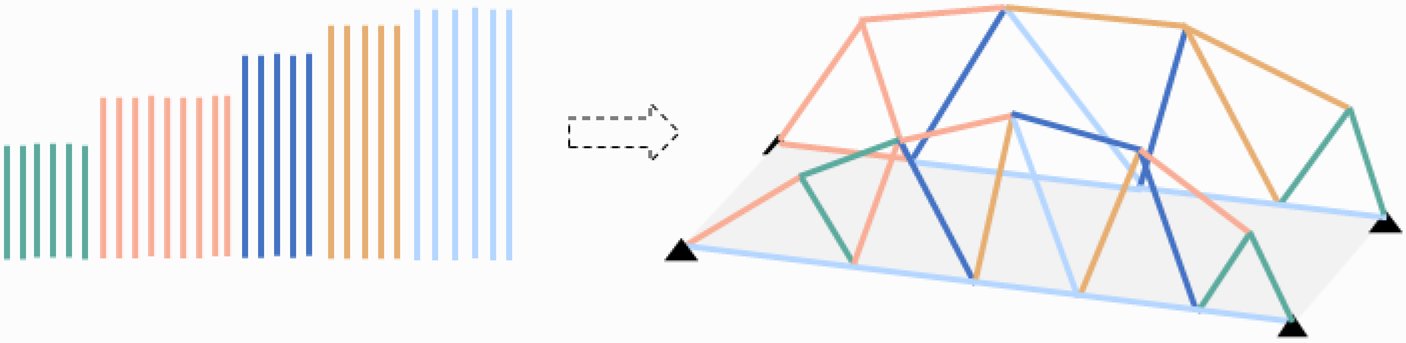
Efficiently including reclaimed steel elements in a truss bridge design

By performing a stock-constrained shape and topology optimisation

Fé van Lookeren Campagne



Efficiently including reclaimed steel elements in a truss bridge design

By performing a stock-constrained shape and topology optimisation

by

Fé van Lookeren Campagne

to obtain the degree of Master of Science
at the Delft University of Technology

Student number:	4542851	
Project duration:	March 1, 2022 – December 2, 2022	
Thesis committee:	dr. ir. F. P. van der Meer,	TU Delft, Chairman
	ir. C. Noteboom,	TU Delft
	dr. ir. F. Kavoura,	TU Delft
	ir. M. Sonneveld	Arup

Preface

This master thesis will conclude my studies at the faculty of Civil Engineering at the TUDelft, which began with the bachelor Civil Engineering and ended with the master Structural Engineering, specialisation Structural Mechanics. I carried out my master thesis in cooperation with Arup.

I have really enjoyed working on this thesis over the past months. I feel fortunate to have been able to combine my interests in parametric design, structural mechanics and sustainable design in my research topic. During my minor, I learnt about parametric design and a world opened up to me. I became very interested in the possibilities that parametric design and the inclusion of optimisation steps offer in developing smarter and more thoughtful design solutions. I am glad that in my thesis I was able to make the bridge between parametric design and answering puzzling structural mechanics problems. This graduation project has also given me confirmation that I want to seek this algorithmic way of thinking in my future professional life.

Furthermore, I feel fortunate with the people around me who guided and supported me during this large project. First, my supervisor Michelle Sonneveld, with whom I had weekly meetings about my research. I am very grateful for your close involvement in my project and the critical questions you asked me, so that I ended up taking every step with confidence. I also appreciate how I could always share my doubts and that our conversations often led me to new insights.

I would also like to thank the rest of my thesis committee: Frans van der Meer, Chris Noteboom and Florentia Kavoura who helped me a lot with their feedback and were always very supportive. You gave me the courage to proceed with certain research choices.

Furthermore, I would like to thank my colleagues at Arup, not only for the pleasant working environment, but also for the valuable conversations and for always being there for questions.

Also, I would like to thank my parents for their enthusiasm and interest in my thesis and the conversations we had when I started this project, which helped me choose a direction.

Finally, I would like to thank my housemates for the enjoyable distractions at the end of the sometimes long study days.

In conclusion, I would like to say: enjoy reading!

Fé van Lookeren Campagne
Delft, November 2022

Abstract

The construction industry is accountable for a large part of the global energy use and greenhouse gas emissions. It is important to change the current industry into a more circular system so that energy usage and emissions are reduced. To climb up the circular economy hierarchy, structural steel may be reused instead of recycled, which is currently the most common practice. An overarching digital database to efficiently match supply and demand of reclaimed structural elements may stimulate their application. Nevertheless, a limitation in the availability of reclaimed elements is expected. This thesis aims to overcome this bottleneck by providing an alternative approach to structural design. A change in design process will enable the design of structures utilising the availability of elements. In this design process, form follows availability. Structures that have a highly flexible geometry, which can be used to the advantage of maximising the inclusion of reclaimed elements, are trusses. Trusses are flexible in that they can have irregular geometry and the elements may have different profiles. The research aim and scope result in the following research question:

How can the design process of a truss footbridge be adapted such that the structural design efficiently includes reclaimed steel elements from a random and limited stock?

Literature has been reviewed on the characteristics of reclaimed steel and the construction of truss footbridges to clarify what the objectives, requirements and constraints of an adapted design process should entail. This overview includes several optimisation objectives, an overview of the required calculations, dimensional and geometrical constraints considering the manufacturability of connections and lastly, an adapted buckling safety factor to account for uncertainties regarding the material properties of reclaimed steel.

To utilise the opportunity of trusses to include irregular geometry, truss topology optimisation methods can be used to generate and compare non-standard truss designs. By changing the geometry and topology of a truss, available reclaimed elements may be efficiently included in the design. Various truss topology optimisation methods were compared in their adaptability to stock-constrained design. One of these methods is the ground structure method, which has been used in other research on stock-constrained truss design. The resulting design processes include fitting elements into a prescribed regular grid of positions. Steps are included to optimise the placement of elements. However, creation of cut-off waste and inclusion of new elements are in this method challenging to avert. A different topology optimisation method, the growth method, provides inspiration for a new and more intuitive stock-constrained truss design. Using a growth algorithm, truss structures may be grown in density or dimensions by adding, shifting and eliminating elements or connecting nodes. In stock-constrained design, available elements may in this case be attached to each other to form a triangular mesh resulting in a 100% reclaimed truss design without cut-off waste. The geometry in this case effectively follows from availability.

Using the growth method as inspiration and considering the framework of objectives, constraints and requirements that emerged from research, a complete stock-constrained growth process has been developed in the parametric design environment Grasshopper. The algorithm has been scripted in Python. By defining the required width and span of a truss bridge and presenting a stock of reclaimed elements, a truss bridge can be generated consisting out of only reclaimed steel elements, with almost no cut-off waste. In the algorithm, optimisation steps are implemented to maximise the material efficiency. The truss bridge design complies to Eurocode provisions and only contains manufacturable connections. The design process includes the following steps:

1. The required bottom chord of the truss is calculated and chosen from the stock of elements so that material efficiency is maximised, the number of connections minimised and whereby preference is given to a sequence of identical profiles.

2. From the start of the bottom chord, truss elements are added in pairs of 3.

   2.1. Elements are selected randomly from the database, whereby certain constraints to the selection are added to increase the suitability of the chosen options.

   2.2. The geometry follows from the length of the selected elements.

   2.3. The resulting internal forces are calculated with the method of section and tested against the element capacities. Strength and member instability are considered. The method of section provides the possibility of calculating the internal forces while the geometry is still being grown.

   2.4. If the checks comply, the geometry is added to a list of valid options.

   2.5. Steps 2.1-2.4 are repeated until the list of valid options reaches a specified length.

   2.6. From this list, the most material efficient option is chosen and the algorithm moves on to the generation of the next triangle. Intermediate steps are visualised in Rhino.

3. After the full truss is grown, global calculations are performed using the finite element analysis tool Karamba. Vertical deflection and global lateral stability are checked.

4. If the truss satisfies all required checks, it is added to a solution cloud of valid truss options. Steps 1-3 are repeated until the solution cloud includes a chosen number of unique, locally optimal truss designs.

5. In the solution cloud, a manual step is presented whereby the best truss design is selected according to a chosen objective. The possible objectives are:

   - Minimal environmental shadowcost
   - Minimal mass
   - Maximum capacity utilisation
   - Least number of elements

6. Lastly, full truss bridge designs may be generated by combining two unique trusses. The resulting truss bridge design is checked with Karamba. The valid design options are again collected and compared in a solution cloud.

The main objective of the new design process is to include reclaimed elements as efficiently as possible, so that a truss bridge design can be generated with reduced environmental impact. The developed growth method has been compared to existing traditional and reuse methods to check whether it can provide more environmental friendly designs. Compared to a traditional truss made of new steel, the truss of the growth method has 16% more mass. However, the case study shows that by avoiding new steel in the design, presenting a method that does not produce any cut-off waste and adapting the geometry to element availability, the developed algorithm still results in a steel truss with 17% less embodied carbon than the truss design whereby reclaimed elements are fit in afterwards, and 63% less embodied carbon than the traditionally designed new steel truss.

Contents

1 Introduction

1.1. Relevance

1.1.1. Reuse of steel

The construction industry is accountable for around a third of the world's total energy use and global energy-related carbon dioxide emissions (IEA, 2021). The production of construction materials, with steel being one of the most used materials, is the largest contributor to the production of the emissions of the construction sector (Iacovidou and Purnell, 2016). Of the total global carbon emissions, 3.3 % can be allocated to the production of structural steel (Iacovidou and Purnell, 2016). The world population is growing and the need for new buildings and infrastructure will only increase(Iacovidou and Purnell, 2016). The latest global status report for buildings and construction from the UN Environment Programme states that a significant reduction of the greenhouse gas emissions coming from the building industry is needed to reach the 2050 Paris Agreement goals (IEA, 2021). This goal is to have net zero carbon emissions by 2050 (United Nations, 2020). In the current situation, excluding the effects of the Covid pandemic, decarbonisation is at 40 % of the reference path to reach this goal in 2050(IEA, 2021). In addition, the Ministry of Infrastructure and Water Management in the Netherlands has defined its own goal to operate in a circular and climate-neutral manner by 2030 (Ministerie van Infrastructuur en Waterstaat, 2022).

A circular building industry can reduce carbon emissions, energy use and waste. The PBL Netherlands Environmental Assessment Agency presents a scheme on how a circular economy can be incorporated and the hierarchy that exists between the various strategies (see figure 1.1) (PBL Netherlands Environmental Assessment Agency, The Hague, 2017). They do this by presenting the 10 R's. The first R is Refuse and the last R Recover energy. The goal is to get as close as possible to the first R in order to obtain the highest value reuse of resources within the cycle (PBL Netherlands Environmental Assessment Agency, The Hague, 2017). In the scheme it is stated that one should aim for reuse before thinking about recycling.

