a 3D surface, show that as inventory spread increases, so does the variance in both displacement and fit, with the most pronounced variance occurring when inventory quantity is smallest. The variance is most pronounced in the fit results (Figure 39). Overall, the Structural Agent is very reliable/robust, and outputs results within a 10% performance range for almost all combinations of quantity and spread.

4.3.4 Geometric Diversity Results

Six inventories and six target gestures are defined to evaluate how the method performs when sampling across different wild inventory types and wild target types. These are chosen to show the greatest variety and accurately represent real-world scenarios. The inventories include (1) Self-similar members, (2) A split inventory with many elements of one length and many of another, (3) A short heterogeneous inventory with moderate spread, (4) A large heterogeneous inventory with a wide spread, (5) A split heterogeneous inventory, and (6) A stepped inventory with 10 different lengths and 10 copies of each length.

These six inventories are then cross evaluated with the six target gestures: (1) Positive curvature, (2) Straight, (3) Negative curvature, (4) Slope to straight, (5) Mixed curvature, and (6) Standard pitched roof.

Multiple trained agents are sampled to demonstrate geometric diversity. In the first result, illustrated in Figure 40, different Experimental Agents are sampled for each truss, and visually interesting results are picked. In Figure 41 are trusses generated by the Structural and Non-Structural Agent.

Layout diversity result 1: Experimental Agents

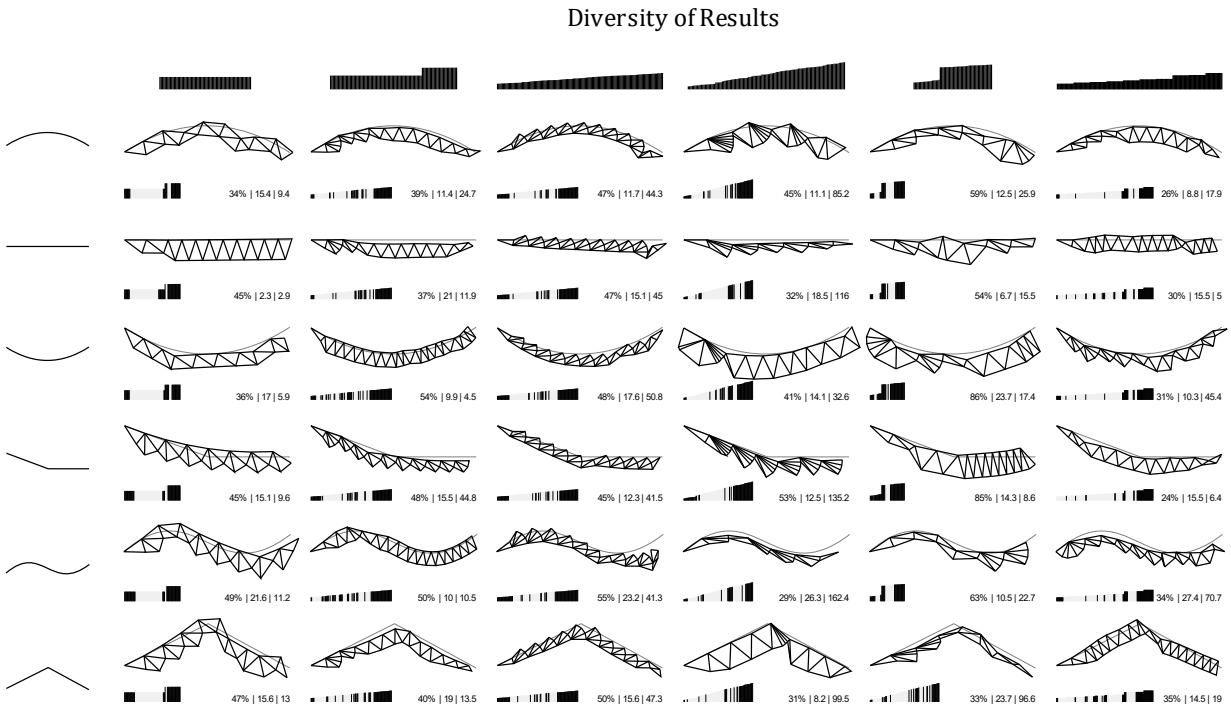


Figure 40: Diversity of truss-layouts for experimental agents sampled across different inventory categories (x axis), and design gestures (y-axis). The individual trusses are chosen manually by hand-picking Experimental Agents trained throughout the development of the method.

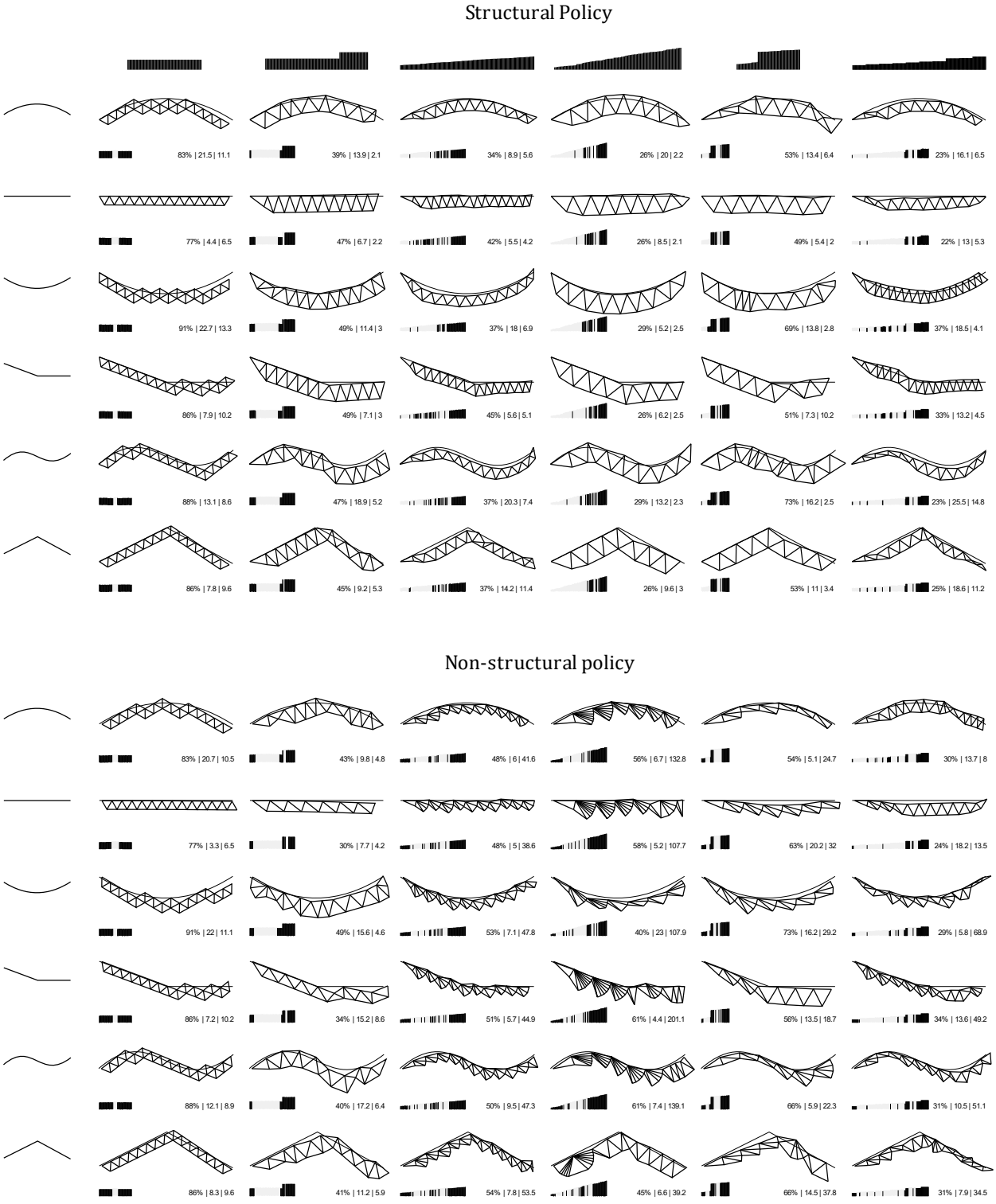


Figure 41: Trusses generated from different inventories and target gestures. Every column has the same inventory and every row the same gesture. Below each truss is its inventory, with highlighted elements used. There are also three numbers showing the (i) inventory percentage used, (ii) fit-score (iii) max displacement score. The top array is generated using the Structural Agent, the bottom array uses the Non-Structural Agent.

4.3.5 Discussion

A couple of observations can be made from the layout diversity results. One is that the agents can handle a variety of design gestures and inventory types and will produce a result in most cases. Even if a generated truss is not desirable or functional, it is still, in most cases, valid as it approximates the design gesture, uses the inventory, and avoids self-intersections or choosing elements that would result in invalid triangles. A comparison between the Structural and Non-Structural agents reveals that the Structural Agent tends to produce more regular Warren-Truss-type layouts, while the Non-Structural Agent often uses a 'fanning' strategy to closely approximate the design gesture. An example of this can be seen in Figure 41, 3rd and 4th rows. The Non-Structural Agent achieves a better fit score; however, it tends to introduce small kinks in the trusses, leading to stress concentrations and hence large (poor) displacement scores. On the other hand, the Structural Agent performs worse in terms of fit but produces stiffer truss layouts, with a few exceptions, such as in column 5, row 4 of Figure 41. Adding the structural reward constrains the agent from producing similar-looking regular trusses, whereas omitting the structural reward opens up opportunities for greater diversity in design. This could be exploited if the designer's primary concerns are fit and inventory usage. One might argue that the Non-Structural and Experimental agents produce more visually expressive results. Another observation is that the Experimental agents, often terminated before training converges, exhibit the greatest diversity in results. As is apparent, an agent sometimes produces structurally sound but not structurally efficient trusses, such as in Figure 40, column 5, row 2. This is arguably acceptable because reused material can be considered "free." Another interpretation of this behavior is that it creates an ornament—an AI ornament.

4.4 Design case-study

To test the method on a realistic case study, the approach is applied to the Circular Design challenge of disassembling a transmission tower into its constituent angle profiles and using those to design an indoor tennis court without producing any off cuts. Transmission towers are common lattice structures that constitute a major part of the High-Voltage electricity infrastructure. In Denmark, many transmission towers are demolished and recycled in an effort to convert to underground electric cables. For instance, in a project planned for 2028, an 18 km long line of high-voltage transmission towers is being replaced with underground lines in Roskilde, Denmark (Energinet, 2023). What if the elements of transmission towers could be directly reused for new structures, thus avoiding the energy and carbon emissions associated with re-processing and re-melting the steel?

Figure 42: Transmission towers to be demolished in Roskilde, Denmark. (© Jens Dresling)

4.4.1 Transmission tower definition

lattice towers are commonly made from L-profiles that are bolted together. The geometry of such a tower is generously borrowed from (K. Lee & Mueller, 2024). Because the RL for Circularity method only works with a single cross-section at this stage of development, a simplifying assumption is made that all members in the tower have the same cross-section. A typical bracing angle section cross section is used (Albayrak & Morshid, 2020), with properties outlined in Table 2.

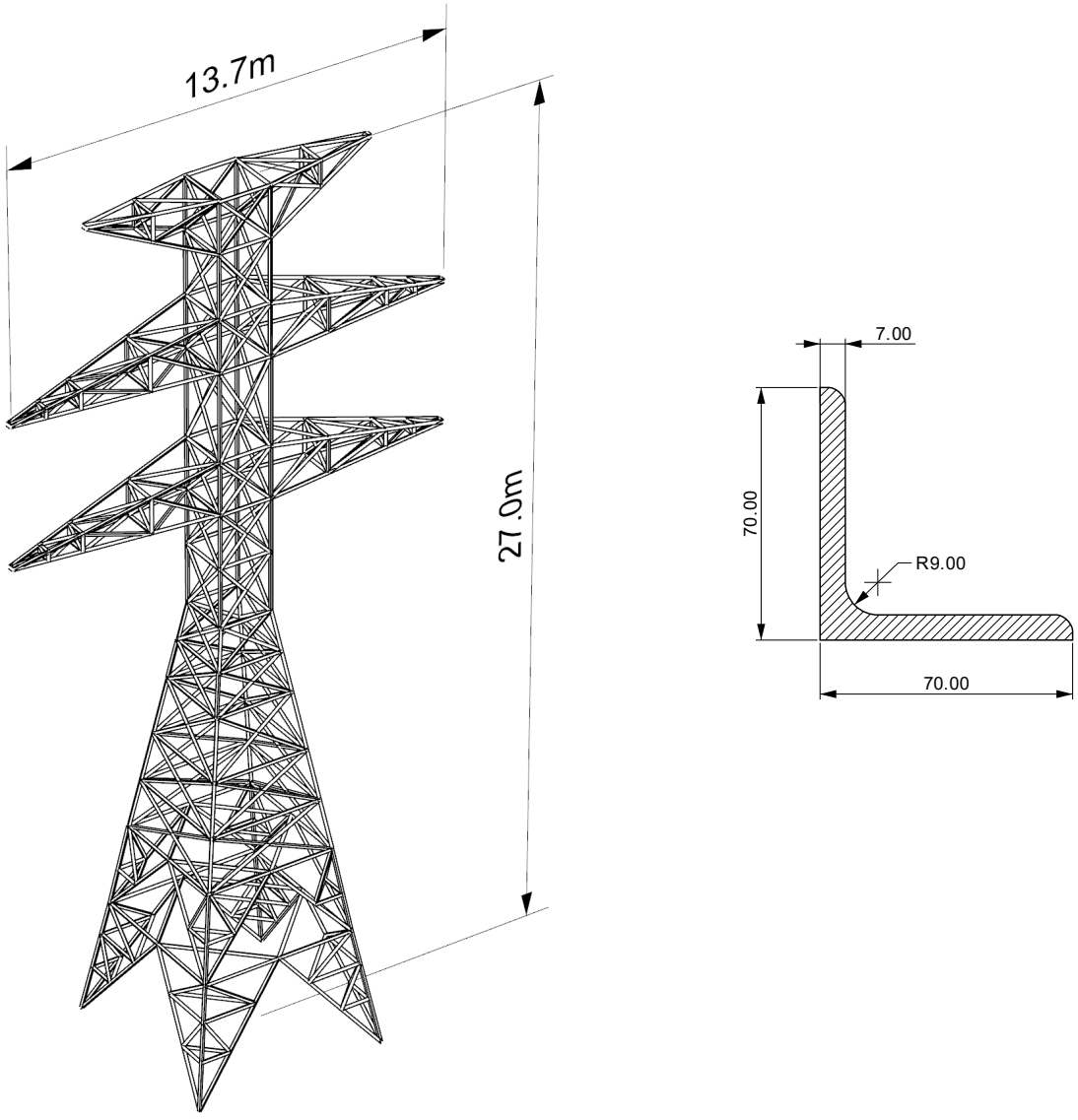


Figure 43: Transmission tower and assumed element cross section with dimensions in mm.

Table 2 Angle profile cross-sectional properties

Dimensions [mm]	Cross-sectional area [cm^2]	Material	Ultimate strength [MPa]	Yield strength [MPa]
L 70x70x7	9.397	S355	510	355

4.4.2 Inventory

Figure 44 shows how the transmission tower is disassembled. It is assumed that the disassembly sequence proceeds from top to bottom. As the tower is disassembled, the elements are labeled with Radio Frequency Identification tags and measured using a computer vision app to digitize the inventory on the fly quickly. Note that this process does not require any sorting of the elements. Depending on the state of the tower, some bolted connections might have corroded and become impossible to unscrew, necessitating the cutting loose of the element. This is simulated by subtracting a random number [0, 0.2m] from every element in the digital inventory. Long elements with a length greater than 4 meters are cut in half to simplify disassembly and packaging. Once measured and labeled, the steel is then packaged on pallets and shipped to temporary storage.