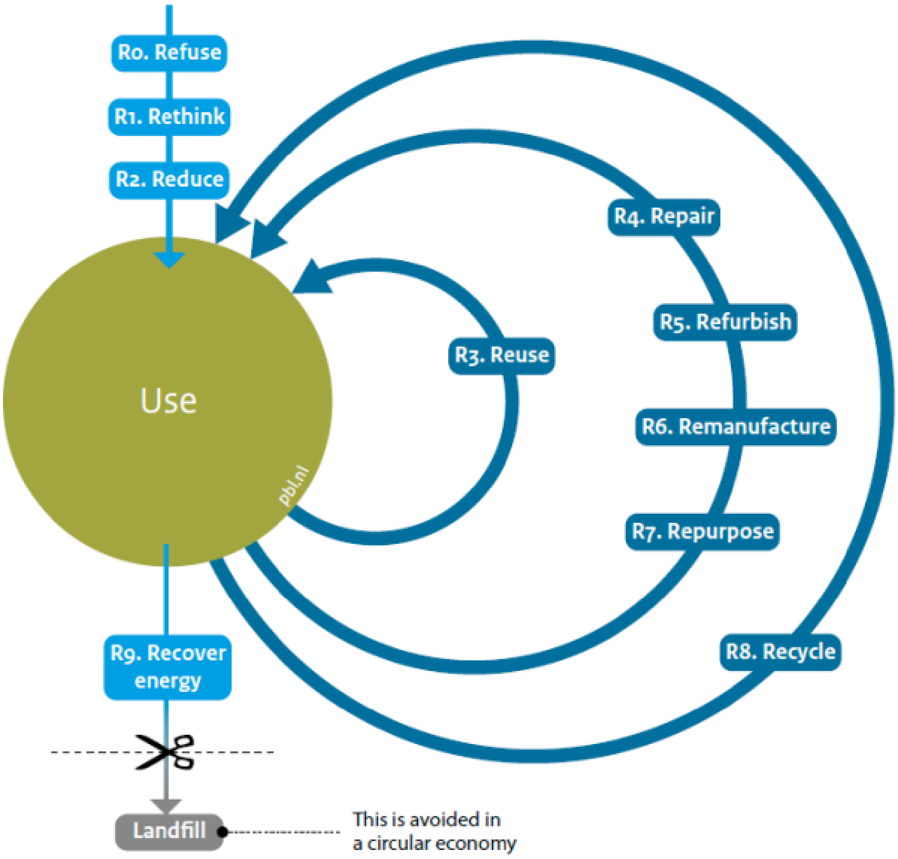


Figure 1.1: A circular economy (PBL Netherlands Environmental Assessment Agency, The Hague, 2017)

Currently, 90-95% of the metals are being recycled, which is very high (MetaalNieuws, 2022). Steel is a material that can continue to be recycled without loss of quality, which is a big advantage compared to other materials (World Steel Association, 2022). But steel is also an excellent material to be reused. Steel is compared to other building materials a very durable one. It can have a service life of 100 years when being maintained correctly (World Steel Association, 2022). Another aspect that makes steel elements excellent candidates for reuse is the fact that steel structures can be dismantled quite easily compared to structures made out of other construction materials. Steel elements are often connected by bolts and in addition it is possible to cut of end plates or trim elements to the correct length. Reusing steel instead of recycling it, has a large environmental benefit. A 50% reduction in carbon emissions can be achieved when steel reuse is increased to 75%. (LBPSight, 2022).

1.1.2. Examples of reuse of steel

In the Netherlands, the reuse of steel structural parts has already been used a few times in buildings, under the name 'Donorskelet'. The idea is to dismantle the steel structure of a building to be demolished -the donorskeleton- and use these elements to construct a new building. By designing the structure of the new building in a similar way as the dismantled building, the reclaimed steel elements can efficiently and easily be reused (Terwel et al., 2021). An example is the new laboratory for Biopartner Center Leiden whereby a neighbouring building is used as the donorskeleton (figure 1.2).

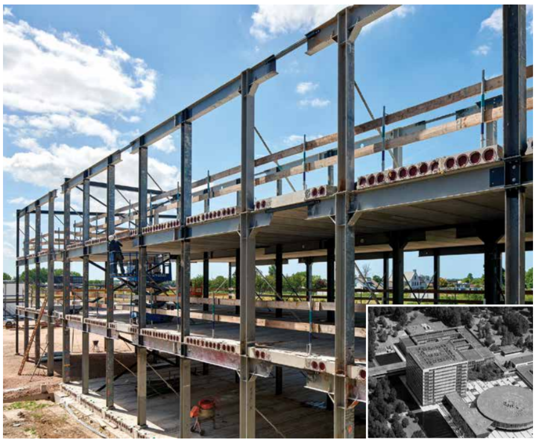


Figure 1.2: Construction of Biopartner 5, whereby steel elements are used from the neighbouring Gorlaeus-building (see below right) (Terwel et al., 2021)

An interesting project that was conducted in California is the Bay Bridge Steel Program. In this program, steel elements from the demolition of the Eastern span of the Oakland-San Francisco Bay Bridge were awarded to a mix of designers, architects and artists to be reused in a creative way (Oakland Museum of California, 2016). This way, the iconic and historic bridge could be celebrated and remembered. An example is ”The Field”, designed by CMG Landscape Architecture in which eyebars from the bridge are used to create an installation on the future shoreline park of the newly redeveloped Treasure Island (figure 1.3) (Wakefield, 2017).

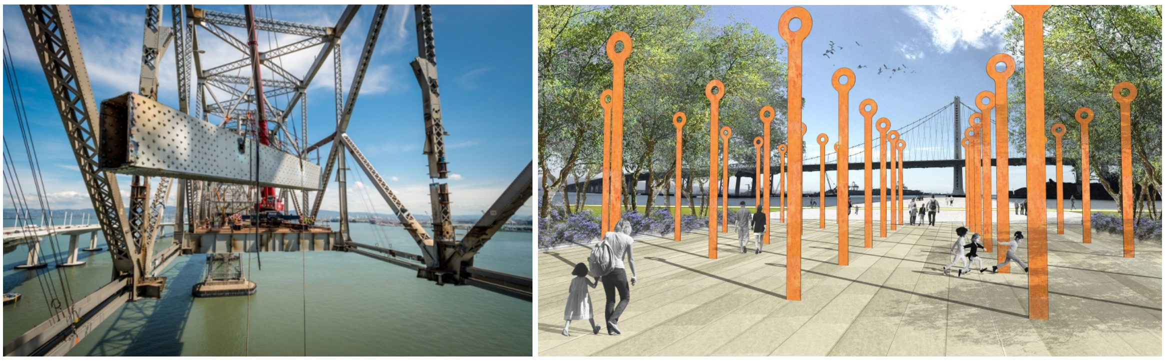


Figure 1.3: Left: Dismantling of the Bay Bridge (Oakland Museum of California, 2016) Right: The Field, an installation by CMG Landscape Architecture whereby steel bars from the Bay Bridge have been reused (Wakefield, 2017)

In Ohio, steel beams from a previously replaced bridge were reused to construct a new short span bridge (Snyder, 2022). This idea saved them $51,000 in superstructure costs (Snyder, 2020). This example shows that reused steel not only provides environmental benefits, but can also have a favourable financial aspect. Analyses in the UK show that the price difference between new steel and scrap sections, considering the long term price (2000-2016), is over £300 per tonne (Steel Construction Institute et al., 2019). Even considering additional costs like dismantling, testing, remanufacturing and storage, using reclaimed steel elements over new steel can save in costs (Steel Construction Institute et al., 2019).

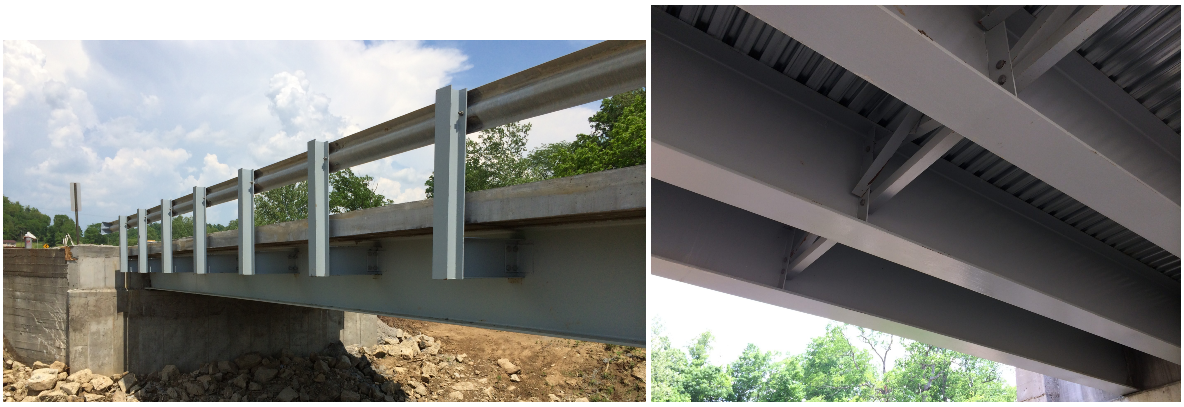


Figure 1.4: Reclaimed steel beams in new short span bridge in Ohio (Snyder, 2022)

1.1.3. Quantifying the environmental benefit

An important way to substantiate the environmental benefits of reusing steel compared to the use of recycled steel or other construction materials is to use accurate and representative quantification.

A Life Cycle Assessment is an internationally used technique to analyse the environmental aspects and impacts of a product from the procurement of raw materials to end-of-life stage which can consist out of reuse, recycling or disposal (International Organization for Standardization, 2006). The product can be an industrial process or product but also structures. It is a frequently used assessment method in the Building industry (van Maastrigt, 2019). The environmental impact that the LCA quantifies consists out of both the use of resources and environmental consequences of release (International Organization for Standardization, 2006). The large amount of different kinds of emissions are unified into several environmental impact categories. Every environmental impact category includes emissions that cause the same impact (Hillege, 2022). An example is the environmental impact category 'Global Warming Potential', which can be expressed in $kgCO_2 - equivalent$. All the environmental impact categories together determine the total impact. The total impact may be expressed in the functional unit €, which describes the cost society is willing to pay for measures to achieve each emission target (Nibe, n.d.). In the Netherlands this value is called the MKI-value (Miliekostenindicator). The term shadowcosts is also often used to describe this quantity.

Using the information obtained out of an LCA results in a more comprehensive decision process, including quantitative information about the total environmental impact of the structure (International Organization for Standardization, 2006).

1.1.4. Reclaimed steel elements stock

There are currently some steel traders in the Netherlands selling reclaimed steel elements. Some of these traders present their supply online. However, to facilitate the efficient use of available steel elements, an overarching digital database would be beneficial (Platform CB'23, 2021). This would make it easier to match supply and demand, which will promote the use of reclaimed elements. Moreover, engineers can then design from a supply of elements originating from different types of structures.

What is important to consider is that still not all profiles, lengths and dimensions may be available at all times when having such an overarching database (Gorgolewski, 2008) (Bukauskas, 2020). The supply will depend on the current structures being dismantled.

Furthermore, reclaimed elements often lack information about their test results and the used standards when being manufactured (Brown et al., 2019). In the Netherlands, a new NTA (Nederlandse Technische Afspraak) will be published early 2023 providing a standard for the reuse of steel elements. This NTA describes how structural steel can be reused in newly designed structures. More specifically, it describes the procedure to guarantee the material properties laid down by NEN-EN 1993-1-1 for new structural steel for reclaimed steel profiles.

The required information about the recovered element can then be included in a material passport, which can be presented in the digital database. Currently, multiple organizations are developing material passports, but all with a different format so that they are not comparable or unifiable (Cirkelstad, 2021). CB'23 therefore proposes substantiated principles and guidelines to set up a material passport, so that an efficient circular building industry can be obtained (Cirkelstad, 2021).

1.1.5. Problem statement

The building industry faces a big and crucial task: the drastic reduction of carbon emissions and energy use. Steel is one of the most popular building materials and in addition, steel is an excellent material to be reused. It is a highly durable material and steel structures can be dismantled quite easily. Reusing steel elements instead of manufacturing new ones from recycled steel can save energy and reduce carbon emissions. Tools are developed to quantify the environmental impact of construction materials, and as a result also the benefits that reuse can provide. These quantifications give adequate information to include in decision-making processes. Next to the environmental benefits of reusing steel, it can also save in construction costs.

Nowadays, steel reuse is not yet normalised. Almost all steel is currently being recycled. There are some examples in the world where steel is reused in a creative or simple manner. However, the full potential of reclaimed steel has not yet been reached. Factors that discourage the wide reuse of steel, including in new structures, are the unclear overall availability of reclaimed steel, the unclear material properties and the limited supply of different profiles and lengths.

1.2. Research aim

This thesis aims to overcome the bottleneck of the limited availability of reclaimed steel by investigating whether the design process can be changed to deal efficiently with the available stock and thus avoid the need for new steel or excessive overdimensioning of the structure. Instead of considering at the end of a design process whether certain elements can come from reuse, while the rest will be newly manufactured, a change in the way of designing can potentially lead to 100% use of reclaimed steel. When turning the design process around, you let the availability of reclaimed elements determine what geometry the structure should have to accommodate as many reclaimed steel elements as possible. This way, steel reuse is not limited to members that have a perfect match in the stock of reclaimed steel. What the inverted design process should look like will be investigated through literature research. Afterwards, the new design process will be developed and compared with current methods to see whether it can produce designs that make better use of the availability of reclaimed steel and whether they therefore design in a more environmentally friendly way.

1.3. Research scope

Steel element structures with highly flexible geometry, which can work as an advantage in maximising the inclusion of reused elements, are trusses. Trusses are flexible in that they can have irregular geometry and the elements can have different profiles. There is enough room to vary the geometry and the types of profiles to optimally include a limited supply of reclaimed steel elements.

This thesis will focus on the design of truss bridges. More specifically, it will look at the design process of footbridges, as they represent a safer and smaller first step to include reused elements in bridge designs for the first time. Additionally, footbridges have more structural freedom. Finally, it was decided to look at half-through truss bridge designs, as in that case the top chords of the two trusses can vary as there is no need to add overhead stabilising elements. This makes it possible to generate two unique trusses, where the top chords need not be straight. The lower chords, however, are designed straight, allowing efficient incorporation of a straight bridge deck.

Finally, this thesis assumes that an overarching digital database of reclaimed elements will be available in the future, so that the current stock of elements is available as input for the design process. As such a database does not yet exist, this thesis aims to develop a design process applicable to stocks of different sizes, including elements of different dimensions. In addition, this also allows the tool to be used for projects where steel is recovered from one specific dismantled structure, making the stockpile much smaller. The stock of reclaimed element is assumed to be tested and CE-marked.

1.4. Research question

The research question of this thesis is formulated as follows.

How can the design process of a truss footbridge be adapted such that the structural design efficiently includes reclaimed steel elements from a random and limited stock?

The research question will be answered using the following subquestions.

1. *What aspects should be considered when designing a truss footbridge?*

The design process of a truss footbridge can require certain structural checks or design constraints considering manufacturability, functionality, stability, etc., of the design. In this subquestion these aspects are summarised to give an overview on what to take into consideration when developing an adapted design process including reclaimed elements.

2. *What are the objectives of the adapted design process and which constraints need to be taken into account?*

The purpose of this question is to provide an overview of the various objectives to be optimised in the design process; most of these will relate to achieving designs with reduced environmental impact, but other viewpoints will also be considered. The list should also summarise the requirements of the design process and the design constraints to be taken into account.

3. *Which methods exist to optimise the structural design of a truss and which method best accommodates the inclusion of reclaimed elements?*

To utilise the opportunity of trusses to include irregular geometry, truss topology optimisation methods can be used to generate and compare non-standard truss designs. By changing the geometry and topology of a truss, available reclaimed elements may be efficiently included in the design. In this subquestion, several existing tools and algorithms related to structural optimisation of trusses are compared for their adaptability to stock-constrained designs. After comparing the existing methods, we should conclude if one is suited to use for our adapted design process.

With the list of objectives and constraints and the chosen method, an adapted design process can be designed which produces a design of a truss bridge, including reclaimed steel elements out of a certain stock. A conclusion to this thesis can be drawn by answering the last subquestion:

4. *Does the developed design process lead to designs that more efficiently incorporate a stock of reclaimed elements and, in doing so, result in designs with reduced environmental impact?*

1.5. Structure report

The remainder of this thesis is structured as follows. In chapter 2, preliminary research is presented which led to the research objective and scope of this thesis. In this chapter the following broader question is answered: ”How can steel be reused in bridges?”. This chapter also includes a literature review on the influence of the safe use of reclaimed steel on the design process. In chapters 3, 4 and 5, further literature study is presented, answering the subquestions of this thesis. In chapters 6 and 7, the new design method is presented. Chapter 8 specifies the input considered in this thesis and presents a design example and three analyses on the performance of the tool. The thesis is concluded with a comparison of the design method with other methods (chapter 9) and a chapter presenting the conclusions and recommendations (chapter 10).

2

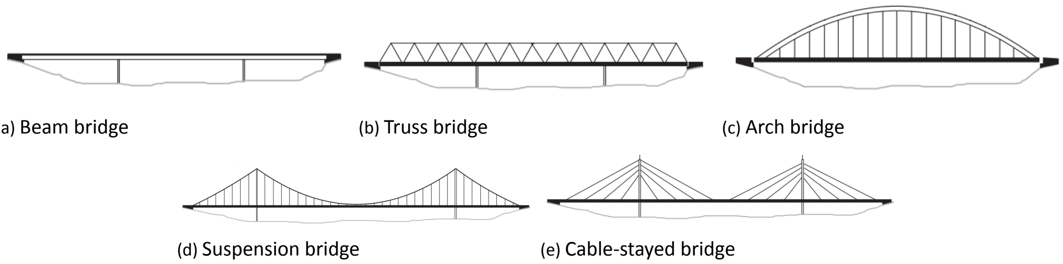
Reuse of steel in bridges

In this chapter, a literature review is presented on how steel can be reused in bridges and more detailed motivation is provided for the considered research aim and scope.

2.1. Types of profiles in steel bridges

In this paragraph a short description is given of the common types of steel bridges and how they are constructed. It will provide an overview of the types of profiles that are used in bridges in order to consider what steel elements can be reused and in what way.

There are 5 main steel bridge types that are used for most bridges world wide. These are: beam and girder bridges, truss bridges, arch bridges, suspension bridges and cable-stayed bridges (see figure 2.1). A short summary of the characteristics of each type of bridge is given below, with a focus on what types of elements are included.


(a) Beam bridge

(b) Truss bridge

(c) Arch bridge

(d) Suspension bridge

(e) Cable-stayed bridge

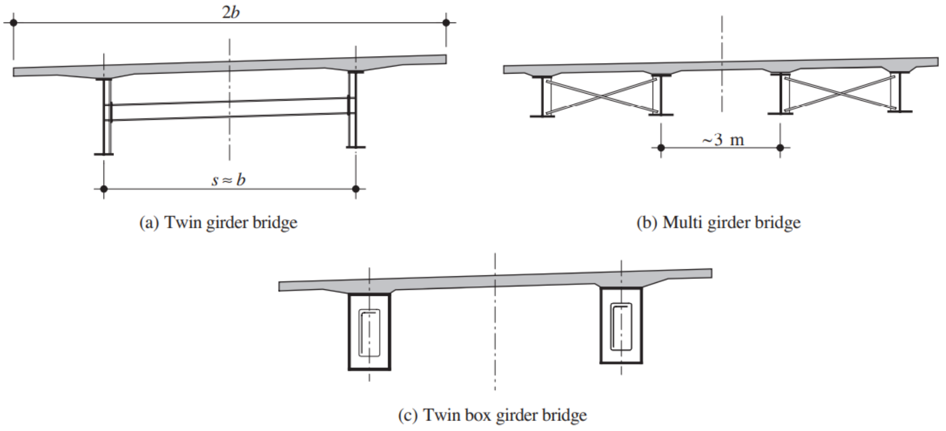
Figure 2.1: Main types of bridges (Hirt and Lebet, 2013)

2.1.1. Beam and girder bridges

In multi-beam bridges multiple parallel beams lay across the span of the bridge, whereupon a bridge deck is constructed. It is also possible to construct the bridge using fewer but larger beams and adding transverse beams whereupon the deck may lie. These are called deck girder bridges (Historic Bridge Foundation, 2021).

The girders can have open sections (I-profile) or closed sections (box-girder). Box girders are preferred when a large span needs to be achieved. The open sections may be rolled sections, but more commonly applied are plate girders as larger dimensions can be created and thus longer spans bridged. Rolled sections can only achieve a span up to 25 metres (Hirt and Lebet, 2013).

When multiple open sections or small box sections are used, the cross section of the bridge is called an open section (see figure 2.2). When a single- or multi-cell box girder is used, the bridge has a closed cross section (see figure 2.3) (Hirt and Lebet, 2013).

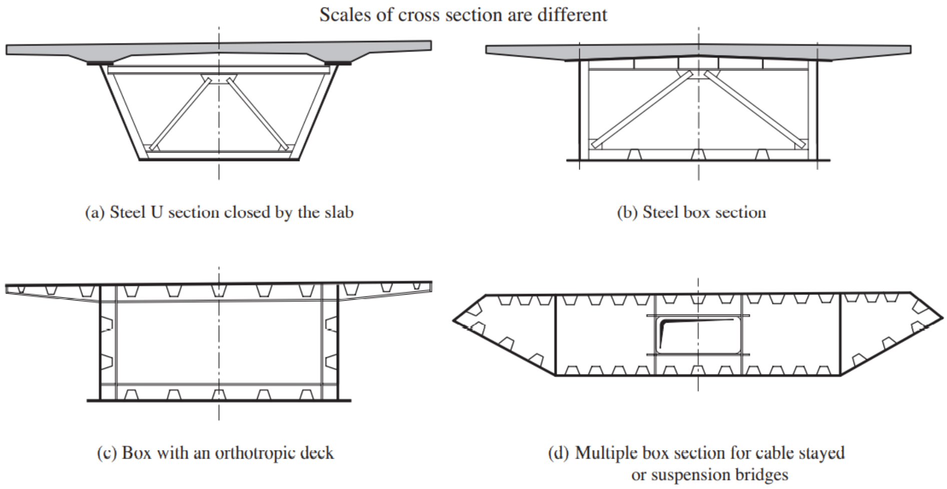


(a) Twin girder bridge

(b) Multi girder bridge

(c) Twin box girder bridge

Figure 2.2: Examples of open cross sections (Hirt and Lebet, 2013)



(a) Steel U section closed by the slab

(b) Steel box section

(c) Box with an orthotropic deck

(d) Multiple box section for cable stayed or suspension bridges

Figure 2.3: Examples of closed cross sections (Hirt and Lebet, 2013)

Plate girders should have a minimum thickness of 10 mm to assure durability and provide enough thickness during fabrication of the section. Dimensions of the flanges should be calculated with local instability taken into consideration. The depth of a plate girder can vary along the span of the bridge to adapt to the amount of stress that it has to take up at a certain location. The slenderness of the girders (ratio span/depth) can be in between 40 and 50 in span and around 20-25 at the intermediate supports (Hirt and Lebet, 2013).

In case of a composite deck, the concrete slab will function as the upper flange, meaning that the upper flange of the steel girder may be less wide than the lower flange (Hirt and Lebet, 2013). The same counts for orthotropic steel decks, which may function as the top flange of a box girder (Mangus and Sun, 2000).

In box girders welded stiffeners should be applied to stabilize the flanges when in compression. The stiffeners can be flat plates, angles or box sections. The thickness of the webs and flanges of box girders can vary between 10 to 50 mm, depending on the function (Hirt and Lebet, 2013).

Plan bracing is needed to transfer horizontal forces to the supports. This can be done by using horizontal trusses or moment resisting frames. The trusses can be located at the top or bottom of the bridge cross section and the moment resisting frames are at the level of the cross girders (Hirt and Lebet, 2013).