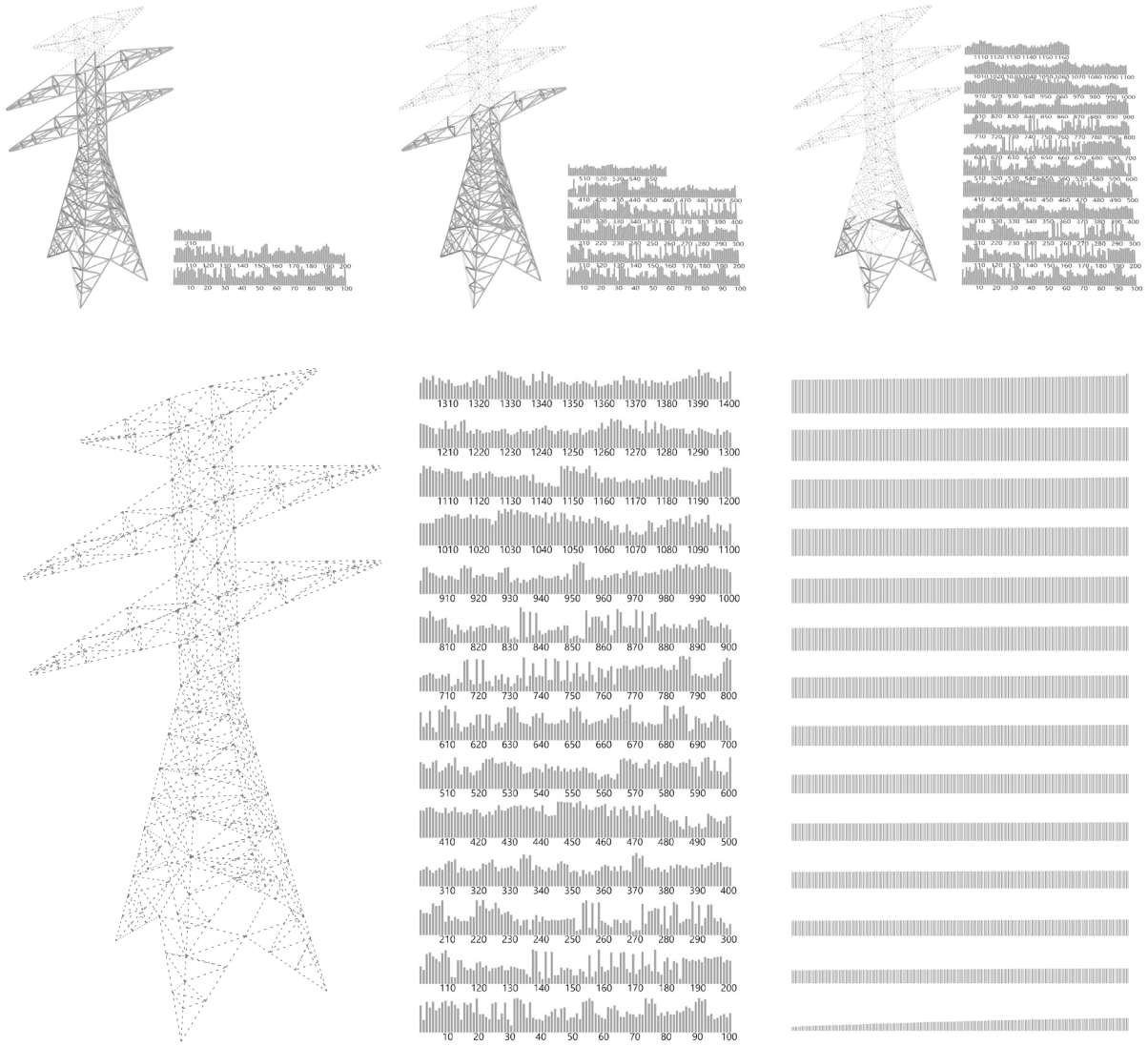


Figure 44: Disassembly of tower from top to bottom and the inventory it creates. The right array shows the lengths sorted digitally; the center array is the inventory indexed in order of disassembly.

4.4.3 Site & program

The proposed program is an indoor tennis court to be built on a site by Musicon a new “rebel neighborhood” in Roskilde, Denmark, near the decommissioned transmission tower.

Figure 45: Proposed tennis-hall site in Roskilde, source: Google Maps.

4.4.4 Design Gesture experiments

Unlike other methods, such as combinatorial matching (Cousin et al., 2023; Huang et al., 2021) or topology optimization (Brütting et al., 2019), design development can be immediately started by testing different gesture curves. A simple long-span warehouse-like roof with planar arch trusses is proposed. Testing agents on different design gestures, it was found that the inventory supports different arch variations; ultimately, a curved three-hinged arch was chosen for further development. n. Usually, it is advantageous to duplicate the arch throughout the roof to limit the number of custom elements that need to be manufactured. However, since there is no need to cut any of the members, and variation in form can be added without requiring additional material processing or custom fabrication, the three-hinged arch was further refined to vary in height throughout the length of the hall, giving the roof double curvature. The arches were also angled to create a crisscross pattern which gives the roof lateral stability in both principal directions. Figure 47 shows the first design iteration of the final roof typology. These design gestures could not be created with the inventory. The simple solution was to scale the size of the curves, and re-run the agent, which resulted in a feasible design (Figure 48), which uses 1237 (88%) of the inventory. The agent did not accurately reach the end point of the gesture curve, necessitating a manual adjustment of the final triangles as shown on Figure 49. The gesture curves were extended by one meter, and the truss was then trimmed at the end by using available elements from the inventory, leading to a small amount of necessary off-cuts. Note that where the arches meet the ground, such adjustment is not necessary, as the agent starts the aggregation from this point.

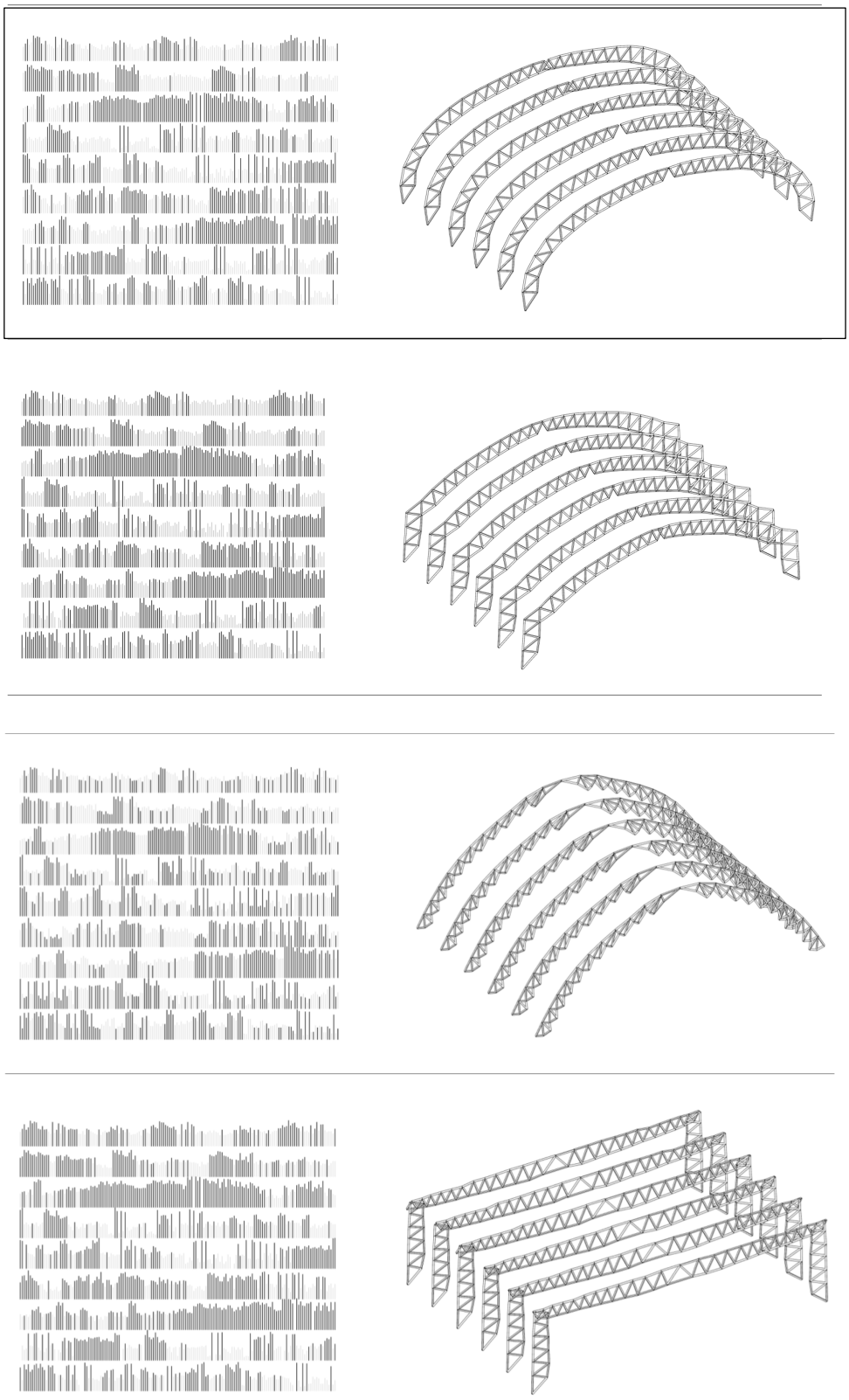


Figure 46: Design experiments with design gesture (top) selected for further development. Highlighted next to each design are the elements used to make it.

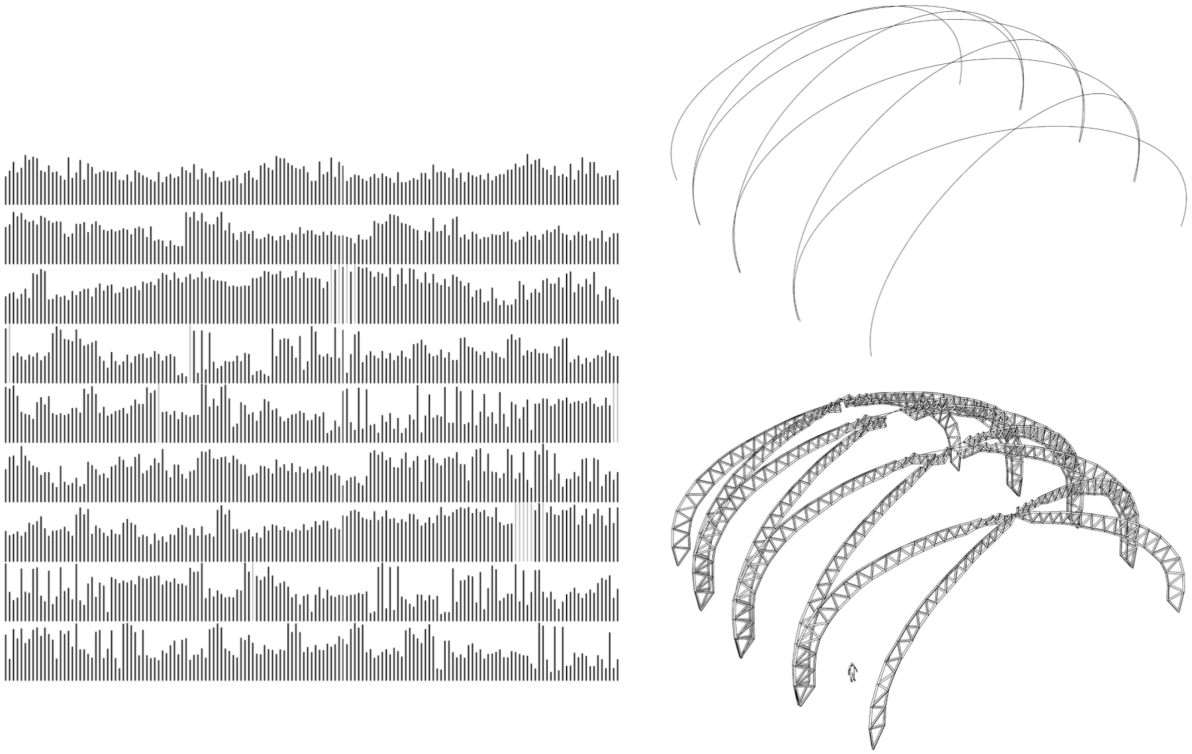


Figure 47: First design iteration with double curvature and crisscross typology. 1397 elements out of the 1413 elements available were used, but the agent failed to complete the truss.

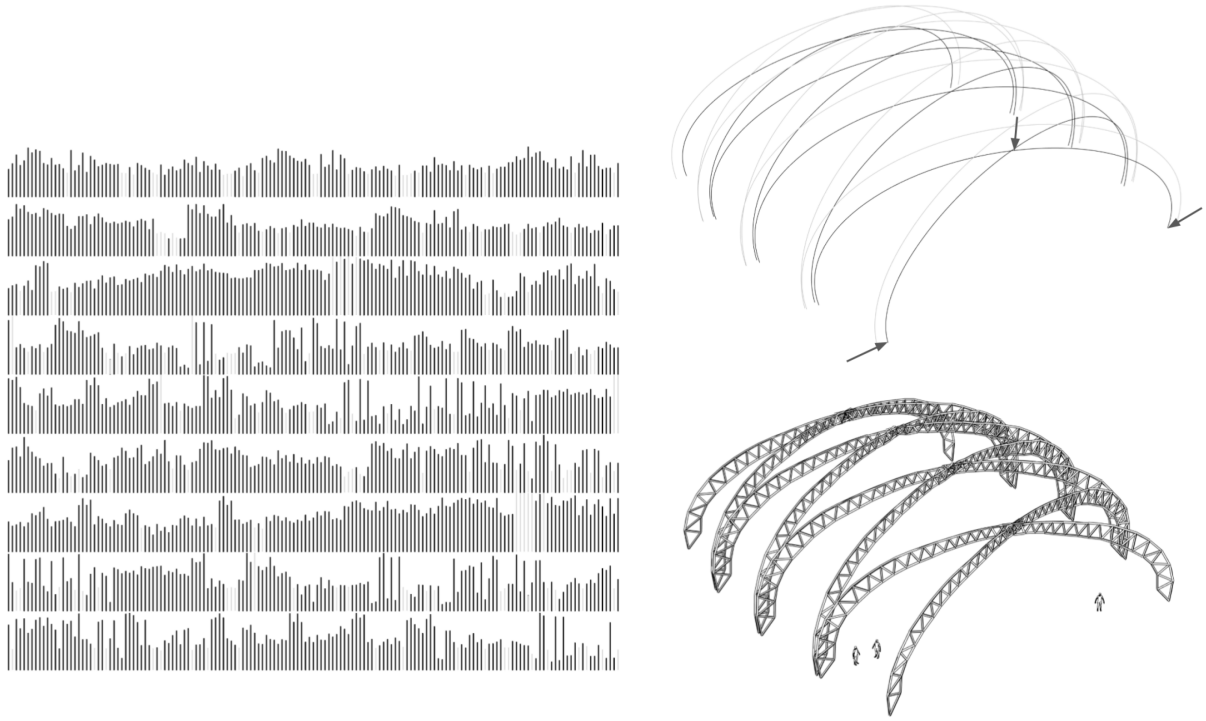


Figure 48: By changing the gesture curves, the agent is able to complete the truss. The final design uses 1237 of the 1413 available elements.