2.1.2. Truss bridges

In truss bridges, trusses function as the large beams where in between a bridge deck is carried by transverse beams or a slab (Historic Bridge Foundation, 2021). The truss can have different types of configurations, all a different combination of diagonal, horizontal and vertical members. Most of them use a triangular configuration to transfer tensile and compressive forces through the members towards the supports. Because nearly only normal forces occur in the members of the trusses, it is one of the most efficient ways of using material. The members may have open or hollow sections.

2.1.3. Arch bridges

In arch bridges the deck is carried by cables suspended from an arch. The arch transfers the compressive forces to the abutments at an angle. It is also possible to include a tie in the bridge deck, connecting the ends of the arch so that there are no horizontal forces that have to be resisted by the abutments. Another configuration is when the arch lies below the bridge deck and the bridge deck is not suspended by cables, but supported by columns (Historic Bridge Foundation, 2021).

The arches must be able to resist high compressive en bending forces. The second moment of inertia must therefore be sufficiently large. A rolled hollow section or fabricated box section may be used. For short spans, rolled open sections can be used, resulting in a easier connection with the hangers. When no bracing is included between the arches, the out of plane buckling of the arch must be resisted with a large enough second moment of inertia in this direction (Hirt and Lebet, 2013).

Hangers often consist out of bars with threaded ends. Another option is to use cables. The hangers may be suspended vertically but can also have an inclination (Hirt and Lebet, 2013).

2.1.4. Suspension bridges

In suspension bridges the bridge deck is also suspended by cables. But these cables are now connected to catenary main cables which stretch out over the span between tall towers. These towers transfer the forces to the ground. To the sides of the bridges, the main cables are anchored into the abutments. The cables consist out of load bearing strands with a protective casing wrapped around (Historic Bridge Foundation, 2021).

2.1.5. Cable-stayed bridges

In cable-stayed bridges the suspension cables connect the bridge deck directly to the supporting tower. The configurations of the connecting cables can vary.

2.2. Reuse of steel sourced from bridges to be dismantled

In the Netherlands, there are a lot of bridges, and a large part of them are 40 to 50 years old (Bruggenbank, 2021). These old bridges are not strong enough for the current traffic intensity and loads and they do not fulfill the current requirements (Bruggenbank, 2021). As a result, these bridges need to be strengthened or replaced.

The question thus arises what should happen to these old bridges. As discussed in the introduction, steel can efficiently be recycled, but reuse of steel elements is much more beneficial in terms of environmental benefits and costs. For bridges, it is possible to even go one step further: reuse the bridge as a whole. This is what two platforms in the Netherlands now make feasible, the Nationale Bruggenbank and the Bruggenbank of the collaboration of Machinefabriek Rusthoven (specialist in processing steel), de Koninklijke Oosterhof Holman (design, advice, realization and maintenance of infrastructural issues) and Royal HaskoningDHV (engineering consultancy) (Bruggenbank, 2021) (Nationale Bruggenbank, 2022). On these platforms, whole bridges are recorded that do not fulfill their current functional requirements anymore, but could be reused under lower traffic intensity and loads. Also bridges that need to be replaced due to broadening of the underlying waterway or highway are presented in the database. These bridges are still sufficient to fulfill their current function but on another location. The platforms facilitate the whole process of reusing bridges, from contact with the original owner of the bridge to the positioning of the bridge on its new location, including the execution of necessary adjustments and design of the new foundation (Bruggenbank, 2021). The platforms also perform an inspection of the technical conditions of the bridge, to decide whether the bridge may be reused and under what conditions. In this inspection fatigue damage is also examined (Bruggenbank, 2021).

The Nationale Bruggenbank (a collaboration between the municipalities of Amsterdam, Rotterdam and Rijkswaterstaat) made a guideline to summarise the steps that need to be followed when reusing bridges. It includes the bottlenecks and managerial, policy and organizational points of attention that need to be acknowledged (AmRoR, 2021). They emphasize that bridge reuse is not yet commonplace, but that it is the goal and to reach this goal it is necessary to keep collecting obtained observations and spreading them through such a guideline. An example of a bottleneck included in the current guideline is that it can be challenging to overlap the schedules of the offering party and the requesting party. Schedules can also often change shortly beforehand. Therefore, flexibility from both parties is important. In addition, you also have to deal with several clients or authorities, each with their own interests, decision-making, responsibilities and project organizations. One of the suggestions in this guideline is therefore to set up an umbrella management team with people from both parties involved, so that all these aspects can be streamlined.

Another bottleneck that the guideline discusses is the transportation of the bridge to its new location. For example, it is not always obvious that the new site will be accessible by water. In addition, an extra location along the route needs to be found where the bridge can be stored temporarily if needed and where maintenance and adjustments can be made so that the bridge can fit into its new location (AmRoR, 2021).

Besides reusing the whole bridge, an option would also be to reuse parts of an old bridge, like the bridge deck or the superstructure. The rest of the design would then depend on the reclaimed part.

In some cases, large span bridges need to be replaced, for example the van Brienenoord bridge. Finding a new location for these large bridges is challenging. What is interesting to investigate is how these bridges can be taken apart so that parts of them or individual elements may be directly reused in new bridge designs. A question would be how a long span bridge can be converted into (multiple) short-span bridges. For example, large span bridges often have a deck supported by plate girders. A possibility would be to cut and weld these plates into new sections, fitting a shorter span.

The advantage of reusing steel elements directly from one, to be dismantled, bridge is that you have direct information concerning the loading history of the elements and the steel strengths and element dimensions in case calculation data of the old bridge is available. Still, it needs to be checked if any properties need to be tested.

The disadvantage is that it may not feasible to reuse elements of multiple bridges as this could become too complex in terms of time management and logistics. This constrains the flexibility of the new design.

2.3. Reclaimed steel database

Looking further than bridges, steel elements may also be reclaimed from other types of structures. The question then arises how these elements may be stored and made available.

Proposed structure

A platform that is currently studying all aspects involved in a circular building industry is CB'23, short for 'Circulair Bouwen in 2023'. This platform was founded in 2018 to connect initiatives working on this topic and to take action in introducing concrete national, construction-sector-wide agreements before 2023 (Platform CB'23, 2018). CB'23 mentions the necessity of a digital database to efficiently match supply and demand of reclaimed structural elements (Platform CB'23, 2021). Design teams or developers should in this case record which elements are included in new structures, accompanied by material passports. This could be done with a BIM-model or Digital Twin. When a structure is to be dismantled, this data may then be presented in an online marketplace where new design teams or contractors may efficiently see which reclaimed elements are available, including all necessary information on their loading history and properties (Platform CB'23, 2021).

Material passports

Currently, multiple organizations are developing material passports, but all with a different format so that they are not comparable or unifiable (Cirkelstad, 2021). The clearer the supply of reclaimed elements for all parties involved, the quicker and better these may be reused in a new structure. CB'23 therefore proposes substantiated principles and guidelines to set up a material passport, so that an efficient circular building industry can be obtained (Cirkelstad, 2021).

The material passport that CB'23 has developed is quite broad, and can be made more specific depending on its goal and use (Platform CB'23, 2022b). Different scales of passports exist, ranging from a passport for raw materials to a passport for a complex of several structures. Multiple steps are given to specify the contents of the material passports, the first three being:

1. Defining the sector of the object (civil & hydraulic engineering sector or residential & non-residential construction sector)

2. Defining the life cycle phase that the object is situated in

3. Defining the goal of the material passport

These steps define the shortlist with required data fields in the passport. The rest of the steps go into which information needs to be collected and what kind of output format is desired (Platform CB'23, 2022b). To reclaim structural steel elements, a more specific passport is needed which includes material properties that need to be tested and fulfilled in order for steel to be safely reused. These properties are defined in European regulations. Which properties need to be declared will be further discussed in paragraph 2.4.

Involved parties

The action team of CB'23 also mapped out the various stakeholders within the construction and deconstruction chain and their roles in capturing and leveraging data (Platform CB'23, 2022b). A summary can be seen in figure 2.4.

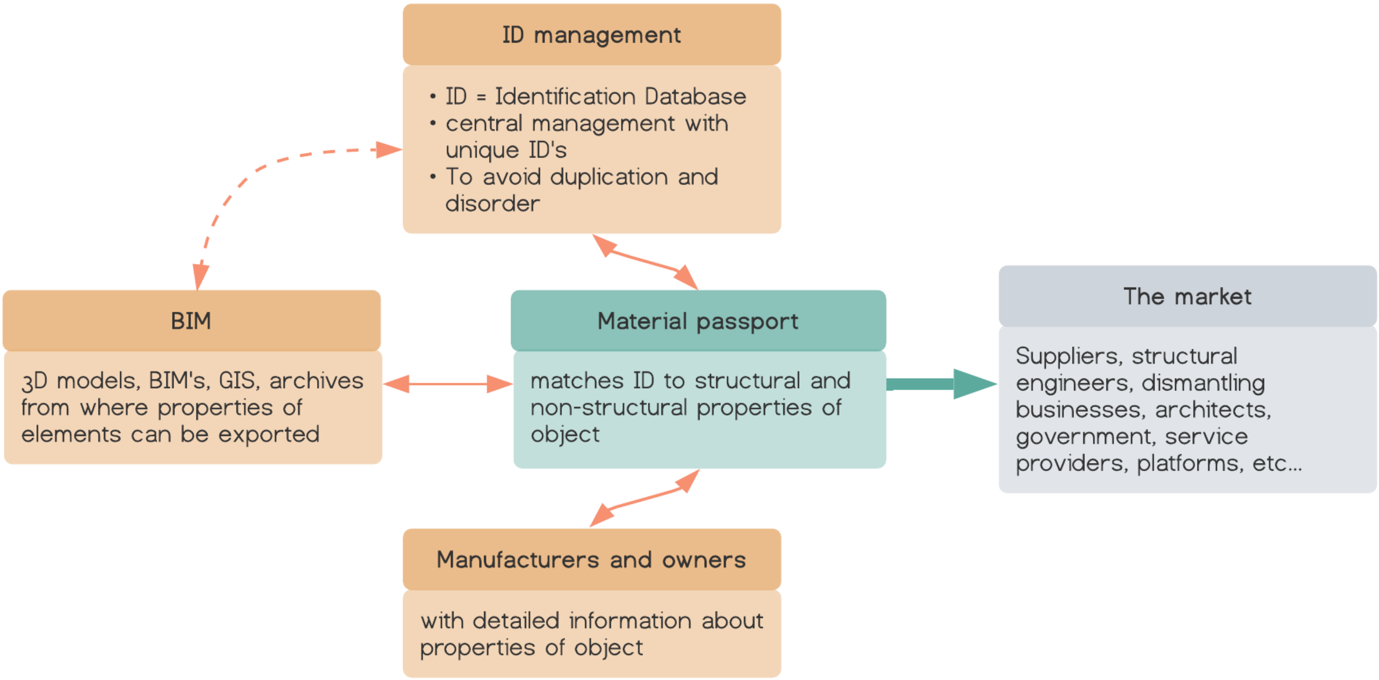

ID management
• ID = Identification Database
• central management with unique ID's
• To avoid duplication and disorder
BIM
3D models, BIM's, GIS, archives from where properties of elements can be exported
Material passport
matches ID to structural and non-structural properties of object
The market
Suppliers, structural engineers, dismantling businesses, architects, government, service providers, platforms, etc...
Manufacturers and owners
with detailed information about properties of object

Figure 2.4: Stakeholders within the construction and deconstruction chain and their roles in capturing and leveraging data (Platform CB'23, 2022b)

The orange blocks are the parties that have information about the properties of the objects. These are then summarized in a material passport, that can be used by 'the market'. In the market, also the term 'platforms' is included. These are organizations which collect or present the objects, using the standard material passport.