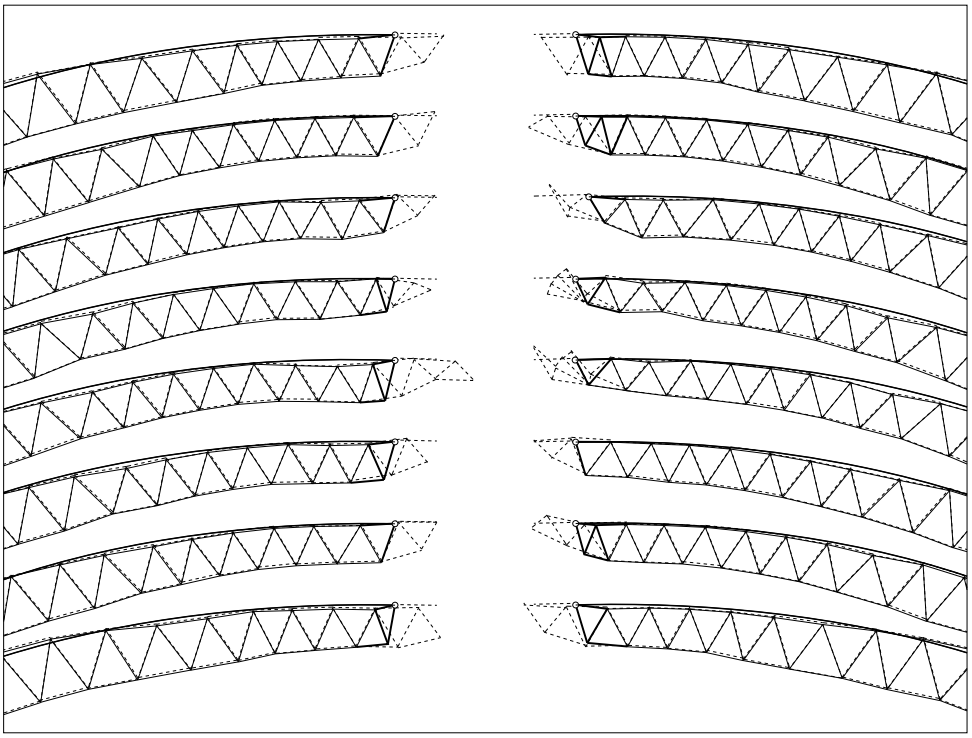
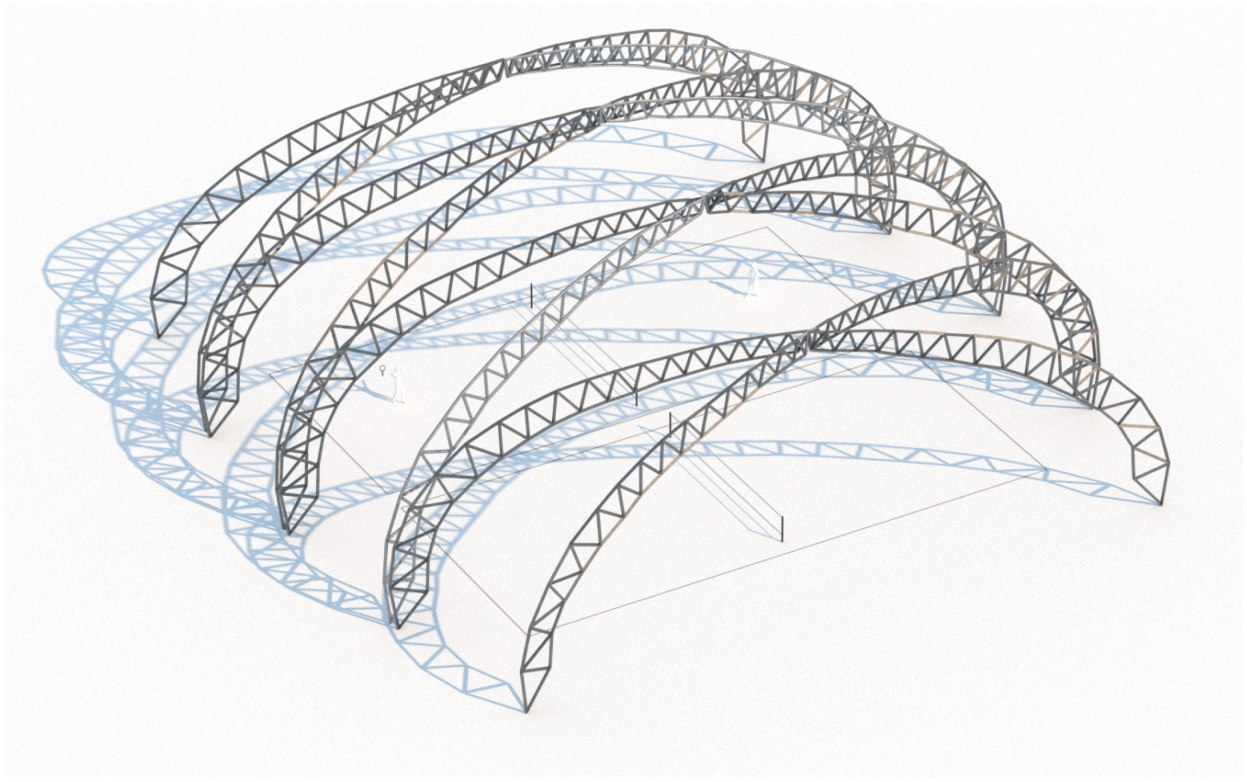


Figure 49: Manually adjusting the truss ends to meet the hinge-point. Dashed lines indicate the raw truss generated by the agent, and the dark triangles indicate the manually adjusted elements.

The final design consists of 8 unique arches arranged in a crisscross pattern (Figure 50). It uses a total of 1237 heterogeneous elements from the inventory, from which only 55 (4.4%) needed to be trimmed to precisely meet the hinge-point at the top of the arch. It spans over two standard-sized tennis courts.

4.4.5 Assembling

The theoretical assembly process is thought to proceed as follows: The pallets with the L-profiles are shipped to the site. Since they were packed roughly in order of disassembly, any given pallet will contain elements with indices close to each other. This simplifies the task for on-site construction workers, allowing them to find the pallet containing the next element required quickly. If locating a specific element proves difficult, they can use an RFID scanner. The planar trusses are assembled on-site.



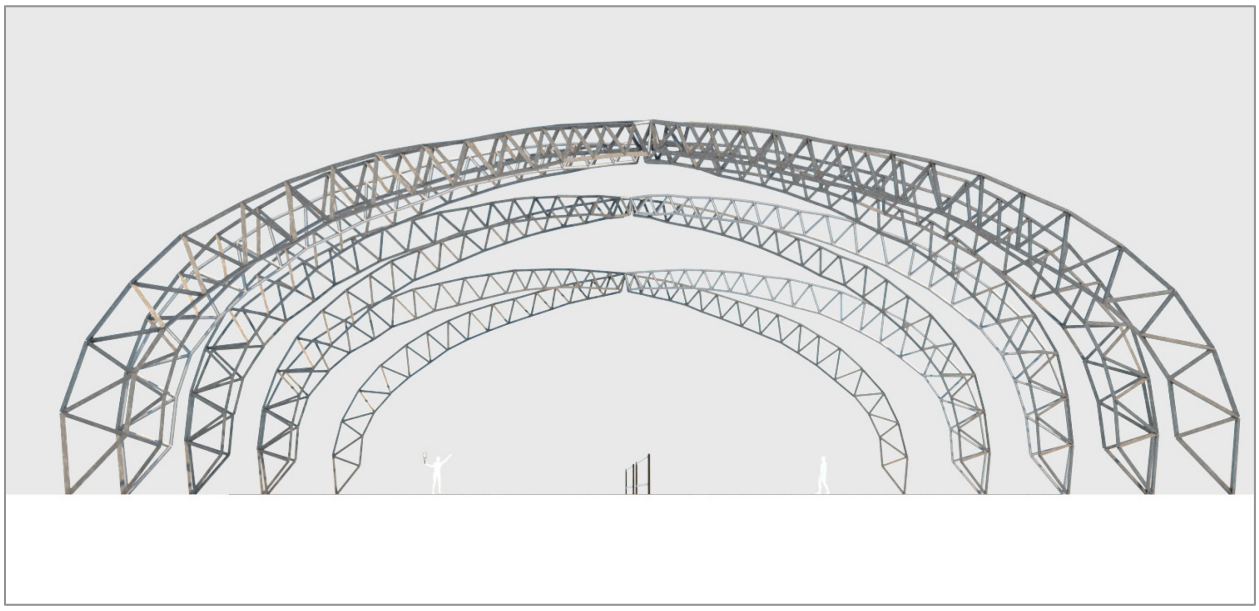


Figure 50: Final Hall structural truss design, rendered in perspective (top) and front view (bottom).

4.4.6 Inventory usage

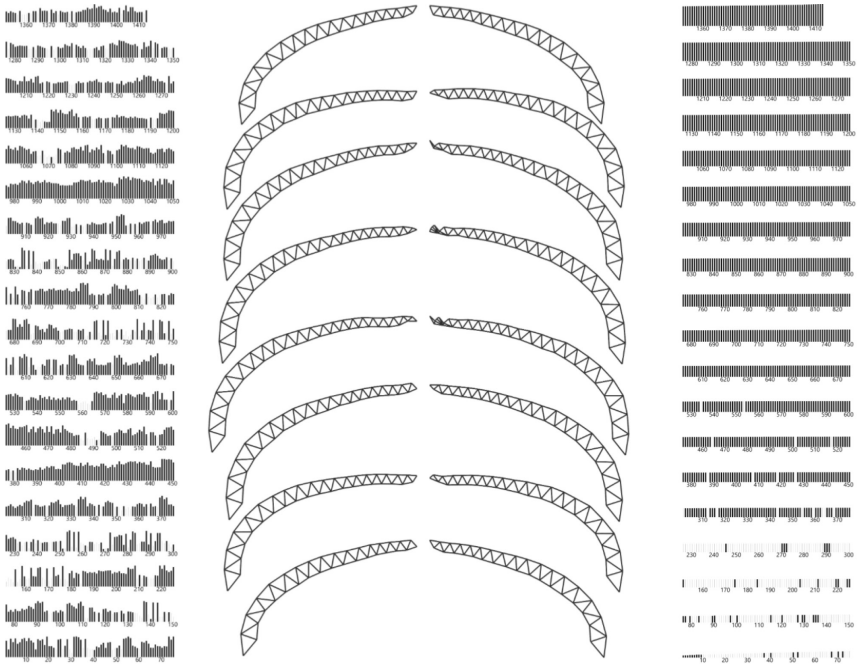


Figure 51: Mapping of elements from transmission tower to roof truss arches. Bold elements are those used in the design. Left shows the inventory sorted by disassembly sequence, right shows the inventory sorted by length.

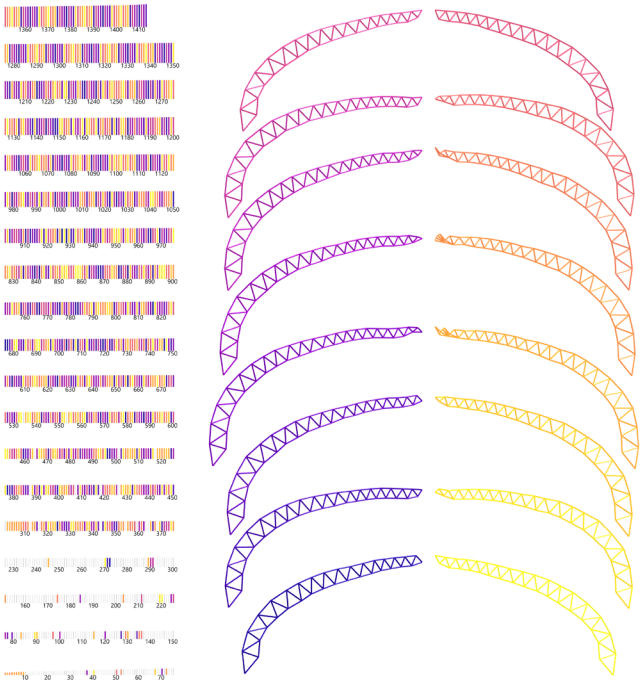


Figure 52: Mapping of elements from transmission tower to roof truss arches. The length-sorted inventory elements are color-coded by the arch they are used in.

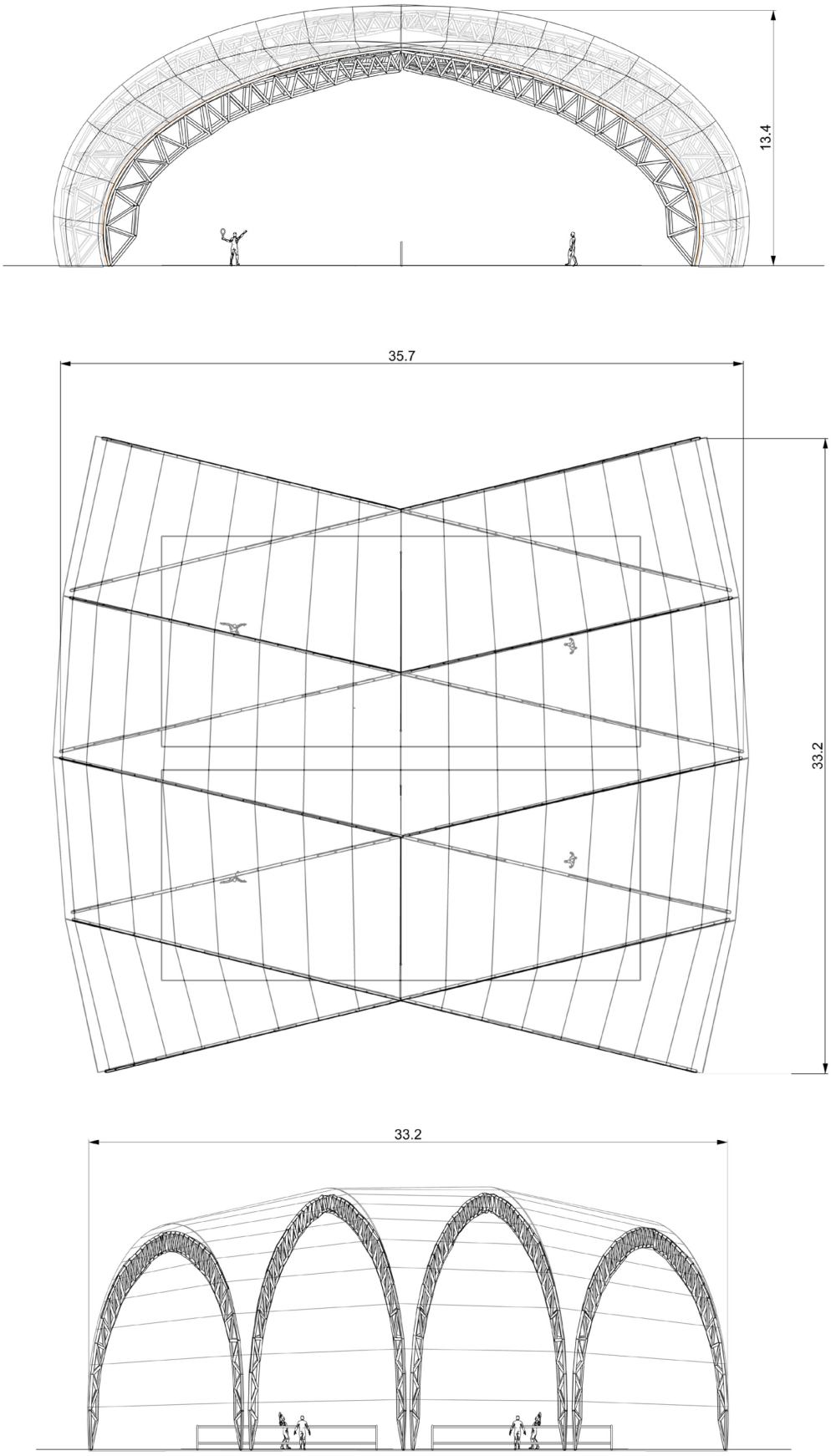


Figure 53: Front, plan, and elevation drawings of the tennis court.

4.4.7 FEM analysis

FEM analysis considering just the gravity-load-case confirms that the span deflects meets the deflection ASCE-7 (2017) deflection criteria of L/240, with no members being over-utilized. The analysis is set up in a similar way to previous analyses in the thesis, with pinned connections for the nodes and a distributed load applied to the top chord. The longest spanning arch was tested. It carries a portion of the roof with a tributary width of 4.21 m. The load is derived from a design load of 50psf:

$$50\ \text{psf} = 2.394\ \text{kN/m}^2 \Rightarrow\ 2.394\ \text{kN/m}^2 * 4.21\text{m} = 10.082\ \text{kN/m} = w$$

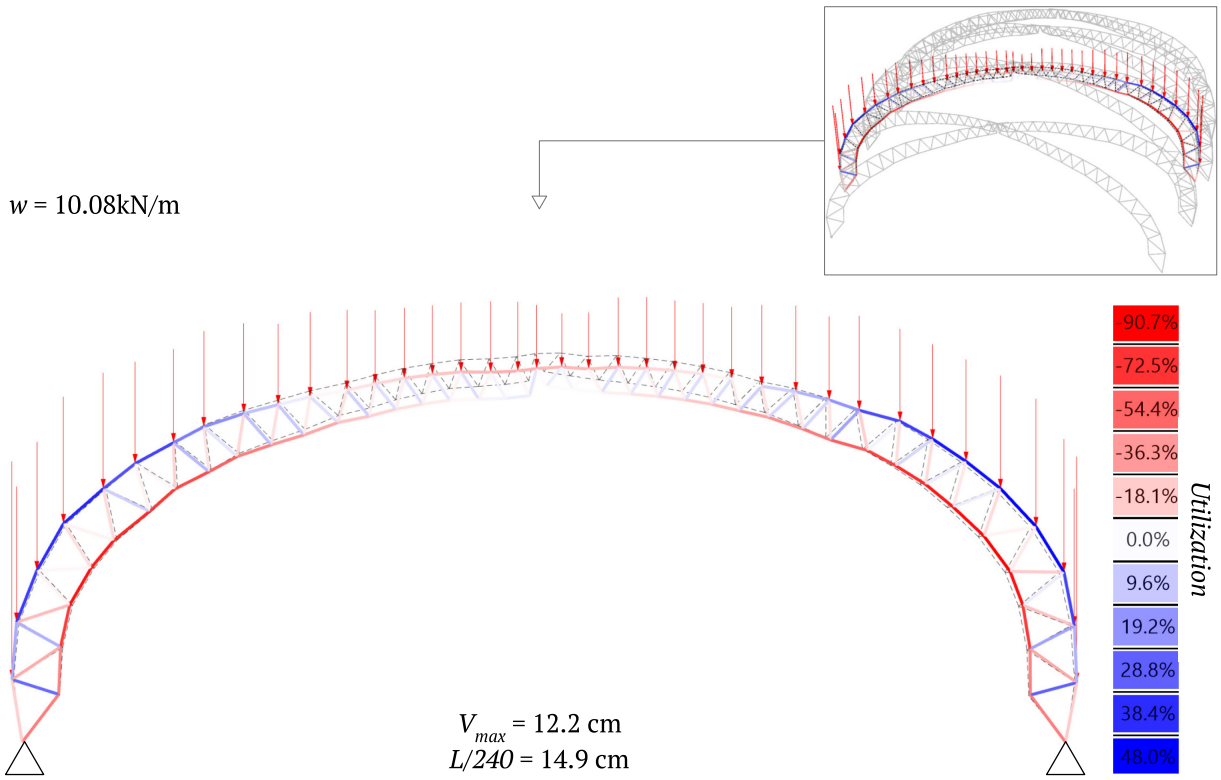


Figure 54: Simple structural analysis of the longest three-hinged arch truss in the design, subject to gravity load. Red arrows indicate the relative magnitude of w applied to the nodes.

4.4.8 Roof-truss interface

The roof covering the tennis courts is a simple sandwich panel construction, consisting of 2x10 plates sandwiched between two ½ inch plywood sheets. These panels are designed to only have single curvature and conform to the design gesture curves. To simplify construction, it was decided that they should not be made unique to meet the trusses. Instead, a spacer block (Figure 55 and Figure 57) is fabricated to accommodate the difference between the gesture curves and the effective truss layout.

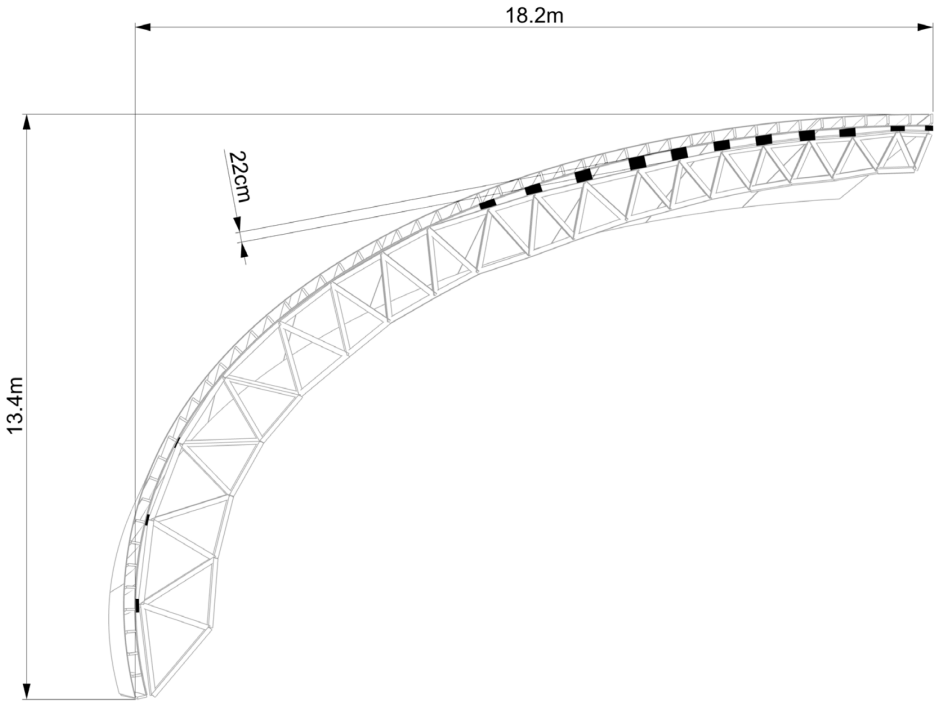


Figure 55: One of the trusses and the spacer blocks (black) added to accommodate the difference between the roof and the truss, with the height (22cm) of the tallest block.

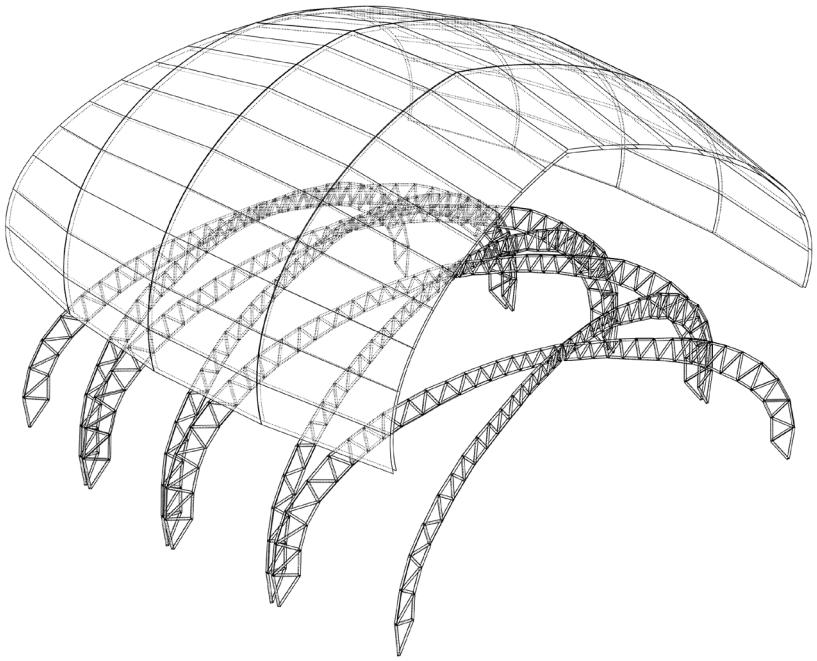


Figure 56: Exploded axonometric of the developable roof panels.

4.4.9 Construction Detail

A bolted gusset detail is used to assemble the space trusses with minimum processing. This allows for a ubiquitous detail that can support every unique node in the 8 unique trusses. The added gusset plate might come from used steel plates or be bought from recycled steel. This detail also highlights the potential for constructing planar trusses out of reused elements because trusses only require pin connections and are locked by the triangulated layout. The connections are free to rotate, allowing room for simple details such as this one, where a single detail has a high tolerance for varying specific node configurations without requiring custom fabrication. Additionally, with a bolted pin connection, the truss is designed for disassembly and can be reused in many projects. A gusset needs to be made for each node valency present in the layout. For the roof, there are valency 2-5 nodes, resulting in just four unique gusset plate joints.

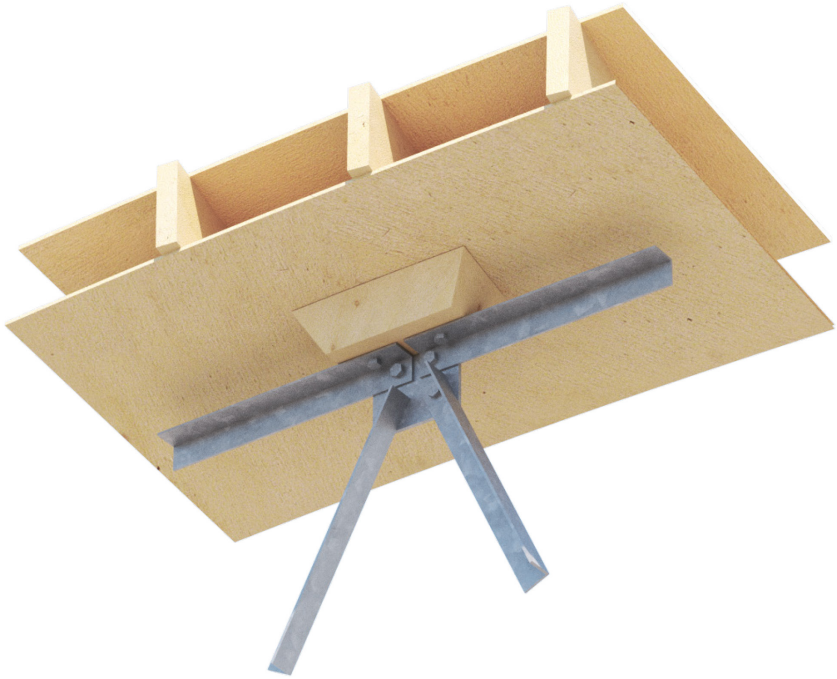


Figure 57: Node detail of valence 4 node, showing angle profiles, gusset plate, pinned connections, spacer block, and roof sandwich panel.

Figure 58: A single developable "roof strip" with three supporting trusses, and spacer blocks. Note that gusset plates are not drawn in this figure.

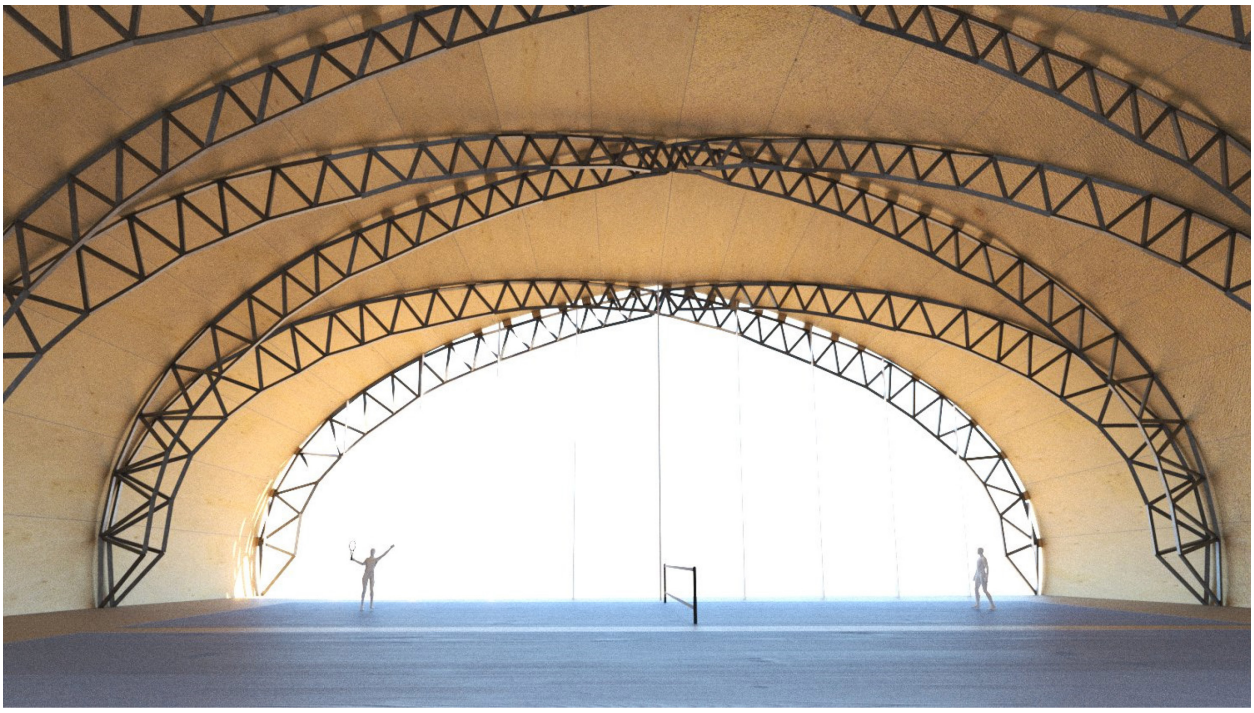


Figure 59: Interior perspective render of the final design.

Figure 60: Interior perspective render of the final design.

4.4.10 Discussion

The case study explores a theoretical yet realistic use case of performing Circular Design with the Deep RL workflow. A real-life heterogeneous inventory from a disassembled transmission tower was examined and utilized by the RL agent in collaboration with a human designer to perform Inventory-Constrained Generative Design. This approach designed the long-span arches for a tennis court. The conceptual design process took less than 1 hour, demonstrating the method's applicability and speed. It also highlighted remaining challenges with directly reusing structural elements, such as the need to customize interfaces as seen in the top hinge of the trusses, and the spacer blocks between the trusses and the roof.

4.5 Discussion of Results

The results demonstrate that RL is a flexible framework that can be tailored to specific design goals, such as the structural planar truss presented here. Other goals can also be envisioned; for example, one might simply apply a reward for regularity by using similarly sized members to achieve more regularly sized triangles. While the results are sometimes unpredictable and wild, this unpredictability is also the strength of the open-ended RL approach. In other words, the downsides of the method are complemented by other optimization methods. Indeed, it might not outperform optimization algorithms under more constrained design problems, but this is where it makes more sense to use conventional optimization. RL truly occupies its own unique niche in design problems, characterized by wide-open combinatorial design spaces.

5 Conclusion

5.1 Summary of contributions

This thesis introduces a novel deep reinforcement learning-based method for circular design that works by sequentially assembling elements from a constrained heterogeneous inventory of linear elements, requiring only a design gesture as input. A custom RL training environment was developed to achieve this, utilizing Proximal Policy Optimization with a Multi-Layer Perceptron Policy. Compared to the presented method, previous methods for computational circular design require more explicit deterministic inputs, such as a ground structure in topology optimization or a fully defined design target for matching-based approaches, and do not guarantee a perfect match with the inventory, often requiring further material processing. As such, the RL method is novel and powerful because it allows a designer to perform fewer steps to explore potential designs from a material inventory. The results showed that this method can generate trusses across various inventory spans and quantities. Generally, the agent can reliably assemble good trusses using 50%-75% of an inventory and use up to 100% for more homogeneous inventories. It was also demonstrated that the method is robust, producing reliable results within a 10% performance margin when faced with random variations of wild data. Furthermore, the model generalizes to a wide range of inventory types and

design gestures and can assemble complex designs from inventories with thousands of elements within a minute. Finally, a realistic example of applying it to a circular design problem was provided, demonstrating its usefulness in an open-ended design problem—a capability that previous computational approaches have not achieved.

5.2 Potential Impact

If the Architecture, Engineering, and Construction sector adopted this method, it could significantly reduce the embodied carbon footprint of new structures and divert C&D waste into new high-value architectural structures. Furthermore, reinforcement learning enables better control of sequential aggregation-based generative design, which might open up use cases for expressive structural design, taking advantage of all kinds of inventories—even standard catalog-based inventories. One could imagine a design workflow where designs and structures are generated on the fly as part of a digital design process, suggesting alternatives, variations, or new designs based on a vast knowledge of available material inventories locally or globally.

5.3 Limitations and future work

This work is a promising first step toward developing RL agents to assemble complex structures bottom-up from constrained inventories. It demonstrates the potential of RL-based approaches for Circular Design. Several limitations have been identified, along with opportunities for future research:

- **3D Extension:** A logical next step would be to extend the methodology to 3D. This could include following 3D design gesture curves to assemble trusses that curve spatially. The approach could be extended more ambitiously to construct space frames by selecting elements to form tetrahedrons rather than triangles. Transitioning to 3D would require rewriting the environment and restructuring the actions and rewards system.
- **Cross-section Consideration**: Currently, the method's real-world applicability is limited because it does not account for different cross-sections. Implementing observations and actions that facilitate the selection of various cross-sections, starting with the cross-sectional area, is crucial.
- **Generalizability:** It would be valuable to investigate how the agent generalizes to conditions beyond those it was trained in. For example, can it manage inventories that differ significantly from those produced by the procedural training algorithm, such as members exceeding the 3m maximum length?
- **Model Complexity:** Expanding the model to include a larger action and observation space could enhance its generalizability. Moreover, the current model is only trained on a single gravity load case. Integrating a more adaptable implementation of structural load cases would enable the agent to assemble structures meeting various demands, such as lateral loads or point loads.
- **Interactivity:** Incorporating more user feedback and control into the generative design process could potentially be achieved through what might be termed "user-input observations." This idea involves the agent receiving some observations not from the environment—as is

typical in reinforcement learning—but directly from users. For instance, a user could input a value as an observation, indicating how regular or irregular the desired truss should be. During training, if high irregularity is input, the system would assign high rewards for diversity in triangle aspect ratios. Conversely, a low variety input would result in rewards for maintaining similar aspect ratios. This method allows for various design inputs to be considered, accommodating multiple objectives such as emphasizing structural designs over precise gesture-following, utilizing every element in the inventory, or restricting the use only to shorter members. In this way, the interactivity with the agent would be more precise and akin to a classic 'parametric' workflow without requiring a pre-defined parametric setup.