Platforms

There are already several platforms in the Netherlands that either aim to register the supply of reclaimed elements (Madaster) or provide an online marketplace (Insert, Oogstkaart, Materialen marktplaats, Matching Materials) (Platform CB'23, 2021).

Madaster is a digital platform where owners of real estate and infrastructure can register their assets, for which a passport is generated summarizing the included materials and products and their properties (Madaster, 2022). Their data is managed thoughout the lifetime of the building and the circular and financial salvage value of the materials and products can be provided. The owners own the data and can share it with external parties. They also make it possible to collect data from BIM's. They are still developing their material passport to make it more compliant with building regulations (Madaster, 2022).

Most of the marketplace platforms currently focus on non structural building objects as window frames, isolation, etc. Some steel traders in the Netherlands also sell reclaimed structural steel elements. Some of these traders already present their supply online. But like mentioned before, an overarching digital system would benefit the efficient match and supply of reclaimed elements and stimulate a circular building industry. The articles by C. Dunant (2018) and E. Iacovidou (2016) confirm the opportunity of increasing the reuse of elements by means of a database and a stockist.

Bottlenecks

The bottlenecks currently hindering the process of digitizing the supply of reclaimed elements are the lack in a financial business case to use material passports, the development still needed in data and technology and the required change in culture and behaviour (Cirkelstad, 2021). The latter may be aided by the implementation of regulations by the government, requiring future structures and structural components to include a material passport (Cirkelstad, 2021).

New bridge design

When designing with reclaimed steel elements out of a database, a much larger supply of elements can be made use of. For certain elements in bridges, their source does not necessarily have to be a dismantled bridge. Also other types of structures can provide in the right steel sections. With the help of a digital database, the engineer knows what kind of elements are available and can report which ones he wants to reserve. The engineer can design the best geometry, with a mix of different kind of elements from different sources. What is important to consider, is that still not all profiles, lengths and dimensions may be available at all time when provided with a reclaimed elements database (Gorgolewski, 2008) (Bukauskas, 2020).

2.4. Aspects to consider when reusing steel

Reusing steel comes with material properties that can differ from newly produced elements. The reclaimed elements have a loading history, they have been exposed to various weather conditions and they may have deviating dimensions due to tolerances, assembly, permanent deflections or the way they were dismantled. In this chapter research will be done on which properties need to be tested or recorded and what the existing regulations are. Lastly, the influence of these deviating material properties on the design process will be researched.

2.4.1. Protocol for the reuse of steel

The Steel Construction Institute (SCI), an independent organisation providing expertise in the steel construction industry, has developed a protocol for the safe reuse of structural steel with information on how to collect data and inspect and test material characteristics. In addition, the protocol gives recommendations when designing with reclaimed steel (Steel Construction Institute et al., 2019). Only reuse of steel that was not subjected to fatigue is considered in this protocol. This means that steel elements originating from steel bridges are left out in this protocol. Furthermore, the steel elements need to originate from after 1970, have no plastic strains, have no significant loss of section due to corrosion and they should not have experienced extreme loads such as fire or impact.

The protocol focuses on including reclaimed steel elements in structures in the UK, but the document is also applicable to countries using the Eurocode suite of Standards. In the Netherlands, a new NTA (Nederlandse Technische Afspraak) will be published early 2023 providing a standard for the reuse of steel elements.

In the flowchart below a summary is given of the advised steps in reusing steel, as discussed in the protocol.

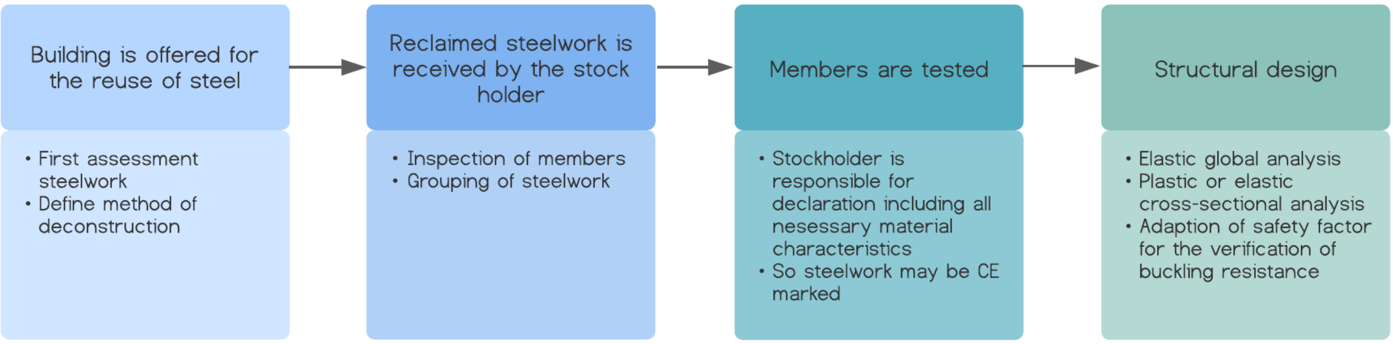

Building is offered for the reuse of steel
• First assessment steelwork
• Define method of deconstruction
Reclaimed steelwork is received by the stock holder
• Inspection of members
• Grouping of steelwork
Members are tested
• Stockholder is responsible for declaration including all nessesary material characteristics
• So steelwork may be CE marked
Structural design
• Elastic global analysis
• Plastic or elastic cross-sectional analysis
• Adaption of safety factor for the verification of buckling resistance

Figure 2.5: Steps in reusing steel (Steel Construction Institute et al., 2019)

In the protocol, the following considerations are mentioned:

Before the dismantling of the structure, an initial assessment of the steelwork should be performed to collect the following data:

- A description of the structure and its use, this should include a description of how the building is stabilised
- The age of the structure, which may be from records, or local information
- A preliminary listing of the steel members
- A preliminary inspection of the members for damage, obvious repairs, significant corrosion
- Any evidence of plasticity (Steel Construction Institute et al., 2019, p20)

When the stockholder receives the reclaimed elements, he is responsible to inspect and record the following properties:

- The dimensions (cross section and length)
- The straightness (assessed against the tolerances)
- Any significant loss of section
- Any signs of damage, or plastic strain (Steel Construction Institute et al., 2019, p20)

The protocol recommends that elements are then assembled in groups, whereby the testing of 1 element is sufficient to establish certain material properties. The elements in a group should thus have the same serial size, detailing (length, connections, etc.), original function and source (Steel Construction Institute et al., 2019).

2.4.2. CE Marking

In July 2014, it became mandatory for all structural steelwork placed on the European market for the first time to have a CE-marking according to EN 1090 (Steel Construction Institute et al., 2019). It declares that the steel meets product standards. The ’future reuse’ guideline by CB’23 expresses that there is some lack of clarity whether reclaimed steel elements also need to comply to this regulation, as they are already in the market (Platform CB’23, 2022a). In addition CE-marking is not necessary when elements are directly reused, without intervention of a manufacturer (Platform CB’23, 2022a). But it is unclear which adjustments to elements are considered an intervention.

Reclaimed elements often lack information about their test results and the used standards when being manufactured (Steel Construction Institute et al., 2019). For non-conform structural steel, the NEN-EN 1090-2 states that the technical properties of the following list should be specified (Normcommissie 351001 ’Technische Grondslagen voor Bouwconstructies’, 2018):

a Strength (yield and tensile)

b Elongation

c Stress reduction of area requirements (STRA), if required

d Tolerances on dimensions and shape

e Impact strength or toughness, if required

f Heat treatment delivery condition

g Through thickness requirements (Z-quality), if required

h Limits on internal discontinuities or cracks in zones to be welded if required (Normcommissie 351001 ’Technische Grondslagen voor Bouwconstructies’, 2018, p25)

In addition, if the steel is to be welded, its weldability shall be declared with:

i A classification in accordance with the materials grouping system defined in CEN ISO/TR 15608 or;

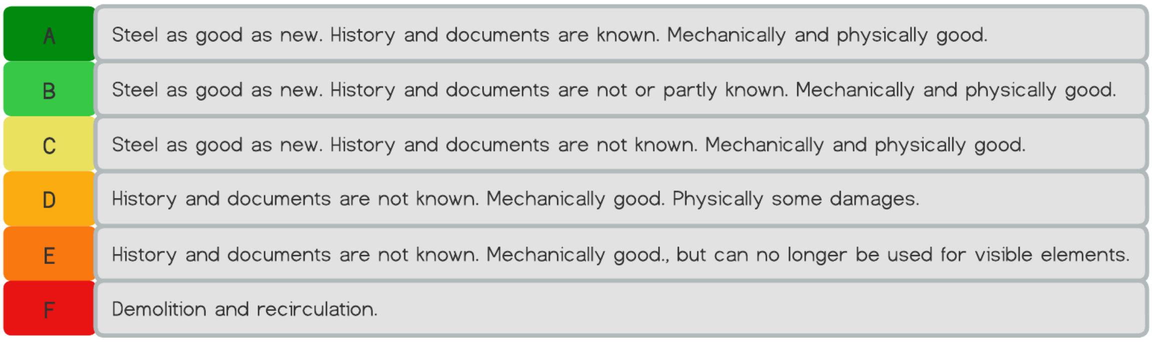
j A maximum limit for the carbon equivalent of the steel, or;

k A declaration of its chemical composition in sufficient detail for its carbon equivalent to be calculated (Normcommissie 351001 ’Technische Grondslagen voor Bouwconstructies’, 2018, p25).

The protocol of SCI expresses that the documentation of these properties must be included by the stockholder when selling the material (Steel Construction Institute et al., 2019). Once an EN 1090-certification is obtained, the CE-marking can be requested (kiwa, n.d.).

A test regime is described in the protocol by SCI to declare these properties. The test regime is more extensive when reclaimed steel is to be used in a Consequence class 3 structure. The regime is based on dimensional survey, by non-destructive tests, destructive tests or by making conservative assumptions (Steel Construction Institute et al., 2019).

The protocol 'future reuse' by CB'23 advises that a standard assessment method can be developed which differs for structural steel being reused for the first, second or third time (Platform CB'23, 2022a). They also give an example on how reclaimed steel elements may be categorised by stockholders, each with a different level of uncertainty in material properties (figure 2.6). Such a categorization could also help in defining which test regime is needed.



A	Steel as good as new. History and documents are known. Mechanically and physically good.
B	Steel as good as new. History and documents are not or partly known. Mechanically and physically good.
C	Steel as good as new. History and documents are not known. Mechanically and physically good.
D	History and documents are not known. Mechanically good. Physically some damages.
E	History and documents are not known. Mechanically good., but can no longer be used for visible elements.
F	Demolition and recirculation.

Figure 2.6: Reclaimed steel categories (Platform CB'23, 2022a, p60)

A current problem is that practice shows that for reclaimed steel elements the CE-marking is more difficult to obtain or maintain, the compliance with NEN-standards is not feasible, or that it is difficult to demonstrate equivalence (Platform CB'23, 2021). To encourage the circular building industry, current regulations need to be updated by the government so that it becomes easier to obtain and maintain a CE-marking for reclaimed elements (Platform CB'23, 2021).

2.4.3. Existing connections

When connections are reused, the welds in particular should be carefully inspected and tested (Steel Construction Institute et al., 2019). The steel grade of the connecting plates and other fittings must be tested by non-destructive tests and the steel elongation assumed to be at least the same as the members (Steel Construction Institute et al., 2019).

2.4.4. Plasticity and fatigue

The protocol of the SCI recommends that reclaimed steel elements should not be used in structures where fatigue can occur or in plastically analysed structures (Steel Construction Institute et al., 2019).