- **Planning Mechanism:** Complex construction requires long-term planning rather than merely reactive decision-making. A planning mechanism like Monte Carlo Tree Search could augment the method and improve performance on larger-scale problems requiring extended planning horizons. Research by Bapst (2019) has shown that such planning mechanisms are generally beneficial for construction tasks.
- **Iterative Actions:** Exploring the benefits of allowing an agent more iterations of actions, and the capability to undo previous steps, could reflect the iterative design process typical of human designers. This would allow an agent to spend more time on a single structural design problem, iterating over previously added elements to refine the final layout.
- **Multi-Agent Approach:** Future studies could investigate a multi-agent approach where multiple specialized agents collaborate to design a structure. For example, one agent could specialize in providing an initial general layout, followed by another specializing in assembling a truss, with a final agent adding bracing or removing certain members, similar to the work discussed in Lin et al. (2020), where different agents specialize in different steps of 3d modeling.
- **Hyperparameters and Models:** Further experimentation with hyperparameters and various RL models and training algorithms, such as Soft Actor Critic, Deep Q Learning, or other network architectures like graph-based neural networks or RNNs, is necessary.
- **Advancements in RL:** RL still requires more fundamental research to become broadly useful, but it remains an exciting field with promising opportunities to address complex challenges.

5.4 Concluding remarks

This thesis introduces a novel approach to Circular Design, showcasing the potential of reinforcement learning to revolutionize building design practices by enabling the use of constrained inventories to generate key architectural elements such as spanning trusses. The results are promising and suggest that emerging machine learning technologies, like the one implemented in this study, could drive significant advancements in Circular Design. This includes managing heterogeneous inventories digitally and designing with these resources to support more sustainable building practices. The author envisions a future where technology does not restrict workflows to standard, component-based methods but rather enhances the care, flexibility, precision, and tailoring necessary to create unique architectural projects from unique elements—addressing a gap left by the current standardized industry.

6 Bibliography

Advantage Actor Critic (A2C). (2024). https://moon-ci-docs.huggingface.co/blog/deep-rl-a2c

Agostinelli, F., McAleer, S., Shmakov, A., & Baldi, P. (2019). Solving the Rubik's cube with deep reinforcement learning and search. *Nature Machine Intelligence*, *1*(8), Article 8. https://doi.org/10.1038/s42256-019-0070-z

Akizuki, Y., Bernhard, M., Kakooee, R., Kladeftira, M., & Dillenburger, B. (2020, August 5). *GENERATIVE MODELLING WITH DESIGN CONSTRAINTS Reinforcement Learning for Object Generation*. https://doi.org/10.3929/ethz-b-000452958

Al Khayat, L. K. Y. (2023). *Fibers and Fragments: Weaving Local Resources into the Arabian Gulf's Modern Material Culture* [M.Arch., Massachusetts Institute of Technology]. https://www.proquest.com/docview/2850177133/abstract/235A1DE7376D408BPQ/1

Albayrak, U., & Morshid, L. A. M. (2020). Evaluation of Seismic Performance of Steel Lattice Transmission Towers. *Civil Engineering Journal*, *6*(10), Article 10. https://doi.org/10.28991/cej-2020-03091600

Allwood, J. M., Cullen, J. M., Carruth, M. A., Cooper, D. R., McBrien, M., Milford, R. L., Moynihan, M. C., & Patel, A. C. (2012). *Sustainable materials: With both eyes open*. UIT Cambridge.

Almeida, P., Solas, M., Renz, A., Bühler, M. M., Gerbert, P., Castagnino, S., & Rothballer, C. (2016). *Shaping the Future of Construction: A Breakthrough in Mindset and Technology*. https://doi.org/10.13140/RG.2.2.21381.37605

Amtsberg, F., Huang, Y., Marshall, D., Gata, K., & Mueller, C. (2022). *Structural upcycling: Matching digital and natural geometry*. 17.

Apellániz, D., Pettersson, B., & Gengnagel, C. (2023). A Flexible Reinforcement Learning Framework to Implement Cradle-to-Cradle in Early Design Stages. In C. Gengnagel, O. Baverel, G. Betti, M. Popescu, M. R. Thomsen, & J. Wurm (Eds.), *Towards Radical Regeneration* (pp. 3–12). Springer International Publishing. https://doi.org/10.1007/978-3-031-13249-0_1

Apolinarska, A. A., Pacher, M., Li, H., Cote, N., Pastrana, R., Gramazio, F., & Kohler, M. (2021). Robotic assembly of timber joints using reinforcement learning. *Automation in Construction*, *125*, 103569. https://doi.org/10.1016/j.autcon.2021.103569

Art, G., Frazer, J., Frazer, J., Xiyu, L., Mingxi, T., & Janssen, P. (2002). *Generative and Evolutionary Techniques for Building Envelope Design*. https://www.semanticscholar.org/paper/Generative-and-Evolutionary-Techniques-for-Building-Art-Frazer/1f8b3becc169a6bfaceb769abd1a35feb330399a

ASCE. (2017). *Minimum Design Loads and Associated Criteria for Buildings and Other Structures*. American Society of Civil Engineers. https://doi.org/10.1061/9780784414248

Bakker, C., den Hollander, M., Peck, D., & Balkenende, R. (2018). Circular Product Design: Addressing Critical Materials through Design. In *Critical Materials: Vol. Volume 5* (pp. 179–192). WORLD SCIENTIFIC. https://doi.org/10.1142/9789813271050_0009

Bapst, V., Sanchez-Gonzalez, A., Doersch, C., Stachenfeld, K., Kohli, P., Battaglia, P., & Hamrick, J. (2019). Structured agents for physical construction. *Proceedings of the 36th International Conference on Machine Learning*, 464–474. https://proceedings.mlr.press/v97/bapst19a.html

Bellman, R., & Dreyfus, S. (2010). *Dynamic programming* (1. Princeton Landmarks in Mathematics ed., with a new introduction). Princeton University Press.

Bello, I., Pham, H., Le, Q. V., Norouzi, M., & Bengio, S. (2017). *Neural Combinatorial Optimization with Reinforcement Learning* (arXiv:1611.09940). arXiv. https://doi.org/10.48550/arXiv.1611.09940

Big Dig Building. (2014, November 22). *SsD*. https://www.ssdarchitecture.com/works/residential/big-dig-building/

Bouma, W., Fudos, I., Hoffmann, C., Cai, J., & Paige, R. (1995). Geometric constraint solver. *Computer-Aided Design*, *27*(6), 487–501. https://doi.org/10.1016/0010-4485(94)00013-4

Brown, N. K., Garland, A. P., Fadel, G. M., & Li, G. (2022). "Deep reinforcement learning for engineering design through topology optimization of elementally discretized design domains." *Materials & Design*, *218*, 110672. https://doi.org/10.1016/j.matdes.2022.110672

Brütting, J., Desruelle, J., Senatore, G., & Fivet, C. (2019). Design of Truss Structures Through Reuse. *Structures*, *18*, 128–137. https://doi.org/10.1016/j.istruc.2018.11.006

Brütting, J., Senatore, G., & Fivet, C. (Eds.). (2019). Form follows availability—Designing structures through reuse. *Journal of the International Association for Shell and Spatial Structures*. https://doi.org/10.20898/j.iass.2019.202.033

Brütting, J., Senatore, G., & Fivet, C. (2022). MILP-based discrete sizing and topology optimization of truss structures: New formulation and benchmarking. *Structural and Multidisciplinary Optimization*, *65*(10), 277. https://doi.org/10.1007/s00158-022-03325-7

Charniak, E. (2018). *Introduction to deep learning*. The MIT Press.

Chicago Wrecking Company. (1906). *Catalogue no. 145: $50,000,000 Louisiana Purchase Expedition bought by the Chicago House Wrecking Co.* The Company. http://archive.org/details/ChicagoWreckingCompany

Chu, Y.-C., & Lin, H.-H. (2019). RePack: Dense Object Packing Using Deep CNN with Reinforcement Learning. *2019 International Automatic Control Conference (CACS)*, 1–5. https://doi.org/10.1109/CACS47674.2019.9024360

Chung, H., Kim, J., Knyazev, B., Lee, J., Taylor, G. W., Park, J., & Cho, M. (2021). Brick-by-Brick: Combinatorial Construction with Deep Reinforcement Learning. *Advances in Neural Information Processing Systems*, *34*, 5745–5757. https://proceedings.neurips.cc/paper_files/paper/2021/hash/2d4027d6df9c0256b8d4474ce88f8c88-Abstract.html

Cloesen, H. (2018). *Structurally informed exploration of a grammar-based design space* [Master Thesis]. Massachusetts Institute of Technology.

Cousin, T., Marshall, D., Pearl, N., Alkhayat, L., & Mueller, C. (2023). *Integrating Irregular Inventories: Accessible technologies to design and build with nonstandard materials in architecture.*

Daehn, K. E., Cabrera Serrenho, A., & Allwood, J. M. (2017). How Will Copper Contamination Constrain Future Global Steel Recycling? *Environmental Science & Technology*, *51*(11), 6599–6606. https://doi.org/10.1021/acs.est.7b00997

Dantata, N., Touran, A., & Wang, J. (2005). An analysis of cost and duration of deconstruction and demolition residential buildings in Massachusetts. *Resources, Conservation and Recycling*, *44*, 1–15. https://doi.org/10.1016/j.resconrec.2004.09.001

Densley Tingley, D., Cooper, S., & Cullen, J. (2017). Understanding and overcoming the barriers to structural steel reuse, a UK perspective. *Journal of Cleaner Production*, *148*, 642–652. https://doi.org/10.1016/j.jclepro.2017.02.006

Dorn, W., Gomory, R., & Greenberg, H. J. (1964). *Automatic design of optimal structures*. https://www.semanticscholar.org/paper/Automatic-design-of-optimal-structures-Dorn-Gomory/9a0a4512b263ef0e6e9f92323da002c0f6a2989d

Du, W., Zhao, J., Yu, C., Yao, X., Song, Z., Wu, S., Luo, R., Liu, Z., Zhao, X., & Wu, Y. (2023). *Automatic Truss Design with Reinforcement Learning* (arXiv:2306.15182). arXiv. https://doi.org/10.48550/arXiv.2306.15182

Dunant, C., Drewniok, M., Sansom, M., Corbey, S., Allwood, J., & Cullen, J. (2017). Real and perceived barriers to steel reuse across the UK construction value chain. *Resources, Conservation and Recycling*, *126*, 118–131. https://doi.org/10.1016/j.resconrec.2017.07.036

Energinet. (2023). *Energinet fjerner 18 kilometer elmaster og luftledninger i og ved Roskilde Fjord*. https://energinet.dk/om-nyheder/nyheder/2023/11/energinet-fjerner-18-kilometer-el-master-og-luftledninger-i-og-ved-roskilde-fjord/

Environment, U. N. (2022, November 9). *2022 Global Status Report for Buildings and Construction*. UNEP - UN Environment Programme. http://www.unep.org/resources/publication/2022-global-status-report-buildings-and-construction

Eurostat. (2018). *Waste statistics*. http://ec.europa.eu/eurostat/statistics-explained/index.php/Waste_statistics

Fujitani, Y., & Fujii, D. (2000). Optimum structural design of steel plane frame under the limited stocks of members. *Proceedings of the RILEM/CIB/ISO International Symposium, Integrated Life-Cycle Design of Materials and Structures*, 198–202.

Ghasemipour, S. K. S., Kataoka, S., David, B., Freeman, D., Gu, S. S., & Mordatch, I. (2022). Blocks Assemble! Learning to Assemble with Large-Scale Structured Reinforcement Learning. *Proceedings of the 39th International Conference on Machine Learning*, 7435–7469. https://proceedings.mlr.press/v162/ghasemipour22a.html

Gilbert, M., & Tyas, A. (2003). Layout optimization of large-scale pin-jointed frames. *Engineering Computations*, *20*(8), 1044–1064. https://doi.org/10.1108/02644400310503017

Goodbun, J., Till, J., Rumpfhuber, A., & Klein, M. (2014). *The Design of Scarcity*. Strelka. https://westminsterresearch.westminster.ac.uk/item/q2x55/the-design-of-scarcity

Gorgolewski, M. (2018). *Resource Salvation: The Architecture of Reuse*. Ryerson University. https://learning.oreilly.com/library/view/resource-salvation/9781118928776/

Hamrick, J. B., Allen, K. R., Bapst, V., Zhu, T., McKee, K. R., Tenenbaum, J. B., & Battaglia, P. W. (2018). *Relational inductive bias for physical construction in humans and machines* (arXiv:1806.01203). arXiv. https://doi.org/10.48550/arXiv.1806.01203

Hayashi, K. (2021). *Reinforcement Learning for Optimal Design of Skeletal Structures* [Doctoral thesis, Kyoto University]. https://repository.kulib.kyoto-u.ac.jp/dspace/handle/2433/263614

Hayashi, K., & Ohsaki, M. (2021). Reinforcement learning for optimum design of a plane frame under static loads. *Engineering with Computers*, *37*(3), 1999–2011. https://doi.org/10.1007/s00366-019-00926-7

Herczeg, M., McKinnon, D., Milios, L., Bakas, I., Klaassens, E., Svatikova, K., & Widerberg, O. (2014). *Resource efficiency in the building sector.* (DG Environment Final Report). European Commission.

Hu, R., Xu, J., Chen, B., Gong, M., Zhang, H., & Huang, H. (2020). TAP-Net: Transport-and-pack using reinforcement learning. *ACM Transactions on Graphics*, *39*(6), 1–15. https://doi.org/10.1145/3414685.3417796

Huang, C. (2021). *Reinforcement Learning for Architectural Design-Build—Opportunity of Machine Learning in a Material-informed Circular Design Strategy*. 171–180. https://doi.org/10.52842/conf.caadria.2021.1.171

Huang, Y., Alkhayat, L., De Wolf, C., & Mueller, C. (2021). *Algorithmic circular design with reused structural elements: Method and tool*. 457–468. https://doi.org/10.35789/fib.PROC.0055.2021.CDSymp.P056

Intergovernmental Panel On Climate Change (Ipcc). (2023). *Climate Change 2022 – Impacts, Adaptation and Vulnerability: Working Group II Contribution to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change* (1st ed.). Cambridge University Press. https://doi.org/10.1017/9781009325844

Janner, M., Fu, J., Zhang, M., & Levine, S. (2021). *When to Trust Your Model: Model-Based Policy Optimization* (arXiv:1906.08253). arXiv. http://arxiv.org/abs/1906.08253

Janner, M., Levine, S., Freeman, W. T., Tenenbaum, J. B., Finn, C., & Wu, J. (2019). *Reasoning About Physical Interactions with Object-Oriented Prediction and Planning* (arXiv:1812.10972). arXiv. http://arxiv.org/abs/1812.10972

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., Bridgland, A., Meyer, C., Kohl, S. A. A., Ballard, A. J., Cowie, A., Romera-Paredes, B., Nikolov, S., Jain, R., Adler, J., … Hassabis, D. (2021). Highly accurate protein structure prediction with AlphaFold. *Nature*, *596*(7873), Article 7873. https://doi.org/10.1038/s41586-021-03819-2

Kavlakoglu. (2023, July 6). *AI vs. Machine Learning vs. Deep Learning vs. Neural Networks: What's the difference?* IBM Blog. https://www.ibm.com/blog/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks/www.ibm.com/blog/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks

Khalil, E., Dai, H., Zhang, Y., Dilkina, B., & Song, L. (2017). Learning Combinatorial Optimization Algorithms over Graphs. *Advances in Neural Information Processing Systems*, *30*. https://proceedings.neurips.cc/paper/2017/hash/d9896106ca98d3d05b8cbdf4fd8b13a1-Abstract.html

Knight, T., & Stiny, G. (2015). Making grammars: From computing with shapes to computing with things. *Design Studies*, *41*, 8–28. https://doi.org/10.1016/j.destud.2015.08.006

Lee, J., Fivet, C., & Mueller, C. (2015). Modelling with Forces: Grammar-Based Graphic Statics for Diverse Architectural Structures. In M. R. Thomsen, M. Tamke, C. Gengnagel, B. Faircloth, & F. Scheurer (Eds.), *Modelling Behaviour* (pp. 491–504). Springer International Publishing. https://doi.org/10.1007/978-3-319-24208-8_41

Lee, J., Mueller, C., & Fivet, C. (2016). Automatic generation of diverse equilibrium structures through shape grammars and graphic statics. *International Journal of Space Structures*, *31*(2–4), 147–164. https://doi.org/10.1177/0266351116660798

Lee, K. J., & Mueller, C. T. (2020). Adapting computational protein folding logic for growth-based, assembly-driven spatial truss design. *Proceedings of IASS Annual Symposia*, *2020*(7), 1–13.

Lee, K., & Mueller, C. (2024). *Differentiable assignment for circularity-driven structural design and optimization*. *43*.

Li, W., Bohg, J., & Fritz, M. (2017). *Acquiring Target Stacking Skills by Goal-Parameterized Deep Reinforcement Learning* (arXiv:1711.00267). arXiv. http://arxiv.org/abs/1711.00267

Mandow, L., Pérez-de-la-Cruz, J.-L., Rodríguez-Gavilán, A. B., & Ruiz-Montiel, M. (2020). Architectural planning with shape grammars and reinforcement learning: Habitability and energy

efficiency. *Engineering Applications of Artificial Intelligence*, *96*, 103909. https://doi.org/10.1016/j.engappai.2020.103909

Marshall, D., Meuller, C., Clifford, B., & Kennedy, S. (2020). COMPUTATIONAL ARRANGEMENT OF DEMOLITION DEBRIS. *Detritus*, 3–18. https://doi.org/10.31025/2611-4135/2020.13967

Mirra, G., & Pugnale, A. (2022). Expertise, playfulness and analogical reasoning: Three strategies to train Artificial Intelligence for design applications. *Architecture, Structures and Construction*. https://doi.org/10.1007/s44150-022-00035-y

Mirtsopoulos, I., & Fivet, C. (2023). Structural topology exploration through policy-based generation of equilibrium representations. *Computer-Aided Design*, *160*, 103518. https://doi.org/10.1016/j.cad.2023.103518

Mitchell, T. M. (2017). *Machine Learning*. McGraw Hill.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). *Playing Atari with Deep Reinforcement Learning* (arXiv:1312.5602). arXiv. https://doi.org/10.48550/arXiv.1312.5602

Mollica, Zachary & Self, Martin. (2016). *Advances in Architectural Geometry 2015 - Tree Fork Truss: Geometric Strategies for Exploiting Inherent Material Form*. vdf Hochschulverlag AG an der ETH Zürich. https://doi.org/10.3218/3778-4_11

Mueller, C. T., & Ochsendorf, J. A. (2015). Combining structural performance and designer preferences in evolutionary design space exploration. *Automation in Construction*, *52*, 70–82. https://doi.org/10.1016/j.autcon.2015.02.011

Nolte, T., Witt, A., Degen, M., & Tucker, J. (2015). *Mine the Scrap Installation*. https://certainmeasures.com/mts_installation.html

Pongiglione, M., & Calderini, C. (2014). Material savings through structural steel reuse: A case study in Genoa. *Resources, Conservation and Recycling*, *86*, 87–92. https://doi.org/10.1016/j.resconrec.2014.02.011

Puterman, M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming* (1st ed.). Wiley. https://doi.org/10.1002/9780470316887

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., & Dormann, N. (2021). Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research*, *22*(268), 1–8.

Rajeev, S., & Krishnamoorthy, C. S. (1992). Discrete Optimization of Structures Using Genetic Algorithms. *Journal of Structural Engineering*, *118*(5), 1233–1250. https://doi.org/10.1061/(ASCE)0733-9445(1992)118:5(1233)

Rossi, A. (2023). *Mediated Assemblies: An Open Source Software Approach to Combinatorial Design and Fabrication* [Ph.D. Thesis, Technische Universität Darmstadt]. https://doi.org/10.26083/tuprints-00024039

Ruby, I., Ruby, A., Bridger, J., Holcim Foundation for Sustainable Construction, & Universidad Iberoamericana (Eds.). (2010). *Re-inventing construction: International Holcim Forum for Sustainable Construction on "Re-Inventing Construction", that took place in April 2010 at the Universidad Ibéroamericana in Santa Fé, Mexico City ; 3rd Holcim Forum*. Ruby.

Ruiz-Montiel, M., Boned, J., Gavilanes, J., Jiménez, E., Mandow, L., & Pérez-de-la-Cruz, J.-L. (2013). Design with shape grammars and reinforcement learning. *Advanced Engineering Informatics*, *27*(2), 230–245. https://doi.org/10.1016/j.aei.2012.12.004

Schoon, N. (2016). *The BedZED Story*. Bioregional Development Group. https://storage.googleapis.com/www.bioregional.com/downloads/The-BedZED-Story_Bioregional_2017.pdf

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). *Proximal Policy Optimization Algorithms* (arXiv:1707.06347). arXiv. https://doi.org/10.48550/arXiv.1707.06347

Seats, D. C. (2020). *Form finding of structural reused material trusses with graphic statics* [Thesis, Massachusetts Institute of Technology]. https://dspace.mit.edu/handle/1721.1/127322

Seo, J., & Kapania, R. K. (2021). Development of an Artificial Intelligence System to Design of Structures using Reinforcement Learning: Proof of Concept. In *AIAA Scitech 2021 Forum*. American Institute of Aeronautics and Astronautics. https://doi.org/10.2514/6.2021-1692

Shea, K., Aish, R., & Gourtovaia, M. (2005). Towards integrated performance-driven generative design tools. *Automation in Construction*, *14*(2), 253–264. https://doi.org/10.1016/j.autcon.2004.07.002

Shea, K., & Cagan, J. (1999). The design of novel roof trusses with shape annealing: Assessing the ability of a computational method in aiding structural designers with varying design intent. *Design Studies*, *20*(1), 3–23. https://doi.org/10.1016/S0142-694X(98)00019-2

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., & Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, *529*(7587), Article 7587. https://doi.org/10.1038/nature16961

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., & Hassabis, D. (2017). *Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm* (arXiv:1712.01815). arXiv. https://doi.org/10.48550/arXiv.1712.01815

Steelman, A. (2022). *A Computational Framework for Zero Waste Structural Design* [Thesis, Massachusetts Institute of Technology]. https://dspace.mit.edu/handle/1721.1/144586

Sterrenberg, A. (2023). *Deep Generative Design: A Deep Learning Framework for Optimized Spatial Truss Structures with Stock Constraints*. https://repository.tudelft.nl/islandora/object/uuid%3A089c0e3f-9c87-4087-b919-ae86d20342a9

Stiny, G. (1980). Introduction to shape and shape grammars. *Environment and Planning B: Planning and Design*, *7*(3), 343–351. https://doi.org/10.1068/b070343

Sun, H., & Ma, L. (2020). Generative Design by Using Exploration Approaches of Reinforcement Learning in Density-Based Structural Topology Optimization. *Designs*, *4*(2), Article 2. https://doi.org/10.3390/designs4020010

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement Learning: An Introduction* (First Edition). Bradford Books.

United Nations Environment Programme. (2022). *2022 Global Status Report for Buildings and Construction: Towards a Zero-emission, Efficient and Resilient Buildings and Construction Sector*.

US EPA, O. (2016, March 8). *Sustainable Management of Construction and Demolition Materials* [Overviews and Factsheets]. https://www.epa.gov/smm/sustainable-management-construction-and-demolition-materials

Vallat, G., Wang, J., Maddux, A., Kamgarpour, M., & Parascho, S. (2023). Reinforcement learning for scaffold-free construction of spanning structures. *Proceedings of the 8th ACM Symposium on Computational Fabrication*, 1–12. https://doi.org/10.1145/3623263.3623359

Van Marcke, A., Laghi, V., & Carstensen, J. V. (2024). Automated planar truss design with reclaimed partially disassembled steel truss components. *Journal of Building Engineering*, *84*, 108458. https://doi.org/10.1016/j.jobe.2024.108458

Wibranek, B., Liu, Y., Funk, N., Belousov, B., Peters, J., & Tessmann, O. (2021). *Reinforcement Learning for Sequential Assembly of SL-Blocks*. https://doi.org/10.52842/conf.ecaade.2021.1.027

Wietschorke, L., Liu, Y., & Chen, J. (2020). *Reinforcement Learning for Architectural Combinatorial Optimization*.

Winston, P. H. (1980). Learning and reasoning by analogy. *Communications of the ACM*, *23*(12), 689–703. https://doi.org/10.1145/359038.359042

World Steel Association. (2020). *Life cycle inventory study report, 2020 data release*. https://worldsteel.org/publications/bookshop/life-cycle-inventory-study-report-2020-data-release/

Xie, Y. M., & Steven, G. P. (1993). A simple evolutionary procedure for structural optimization. *Computers & Structures*, *49*(5), 885–896. https://doi.org/10.1016/0045-7949(93)90035-C

Xu, H., Hui, K.-H., Fu, C.-W., & Zhang, H. (2019). Computational LEGO® Technic Design. *ACM TRANSACTIONS ON GRAPHICS*, *38*(6), 196. https://doi.org/10.1145/3355089.3356504

Yang, Z., Yang, S., Song, S., Zhang, W., Song, R., Cheng, J., & Li, Y. (2021). PackerBot: Variable-Sized Product Packing with Heuristic Deep Reinforcement Learning. *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 5002–5008. https://doi.org/10.1109/IROS51168.2021.9635914

Zhang, C., Hu, M., Di Maio, F., Sprecher, B., Yang, X., & Tukker, A. (2022). An overview of the waste hierarchy framework for analyzing the circularity in construction and demolition waste management in Europe. *Science of The Total Environment*, *803*, 149892. https://doi.org/10.1016/j.scitotenv.2021.149892

Zhou, G., Luo, L., Xu, H., Zhang, X., Guo, H., & Zhao, H. (2022). Brick Yourself within 3 Minutes. *2022 International Conference on Robotics and Automation (ICRA)*, 6261–6267. https://doi.org/10.1109/ICRA46639.2022.9812161

Zirek, S. (2023). Bottom-up generative up-cycling: A part based design study with genetic algorithms. *Results in Engineering*, *18*, 101099. https://doi.org/10.1016/j.rineng.2023.101099

7 Appendix

7.1 Roof sandwich plate

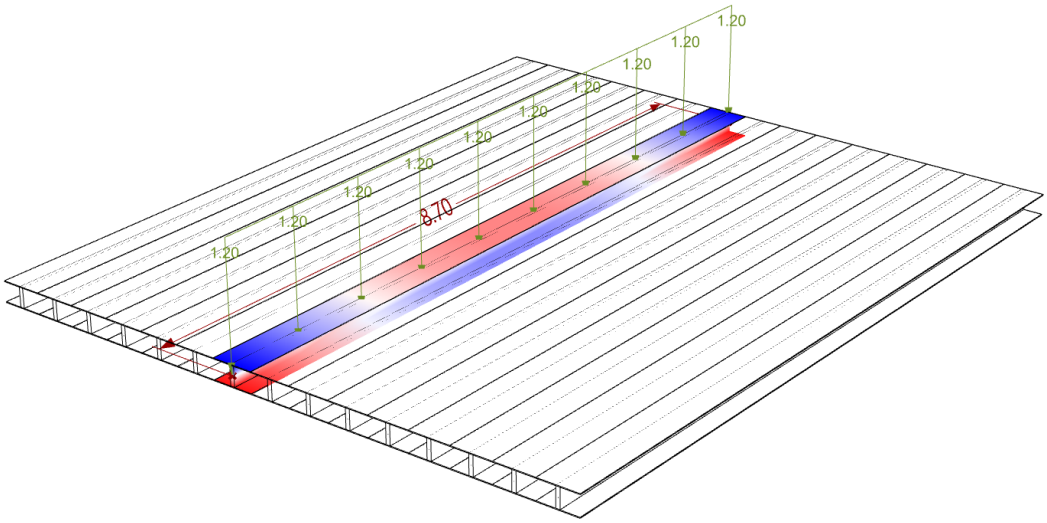


Figure 61: Structural analysis of roof panels.

While not the focus of the thesis, a basic roof design is proposed, consisting of sandwich plate panels with 12mm plywood and a core of 2X10 plates spaced at an interval of 50cm. The calculated deflection of the longest (8m) span is 0.90cm.

7.2 Example sequence

Below is a full 21 step example sequence generated by the Structural Agent:

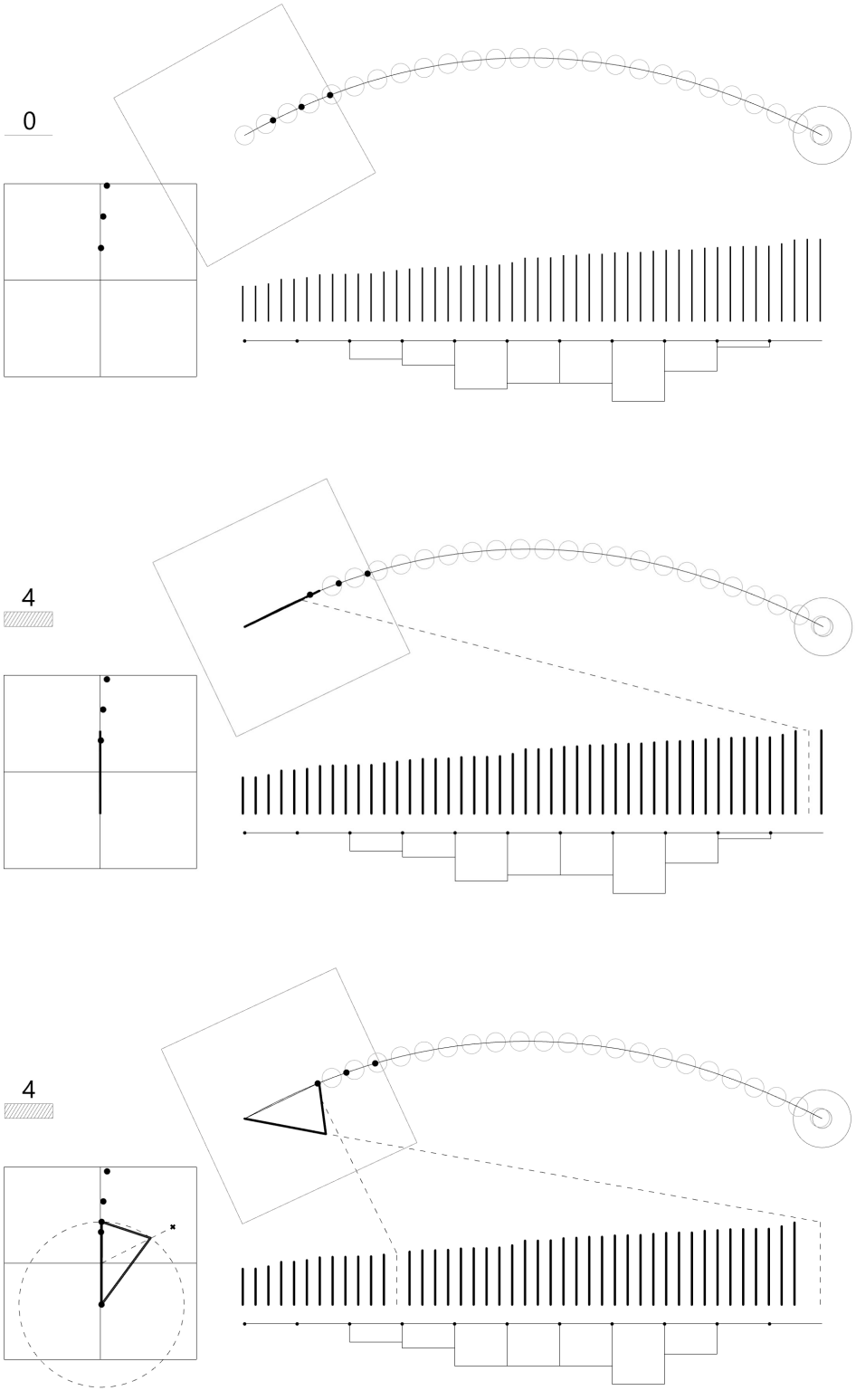


Figure 62: Example generated truss (step 1-3).