Through careful visual inspection it should be assessed whether a member has not undergone any plastic deformation so that the residual strain and reserves of ductility are still as for new steel (Steel Construction Institute et al., 2019).

To have an idea of how much plasticization occurs in structures, the following numbers can be followed: for elastically analysed structures, only 0.3 % of the elements will have been plastically deformed after 25 years, for plastically analysed structures, this will be a percentage of 5% (Nijgh, lecture slides, February, 2021).

2.4.5. Structural design

First of all, the protocol of SCI recommends to only undertake elastic global analysis when designing with reclaimed steel, as plastic global analysis needs a high level of ductility (Steel Construction Institute et al., 2019). For cross-sectional resistance, reclaimed steel is assumed to be sufficiently ductile to use plastic properties.

The protocol recommends that the only adaption that needs to be made is to the safety factor to verify buckling resistance: $\gamma_{M1,mod} = 1.15\,\gamma_{M1}$, with the current value of γ_{M1} in the Eurocode being equal to 1. The buckling resistance of an element depends on its strength, cross sectional dimensions and initial imperfections, the latter determining the buckling curve to choose. As these properties all need to be declared according to the list presented in NEN-EN 1090-2, the design procedure including reclaimed elements should not immediately need to change. However, some uncertainty is still present when reusing structural steel members. The members may for instance have torsional imperfections, increasing the second order effects. In addition one can assume that assessment processes are less reliable than those performed during the continuous production of new steel, which will be more automated and calibrated. This is why the protocol does recommend an increase of the safety factor used to verify buckling resistance. The increase is based on changing the target reliability index (β) from 3.8 to 4.3, for a 50-year reference period (Steel Construction Institute et al., 2019).

γ_{M0} and γ_{M2} may stay the same as the cross sectional resistance depends on the profile dimensions and the material strength, both tested according to NEN-EN 1090-2.

2.4.6. New connections

Next to the properties of the material, attention should also be given to how reclaimed elements can best be connected in a new design. Welding could for example not be a good idea in terms of existing internal stresses or the way the old steel is produced. However, in the list of properties that need to be declared, ”a maximum limit for the carbon equivalent of the steel” or a ”declaration of its chemical composition” is needed so that appropriate welding procedures can be defined (Steel Construction Institute et al., 2019). Furthermore, bolted connections are possible and may be preferred as they result in a structure which can again be easily dismantled, making way for a third reuse of elements.

2.5. Overview reuse of steel in bridges

When reusing steel in steel bridges, three overarching steps can be distinguished. First of all the origin of the reusable steel. From which sources can reclaimed steel elements be collected? The next important step is the collection of all the required properties needed to safely design with reclaimed elements. Lastly we have the design phase, wherein reclaimed elements are incorporated in designs of new bridges.

In the first phase, one could directly look at existing old bridges which do not fulfill their current requirements anymore. Platforms like the Nationale Bruggenbank and the Bruggenbank from RHDHV present these bridges on their website. From these platforms a bridge may be picked. The advantage of reclaiming elements from 1 structure is that you have direct information on how the elements were loaded in the past. The platform performs a thorough technical assessment of the bridge to examine its reuse capacity. When a bridge cannot be used for its current function anymore, a possibility could be to deploy the bridge with a weight restriction, such as using it for a lower traffic class.

The most efficient way of incorporating reuse on an old bridge is by reusing the whole bridge. Relatively minor alterations are then needed to let the bridge fit into its new location. There are some managerial and logistical bottlenecks and points of attention that need to be considered when undertaking such a project, such as a flexible time management, the appointment of an umbrella management team and planning the transportation and temporary storage site of the bridge.

Next to reusing the bridge as a whole, it is also possible to reuse only parts of the old bridge, for example the superstructure or the bridge deck. The reclaimed part will then determine the design of the rest of the structure.

When old bridges are too large to find another suitable location or function, research could be done to determine whether the bridge can be dismantled in such a way that its elements can be used in a different kind of configuration, resulting in smaller span bridges.

Going back to the first phase, it is also possible to retrieve elements from a stockholder with a digital element database, including elements from all different types of structures. The SCI-protocol states that no elements from bridges may be reclaimed as these can be subjected to fatigue. However, this is a topic that could be further researched so that the requirement can be made more specific, as writing off this large source of materials may be too restricting.

These databases need to be further developed so that an efficient way can exist to retrieve elements, test and store them, and present them on an online platform for engineers to pick and choose from. Retrieving elements from this system gives the opportunity to choose from a broad supply of elements, all stored and ready to be transported.

Essential is that these elements have all been tested and that all important properties are summarised in a standard material passport in order for the elements to be allowed to be incorporated in new designs.

Current bottlenecks in this process is that the use of material passports needs a financial business case, it needs to be required through the government and in addition, further development in data and technology is needed to make the collection and exchange of data possible. Furthermore, current regulations need to be updated so that obtaining and maintaining a CE-marking for reclaimed elements is more feasible. As CB'23 expressed, it should also become more clear if and when reclaimed steel elements need to be CE-marked.

Assuming these databases and material passports will be more developed in the future, the possibility emerges for an engineer to 'shop' in the database to match available reclaimed elements to designed members with the same properties.

2.6. Change in design process

Motivation

Two points of interest emerge that show that changes to the standard design process may be desired when designing with reclaimed elements.

The first one is the limited supply of reclaimed steel elements. The supply will depend on the current structures that are dismantled. This is a big difference compared to having the possibility to choose between all types of standard elements as is the case in a standard design procedure. Instead of considering at the end of a design procedure whether certain elements can come from reuse, while the rest will be newly manufactured, a change in the way of designing could possibly lead to 100% use of reclaimed steel.

The second interesting point follows from the first and can be summarised as the lower flexibility of changing a design when it is dependent on the available reclaimed elements. In the current design process, a distinction is made between the early design stage and the later design stages where elements may have slightly been changed in dimensions due to additional structural checks that have been performed. When all dimensions are available, these changes are minimal and easy to carry out. However, when the availability of elements is set, it is not so obvious anymore that small changes can be fulfilled. This results in a less flexible later design stage, making structural checks quite important in the early design phase when designing with reclaimed elements.

Turning the design process around

The first point of interest may be further explored. Clear is that a change in design process could benefit the reuse of steel elements. But how could this look like? The aim would be to let the availability of reclaimed elements decide what kind of geometry the structure should have in order to include as much reclaimed steel elements as possible. You would actually turn the design process around. This way, the reuse of steel will not be limited to members that have a perfect match in the reclaimed steel stock.

Truss bridge

Unfortunately, this is more challenging for certain types of bridges than for others. A type of bridge that is well suited to have a flexible design process is a truss bridge. Trusses can be constructed using different types of sections and they can have different kinds of configurations, as long as tension and compression can be transferred in an efficient way through the bars of the truss towards the supports. The bars may have different lengths and dimensions and they do not necessarily have to be repeated in the design. There is enough room to vary the geometry and the types of profiles to optimally include a limited supply of reclaimed steel elements.

Topology optimisation

To make use of the flexibility of a truss design, it is necessary to find a way to generate different topologies whereby irregularities are allowed. In other words, simply varying the height of the truss or the number of triangles puts a constraint on the flexibility that a truss geometry can provide, making it more difficult to include reclaimed elements with different dimensions and lengths. This is why looking into truss topology optimisation methods could be interesting. Multiple methods exist that can generate different truss topologies in a smart and unique way. For these methods to work you need to set certain constraints on the design space and define an objective, for example the minimisation of material volume.

Research gap

Already a couple of research projects have been performed on this stock-constrained truss topology optimisation problem, including the master thesis of van Gelderen (2021), research projects by the Structural Xploration Lab of the École Polytechnique Fédérale de Lausanne (EFPL) (2021) and the PhD thesis by Bukauskas (2020). These research projects have been focused on optimising the topology of 2D trusses using one specific topology optimisation method: the ground structure method. The presented optimisation processes still have some aspects which may be improved, especially to make them applicable to the design of truss bridges. The research projects focus on a truss in a 2D-plane and only consider vertical point loads. When designing trusses in a truss bridge, some important aspects need to be considered that are not covered in the simplified scope of the research projects mentioned above. These are instabilities, non-uniform distributed loads (directed out of plane) and manufacturability of connections between the elements and with the bridge deck. To really apply stock-constrained truss topology optimisation to a real life case, this research gap needs to be closed so that realistic and applicable designs are generated as output. This goal is supported by the fact that an early design stage in which reclaimed elements are included, should already incorporate the most important structural checks, as explained above. This means that information should be gathered on the required structural checks of a truss bridge, as well as on other aspects of truss bridges and reclaimed steel that should be considered in the adapted design process.

Furthermore, as mentioned above, multiple topology optimisation methods exist that aim to generate the most efficient truss design. van Gelderen (2021) and Brütting, Senatore, et al. (2018) use the ground structure method in their research projects, but it is relevant to research the other methods in their adaptability to stock-constrained design to see which method works best for the maximum inclusion of reclaimed steel or more general: the reduction of the environmental impact of the resulting designs.

2.7. Choice of bridge

In order to obtain a concrete research scope, one more focus will be defined.

Bridges can be divided into large scale road and railway bridges and smaller footbridges. With footbridges an overarching term for pedestrian and cycling bridges. For the first group it can be assumed that clients or engineers will have some hesitancy towards using reclaimed steel elements as these bridges have a much higher traffic intensity and load to carry and they have bigger consequences when something goes wrong.

Footbridges can be a safer and smaller first step to introduce the incorporation of reclaimed elements in new bridge designs. In addition, because of the smaller span, footbridges provide the opportunity to have more creative freedom in the structural design. This way, the geometry of the bridge can be more easily varied in order to match and fit a random set of reclaimed steel elements. Lastly, the demand for new footbridges will only grow as cities are focusing on making transportation by foot or bike more accessible and appealing in and around the city (Gemeente Rotterdam & Goudappel Coffeng, 2017) (Gemeente Amsterdam, 2017).

Although truss bridges are very efficient in terms of strength versus material use, they are not always the most popular type of structure in terms of aesthetics. Especially because footbridges are often located in areas where a lot of people pass during the day, clients like to give attention to the look and story behind a design. However, the possibility to incorporate reclaimed steel elements in a truss bridge and thereby obtain an environmentally low impact structure will definitely be attractive to reach the sustainability goals of a city while simultaneously presenting an interesting design story.

2.8. Conclusion

The conclusion of the question on how steel can be reused in bridges is summarised in the flowchart below (figure 2.7). It includes the defined research scope that the rest of the thesis will go into.