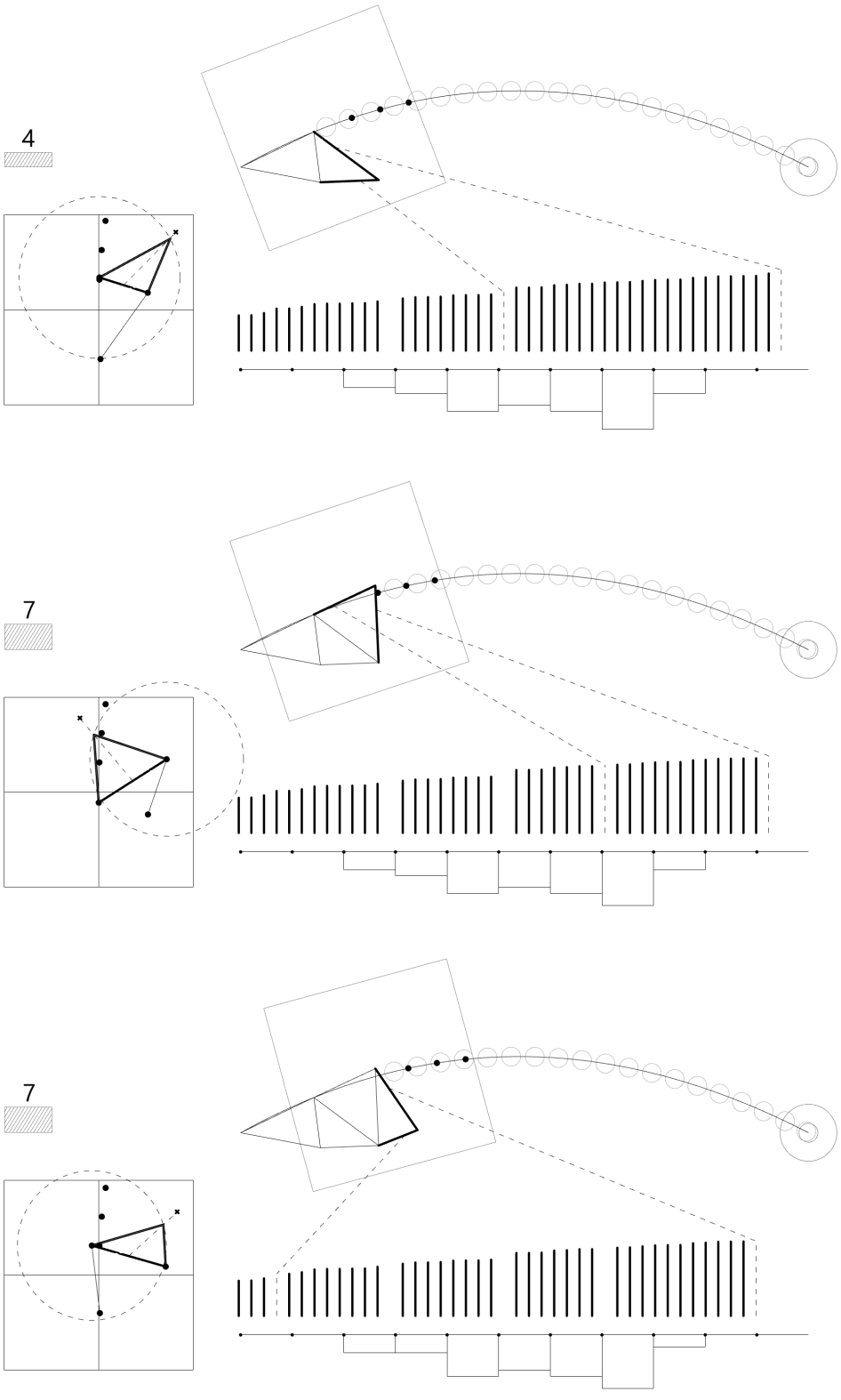


Figure 63: Example generated truss (step 4-6).

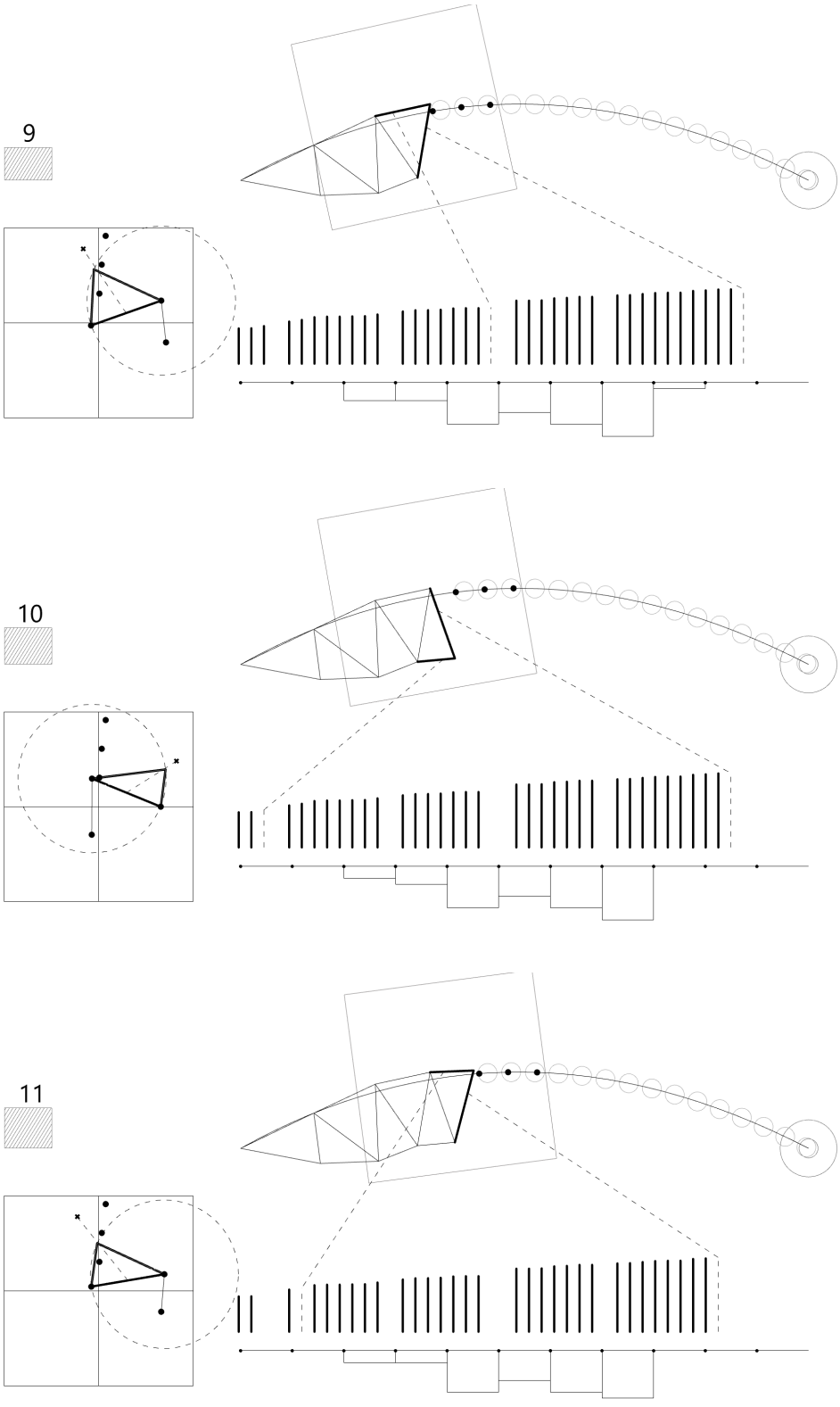


Figure 64: Example generated truss (step 7-9).

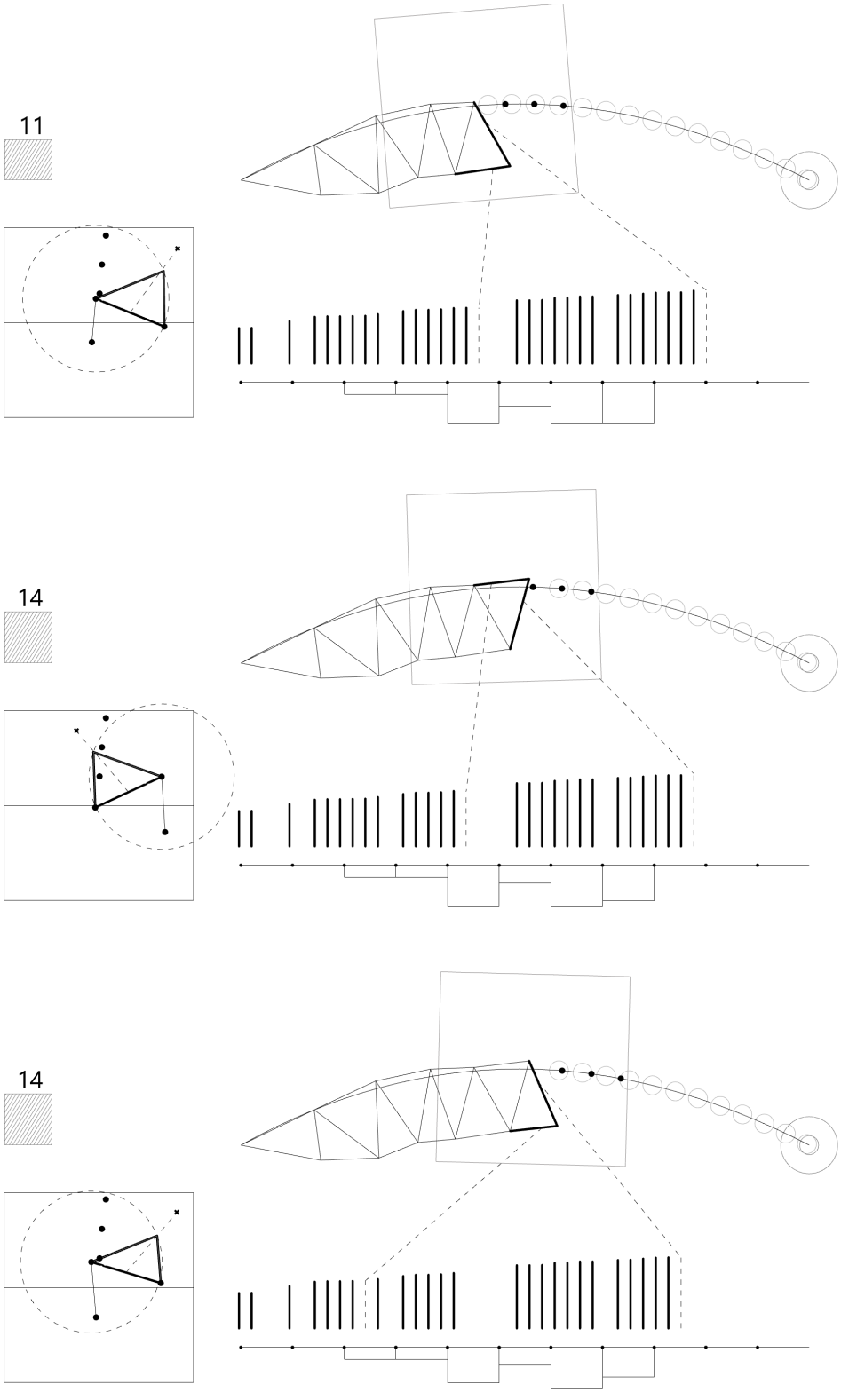


Figure 65: Example generated truss (step 10-12).

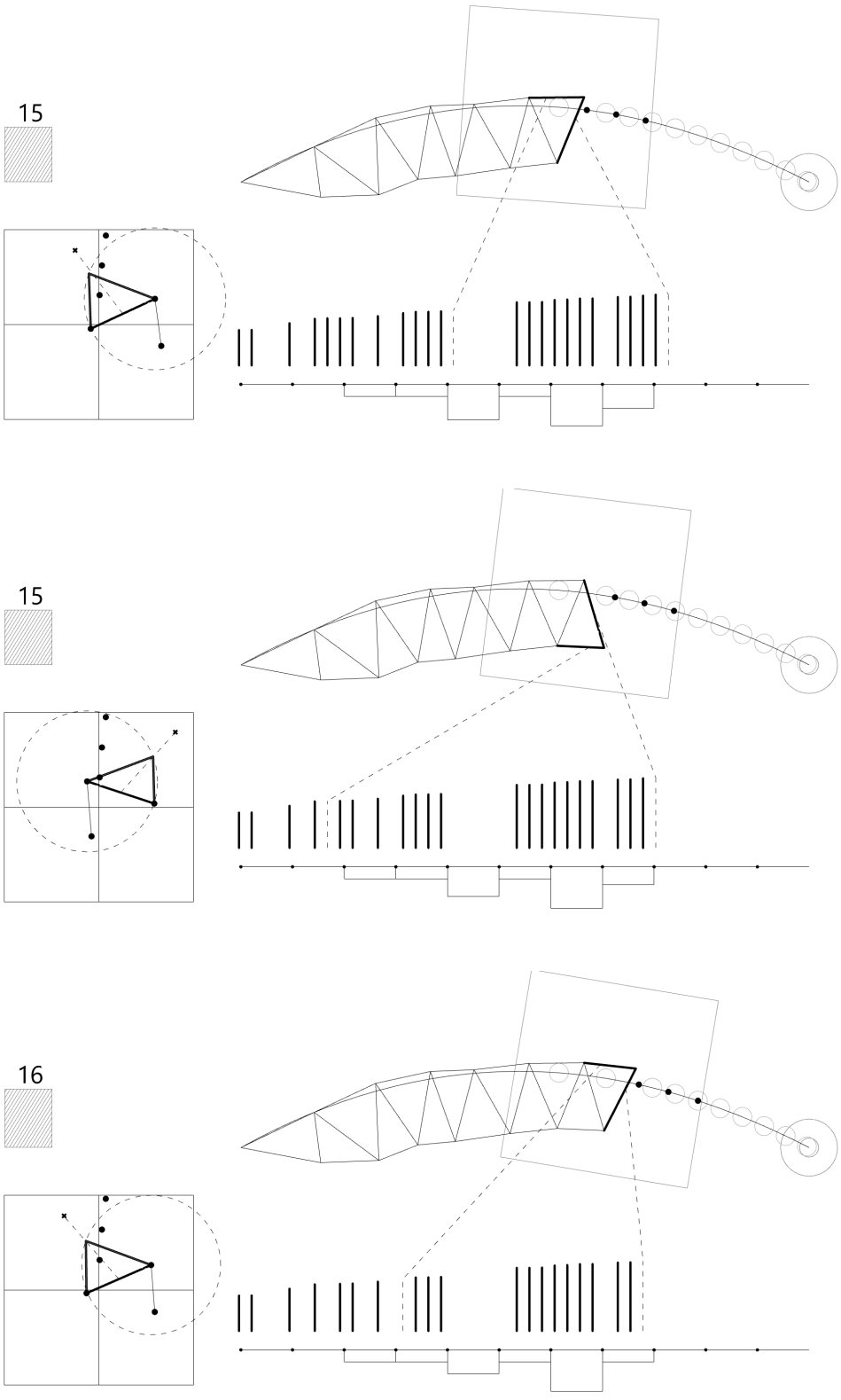


Figure 66: Example generated truss (step 13-15).

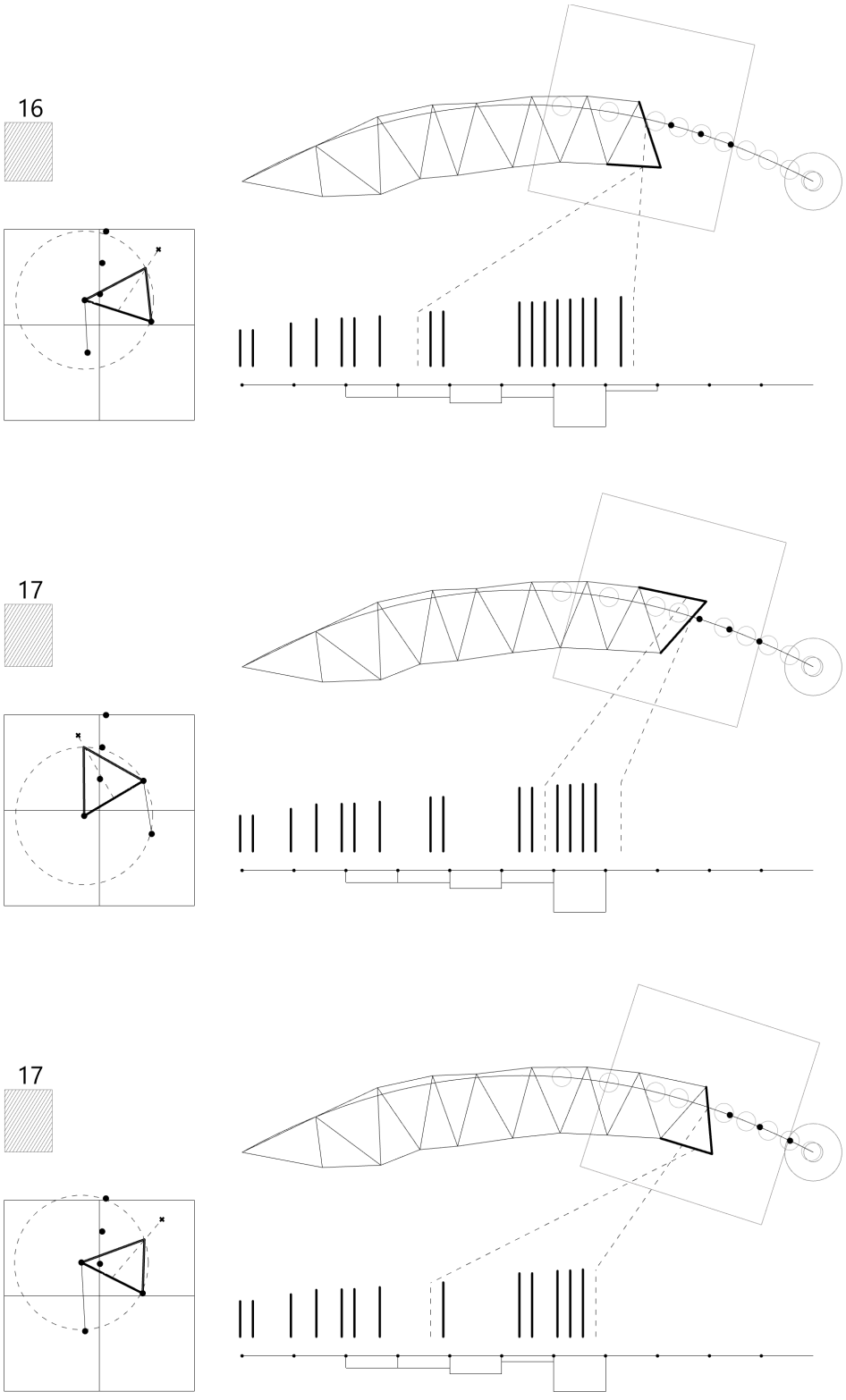


Figure 67: Example generated truss (step 16-18).

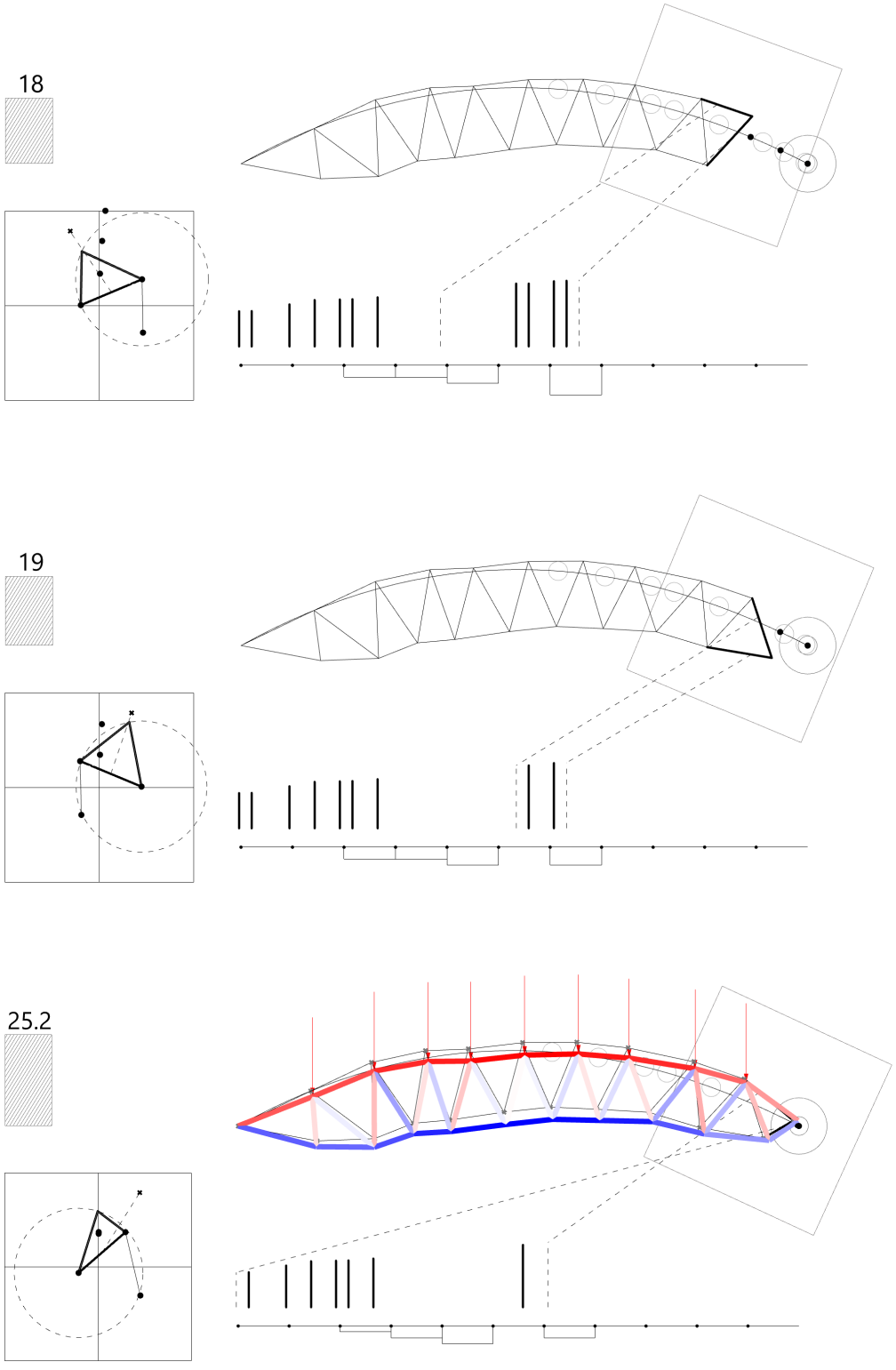


Figure 68: Example generated truss (step 19-21).

7.3 Training script

```
import socket
import struct
import pickle
import numpy as np
import os
import gym
from stable_baselines3 import PPO
from stable_baselines3.common.callbacks import CheckpointCallback
from stable_baselines3.common.monitor import Monitor

################################
# Environment connection code based on
# Wibranek, Bastian, Yuxi Liu, Niklas Funk, Boris Belousov, Jan Peters, and Oliver
Tessmann. 2021.
# Reinforcement Learning for Sequential Assembly of SL-Blocks.
https://doi.org/10.52842/conf.ecaade.2021.1.027.
################################

class Connection:
    def __init__(self, s):
        self._socket = s
        self._buffer = bytearray()

    def receive_object(self):
        while len(self._buffer) < 4 or len(self._buffer) < struct.unpack("<L",
self._buffer[:4])[0] + 4:
            new_bytes = self._socket.recv(16)
            if len(new_bytes) == 0:
                return None
            self._buffer += new_bytes
        length = struct.unpack("<L", self._buffer[:4])[0]
        header, body = self._buffer[:4], self._buffer[4:length + 4]
        obj = pickle.loads(body)
        self._buffer = self._buffer[length + 4:]
        return obj

    def send_object(self, d):
        body = pickle.dumps(d, protocol=2)
        header = struct.pack("<L", len(body))
        msg = header + body
        self._socket.send(msg)

class Env(gym.Env):
    def __init__(self, addr):
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.bind(addr)
        s.listen(1)
        clientsocket, address = s.accept()

        self._socket = clientsocket
        self._conn = Connection(clientsocket)
```

```
        ############################
        self.action_space = gym.spaces.Box(low=np.array([-1.0, -1.0, -1.0]),
high=np.array([1.0, 1.0, 1.0]), shape=(3,), dtype=np.float32)
        self.observation_space = gym.spaces.Box(low=-1., high=1., shape=(26,))
        ############################

    def reset(self):
        self._conn.send_object("reset")
        msg = self._conn.receive_object()
        self.action_space = eval(msg["info"]["action_space"])
        self.observation_space = eval(msg["info"]["observation_space"])
        return msg["observation"]

    def step(self, action):
        self._conn.send_object(action.tolist())  # Convert numpy array to list and
send
        msg = self._conn.receive_object()
        obs = msg["observation"]
        rwd = msg["reward"]
        done = msg["done"]
        info = msg["info"]
        return obs, rwd, done, info

    def close(self):
        self._conn.send_object("close")
        self._socket.close()

# Ask the user for the name of the training session
training_name = input("Enter the name for the training session: ")
timesteps = int(input("enter amount of timesteps: "))

addr = ("127.0.0.1", 50710)
env = Monitor(Env(addr))
obs = env.reset()

num_experiments = 1 # Number of experiments

policy_kwargs = dict(net_arch=[128, 128, 128]) #NN architecture

for i in range(num_experiments):
    log_dir = f"{training_name}"  # Use the user-provided name for the log directory
    tensorboard_log_dir = f"./tensorboard/{log_dir}"  # Adjusted tensorboard log di-
rectory
    checkpoint_callback = CheckpointCallback(save_freq=1000,
save_path=f'./trained_models/{log_dir}',
                                             name_prefix='model')
    model = PPO('MlpPolicy', env, policy_kwargs=policy_kwargs, n_steps=1024,
n_epochs=10, learning_rate=3e-3, gamma=0.99, batch_size=256, verbose=1, tensor-
board_log=tensorboard_log_dir)

    model.learn(total_timesteps=timesteps, log_interval=1, callback=check-
point_callback)
    env.reset()
```

```
cum_rwd = 0
obs = env.reset()
for i in range(300):
    action, _states = model.predict(obs, deterministic=True)
    obs, reward, done, info = env.step(action)
    print(i, action, reward, done, info)
    if done:
        obs = env.reset()
        print("Return = ", cum_rwd)
        cum_rwd = 0
env.close()
```

7.4 Evaluation script (inference)

```
import socket
import struct
import pickle
import numpy as np
import gym
from stable_baselines3 import PPO
from stable_baselines3.common.monitor import Monitor
import argparse

################################
# Environment connection code based on
# Wibranek, Bastian, Yuxi Liu, Niklas Funk, Boris Belousov, Jan Peters, and Oliver
Tessmann. 2021.
# Reinforcement Learning for Sequential Assembly of SL-Blocks.
https://doi.org/10.52842/conf.ecaade.2021.1.027.
################################

class Connection:
    def __init__(self, s):
        self._socket = s
        self._buffer = bytearray()

    def receive_object(self):
        while len(self._buffer) < 4 or len(self._buffer) < struct.unpack("<L",
self._buffer[:4])[0] + 4:
            new_bytes = self._socket.recv(16)
            if len(new_bytes) == 0:
                return None
            self._buffer += new_bytes
        length = struct.unpack("<L", self._buffer[:4])[0]
        header, body = self._buffer[:4], self._buffer[4:length + 4]
        obj = pickle.loads(body)
        self._buffer = self._buffer[length + 4:]
        return obj

    def send_object(self, d):
        body = pickle.dumps(d, protocol=2)
        header = struct.pack("<L", len(body))
```

```
        msg = header + body
        self._socket.send(msg)

class Env(gym.Env):
    def __init__(self, addr):
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.bind(addr)
        s.listen(1)
        clientsocket, address = s.accept()

        self._socket = clientsocket
        self._conn = Connection(clientsocket)
        ############################
        self.action_space = gym.spaces.Box(low=np.array([-1.0, -1.0, -1.0]),
high=np.array([1.0, 1.0, 1.0]), shape=(3,), dtype=np.float32)
        self.observation_space = gym.spaces.Box(low=-1., high=1., shape=(25,))
        ############################

    def reset(self):
        self._conn.send_object("reset")
        msg = self._conn.receive_object()
        self.action_space = eval(msg["info"]["action_space"])
        self.observation_space = eval(msg["info"]["observation_space"])
        return msg["observation"]

    def step(self, action):
        self._conn.send_object(action.tolist())
        msg = self._conn.receive_object()
        obs = msg["observation"]
        rwd = msg["reward"]
        done = msg["done"]
        info = msg["info"]
        return obs, rwd, done, info

    def close(self):
        self._conn.send_object("close")
        self._socket.close()

parser = argparse.ArgumentParser()
parser.add_argument('--folder', help='Folder where the model is located')
parser.add_argument('--timestep', help='Timestep of the model to load', type=int)
args = parser.parse_args()

# Example usage: --folder "path/to/model" --timestep 3000
# Construct the path to the model file based on the provided arguments

model_path = f"{args.folder}\model_{args.timestep}_steps.zip"

addr = ("127.0.0.1", 50710)
env = Monitor(Env(addr))
obs = env.reset()

model = PPO.load(model_path, env=env)

cum_rwd = 0
```

```
assembly_seq = []

deterministic_input = input("Run the model deterministically? (yes/no):
").strip().lower()
deterministic = deterministic_input == 'yes'


for i in range(10000):
    action, _states = model.predict(obs, deterministic=deterministic)
    assembly_seq.append(action)
    obs, reward, done, info = env.step(action)
    print(i, action, reward, done, info)
    if done:
        print("Return = ")
        obs = env.reset()
        cum_rwd = 0
print(assembly_seq)
env.close()
```

7.5 Environmnent files

The Grasshopper definition for the evaluation and training environments can be found by following this link.

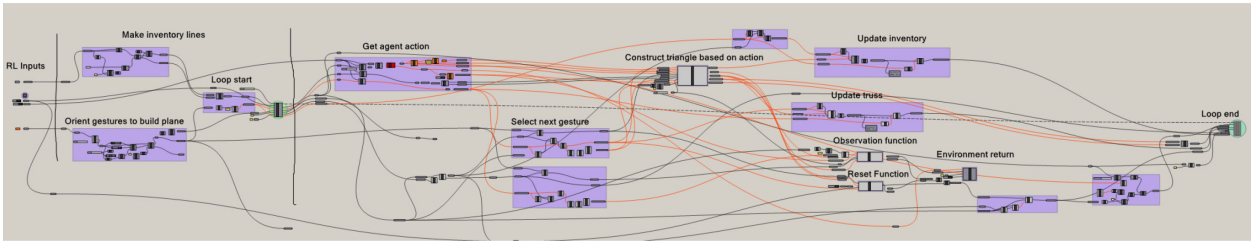


Figure 69: Overview of the evaluation environment definition.

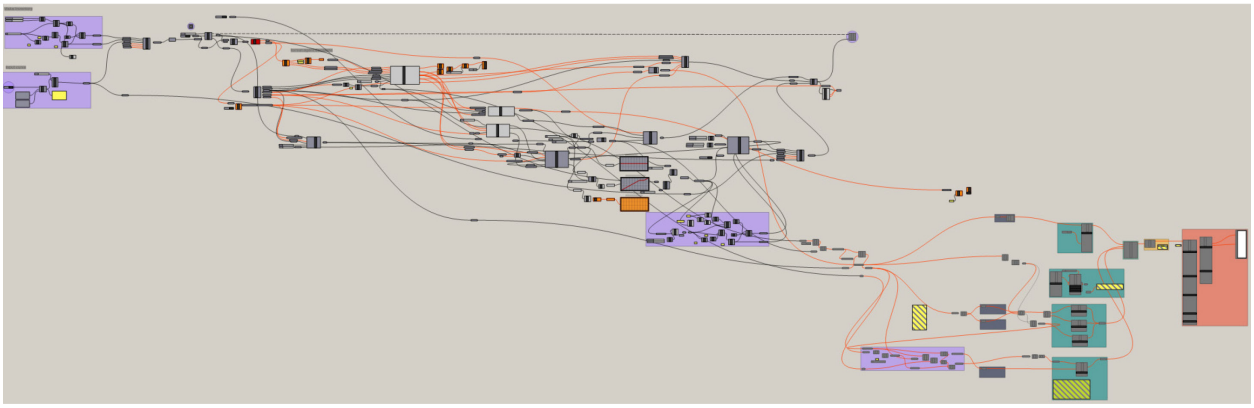


Figure 70: Overview of the training environment definition.