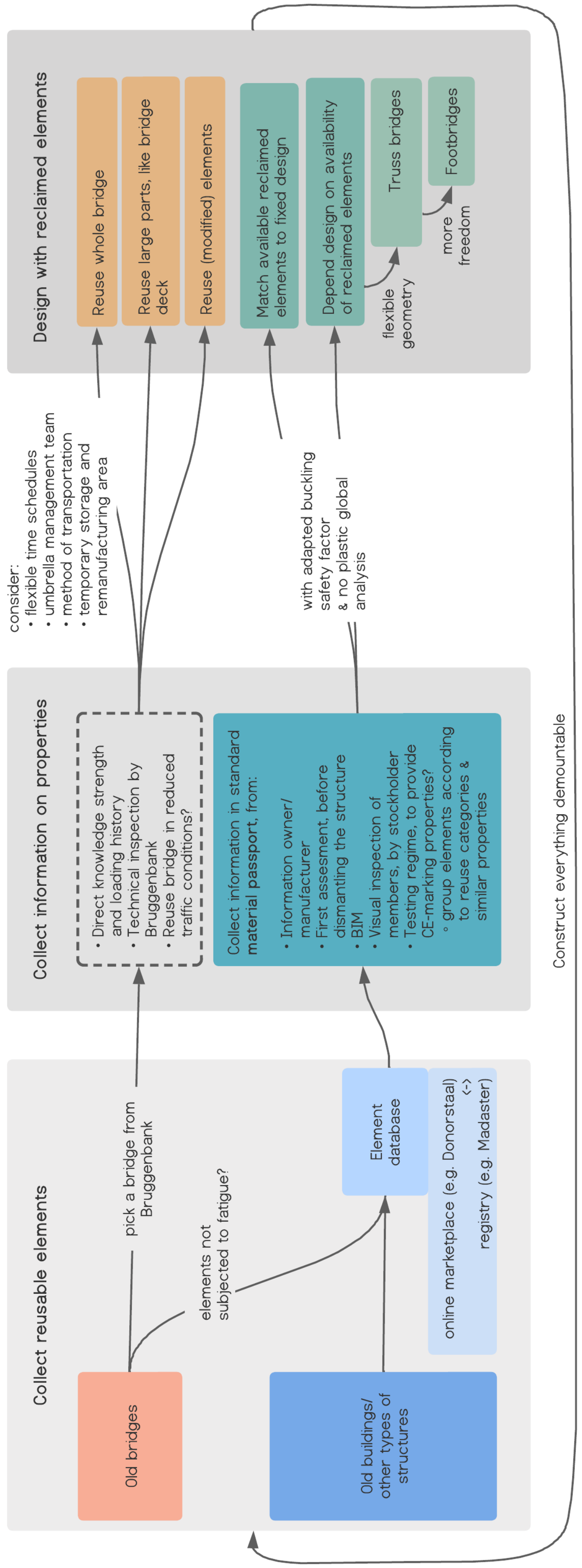

Collect reusable elements
Old bridges
Old buildings/ other types of structures
pick a bridge from Bruggenbank
elements not subjected to fatigue?
Element database
online marketplace (e.g. Donorstaal) <-> registry (e.g. Madaster)
Collect information on properties
• Direct knowledge strength and loading history
• Technical inspection by Bruggenbank
• Reuse bridge in reduced traffic conditions?
Collect information in standard **material passport**, from:
• Information owner/ manufacturer
• First assesment, before dismantling the structure
• BIM
• Visual inspection of members, by stockholder
• Testing regime, to provide CE-marking properties?
◦ group elements according to reuse categories & similar properties
consider:
• flexible time schedules
• umbrella management team
• method of transportation
• temporary storage and remanufacturing area
with adapted buckling safety factor & no plastic global analysis
Design with reclaimed elements
Reuse whole bridge
Reuse large parts, like bridge deck
Reuse (modified) elements
Match available reclaimed elements to fixed design
Depend design on availability of reclaimed elements
Truss bridges
Footbridges
flexible geometry
more freedom
Construct everything demountable

Figure 2.7: How can steel be reused in bridges?

3

Truss footbridges

In this chapter the first subquestion will be investigated: What aspects should be considered when designing a truss footbridge?

3.1. Configuration and Dimensions

Different types of configurations exist for truss bridges. The most commonly applied is the (modified) Warren truss (steelconstruction.info, 2022). Two other options are the Pratt or Howe truss. In a Warren truss only diagonal member are present, forming triangles. The modified Warren truss has vertical members inside of these triangles to divide the span of the lower chord in two. The Pratt or Howe truss consists out of vertical members and diagonal members, in one direction only. See figure 3.1 for a visualization of these different types of configurations.

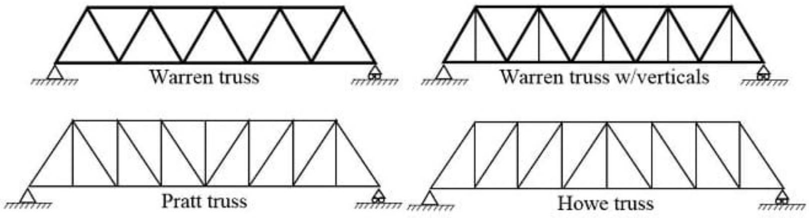


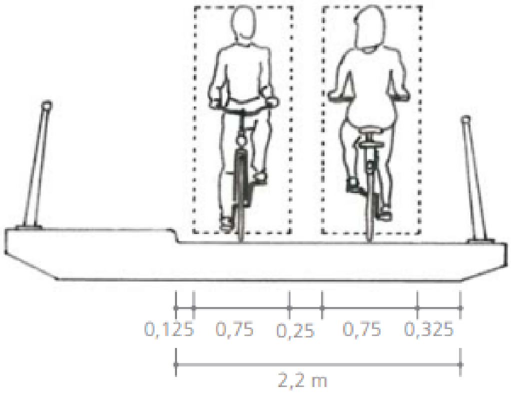
Figure 3.1: Diagrams of four types of truss engineering (Ramirez, 2019)

Vertical members make connections and thus fabrication more complex, however, the deck can create large bending moments in the lower chord in a Warren truss, so dividing up the spans by including vertical members can help lower the bending moment that the member needs to resist (steelconstruction.info, 2022). In addition, vertical members are sometimes preferred in terms of alignment with vertical posts of the parapet or to make fixing of cladding easier (steelconstruction.info, 2022). These are standard configurations, but when changing the configuration according to the supply of available elements, these aspects may still be important in finding the most optimal design, depending on the client's wishes.

Furthermore, truss bridges may be subdivided in through or half-through structures. In the through structure, the vertical trusses are connected by braces at the top, above head level (steelconstruction.info, 2022). In the second variant, the half-through structure, the horizontal bracing at the top is missing. This structure is usually used for small-span bridges where the height of the required truss is lower than clearance height for pedestrians (steelconstruction.info, 2022). The two variants differ in the way that horizontal stability is ensured and lateral forces are transferred to the supports, this will be further discussed in paragraph 3.3.

What is important to consider is how these two types of structures can adapt to the supply of reclaimed elements. In a through truss, the heights of the trusses need to be equal so that top bracing is possible in a logical way. This results in the constraint that from each reclaimed steel profile, 2 elements are needed. One for each truss, so that identical trusses can be designed. This is why for this thesis the choice is made to focus on half-through trusses, whereby the two trusses do not need to be identical. A boundary condition here is that the deflections of the two trusses should not differ too much so that uncomfortable twisted deformations and unfavourable stresses are avoided.

Lastly, regulations are in place in the Netherlands that determine required dimensions of footbridges. The width of the bridge must be at least 1,5 m, but preferably 1,8 m for pedestrians (ipv Delft, 2015). For cyclists, the minimum width increases to 2,4 m (ipv Delft, 2015). These dimensions include a safety margin to the railing, of two times 0.325 m (ipv Delft, 2015). When combining functions, one can summarise required widths like shown in figure 3.2.



Bicycle path width	
Basic width for two cyclists	1,75 m
Safety margin cyclist - railing	0,325 m
Safety margin cyclist - raised pavement > 0,050m	0,125 m
Total width	2,2 m

Figure 3.2: Two way bicycle and pedestrian bridge (ipv Delft, 2015)

As an input to the design process, a required width can be defined by the client or the width can be defined as a variable parameter, whereby one can choose between functions of the bridge, summing up the required dimensions given above.

Sometimes, a ramp is needed to overcome a height difference. A ramp may not be too difficult to cycle, so regulations are set in place defining the maximum 'difficulty' of the ramp. The following formula is given to calculate the difficulty Z, whereby $\frac{H}{L}$ is the average grade of the ramp (ipv Delft, 2015).

$$Z = (\frac{H}{L})^2 * L = \frac{H^2}{L} \tag{3.1}$$

The ideal difficulty of the ramp is set on $Z = 0.075$ with a maximum grade of 7.5% and a minimum grade of 1.75% (ipv Delft, 2015). This target difficulty value, together with a possible bandwidth for the grade of the ramp, is shown in figure 3.3. The eventual slope will depend on amongst others the location, available space and budget (ipv Delft, 2015).

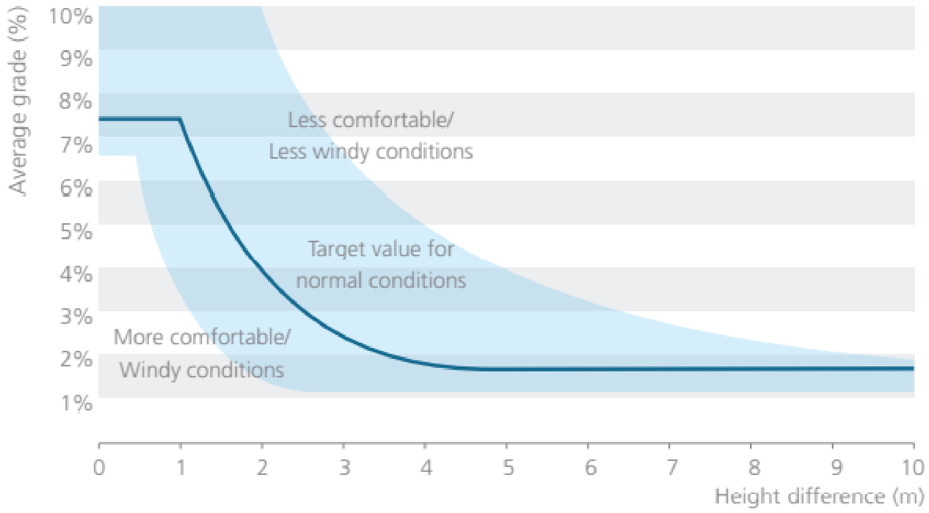


Figure 3.3: Slope bandwidth (ipv Delft, 2015)

3.2. Profiles and Connections

3.2.1. Overview of the possibilities

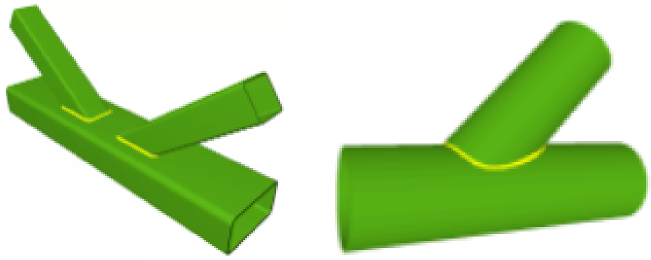
Profiles

Both reclaimed open and hollow sections can be used in a truss design. However, according to Wardenier et al. (2002), the following advantages apply to using hollow sections:

- Better mechanical properties in terms of buckling resistance, torsion and bending in all directions
- A lower drag coefficient when exposed to wind load
- Rounded corners and smooth transition from one section to the other, resulting in a more uniform coverage of the coating preventing corrosion, resulting in a longer protection period
- A 20-50% smaller surface to be painted compared to open sections
- Closed surfaces, preventing rainwater and dirt from building up at the truss joints, a problem that will happen when using open sections and affect the durability of the joints
- Possible aesthetic preference to open sections

Apart from these advantages, a disadvantage of hollow sections is that they are more expensive per unit material than open sections (Wardenier et al., 2002). However, when only looking at reclaimed elements, this may differ.

Important is that these sections can be efficiently connected to each other. This means that the horizontal members must have an equally or larger width than the diagonal or vertical members in order for them to be welded on to these horizontal members. Usually, horizontal members are constructed with rectangular hollow sections as these facilitate easier connections (steelconstruction.info, 2022). Rectangular hollow sections and open sections have the advantage that when welding them onto each other (or welding circular hollow sections to these profiles), the elements can have a straight end cut (Wardenier et al., 2002). The difference between a straight end cut and rounded end cut can be seen in figure 3.4.



(a) Straight end cut

(b) Rounded end cut

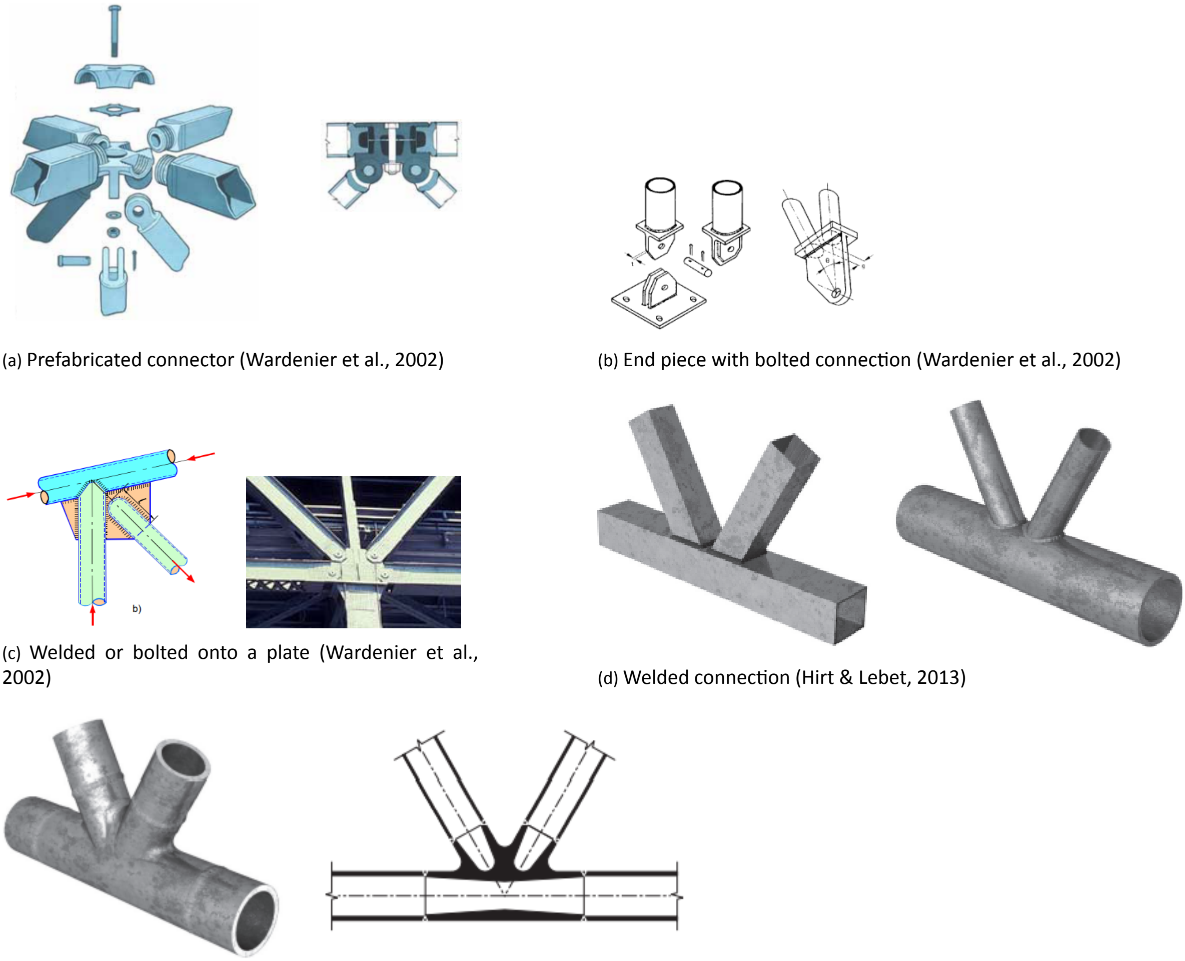
Figure 3.4: Types of end cuts (cype, n.d.)

Connections

A welded connection is also the most economical and common way that connections in trusses with these type of sections are made, as maintenance and protection is simplified (Wardenier et al., 2002). Other types of connections are prefabricated bolted connections (3.5a), welded end pieces with bolted connections (3.5b), members being welded or bolted together by plates (3.5c) and cast connections (3.5e) (Wardenier et al., 2002) (Hirt & Lebet, 2013).

The bolted connection options have the advantage that they can be dismantled easily, facilitating a third reuse of the steel elements. In addition, transportation and erection may be easier when using bolted connections. However, end plates, needed to facilitate these bolted connections, still need to be welded onto the elements.

An overview of the different types of connections can be seen in figure 3.5.



(a) Prefabricated connector (Wardenier et al., 2002)

(b) End piece with bolted connection (Wardenier et al., 2002)

(c) Welded or bolted onto a plate (Wardenier et al., 2002)

(d) Welded connection (Hirt & Lebet, 2013)

(e) Cast connection (Hirt & Lebet, 2013)

Figure 3.5: Truss joints

A grasshopper plugin called KarambaIDEA may be used to incorporate steel joint design in a design process. However, like designing with new elements, detailing of connections may become more relevant in a later stage, when the topology optimisation of the truss is finished. In the first stage, the positions of members in trusses are usually designed such that their centrelines intersect with each other (steelconstruction.info, 2022).

What does need to be considered is how the connections are initially modelled. The members in a truss are usually modelled with hinged connections as this is the most conservative approach. Even when connections are welded, you may not assume that these are moment resisting connections.

Furthermore, the Eurocode describes a minimum required angle between welded hollow sections so that the presented formulae to calculate the stresses in the connection may safely be used. When an angle falls outside of this range, additional calculations are required to check its resistance.

Lastly, what may be included in the adapted design process is the choice to minimise the number of connections or define a limit to this number. When limiting the number of connections less fabrication is needed, resulting in less costs and less embodied carbon.

Composition of the bridge deck

Going on from connections between members of a truss, a short description can be given of the rest of the structure of a truss bridge. The deck of a truss footbridge can be constructed using steel plates supported by cross girders between the chords or by using precast planks supported by steel angles welded on to the lower-chords (steelconstruction.info, 2022). When using the steel plate, the cross-members together with the trusses form U-frames which stabilize the top-chords in a half-through construction. Sometimes, stiffeners are used in between the cross-members (steelconstruction.info, 2022).

3.2.2. Choice of profiles and connections for final truss design

With the developed algorithm, a truss will be generated with a straight bottom chord and diagonals and top chord elements all set at a unique angle and all with a different profile. Considering this final design, the following choices have been made for the included profiles and connections. These choices influence the restrictions needed in the algorithm to generate a manufacturable design. These will also be mentioned.

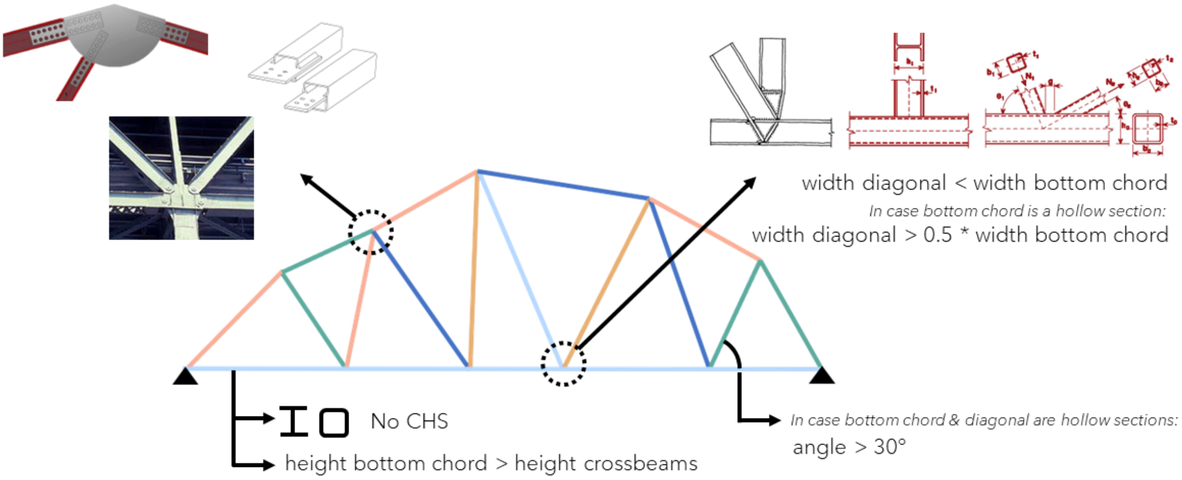

width diagonal < width bottom chord
In case bottom chord is a hollow section:
width diagonal > 0.5 * width bottom chord
No CHS
height bottom chord > height crossbeams
In case bottom chord & diagonal are hollow sections:
angle > 30°

Figure 3.6: Specific connections for final truss design (Tekla, 2021), (Packer, 2018), (NSC, 2019), (ESDEP, 2012), (Normcommissie 351001 'Technische Grondslagen voor Bouwconstructies', 2011b)

For the bottom chord, the choice has been made to only allow open sections or rectangular hollow sections, so that easy straight end cuts are possible. The bottom chord will be as continuous as possible, meaning that the number of elements that need to be welded onto each other to form the bottom chord will be minimized. This way the number of connections is minimized en thus the energy and costs to manufacture them. Another advantage of making the bottom chord as continuous as possible, including the aim to use same profiles, is that the connection of the bridge deck is made easier. When applying a higher number of bottom chord elements, with different profiles, tailor-made connection designs would have to be made for each different element. The last constraint on the selection of the bottom chord is that its height needs to be larger than the height of the crossbeam profiles, so that these may be connected more easily and logical

When the bottom chord elements have the same profile, the most economical way to connect them is to weld them onto each other. When the bottom chord consists out of different profiles, they may be welded onto an endplate and bolted onto each other, or they can welded onto the same endplate. Welding will save costs and energy, while the bolted connection will make dismantling easier.

The diagonals, with different sized profiles, are then welded onto the continuous bottom chord. To make these welds possible the width of the diagonals may not be greater than the width of the bottom chord. In addition, when the bottom chord consists out of open sections, stiffeners need to be added so that the force can be transferred through a larger area. On the other hand, when the bottom chord consists out of rectangular hollow sections, a minimum width of the diagonals needs to be considered so that the RHS is not too susceptible to chord failure. Above this minimum width, the risk is smaller and may be checked using Eurocode formulas. The Eurocode prescribe a minimum width of half the width of the supporting RHS (Normcommissie 351001 'Technische Grondslagen voor Bouwconstructies', 2011b). Lastly, it is chosen to allow all angles in the generation of the truss. This choice has been made as the lower boundary condition on angles between welded connections only apply to hollow sections. Also open profiles will be included in the design and setting a boundary condition on the allowed angles will constrain the design flexibility. Additionally, when angles smaller than 30 °occur between hollow sections, these may still be present as long as the more thorough calculations are performed. A solution is to check whether such situations occur in the final design so that preference may be given to other designs.

Furthermore, the diagonals and top chord elements all have different profiles so the best way to connect them is by using connection plates on to which each profile may be bolted. The advantage of this type of connection is firstly, that all elements are connected in one point, avoiding too large eccentricities. Secondly, the fact that different sized profiles can come together in a joint, and lastly, that bolted connections can be easily dismantled again. This option does not require any additional constraints in the algorithm.

Another option would be to weld the top chord elements to an endplate, which is possible under an angle and also possible for different profile heights when stiffeners are added. Subsequently, the diagonals can be welded onto the top chord profiles, considering the same constraints as discussed for the connection of the diagonals to the bottom chord. As these constraints restrict the freedom of the algorithm and lack the advantages of the first option, the first option is preferred when possible and therefore chosen for the algorithm. In case the second option is to be chosen, these constraints can be added to the algorithm.

Lastly the choice has been made to evaluate the resulting trusses on, among others, the number of elements and whether the design consists of any hollow sections being welded under an angle lower than 30 °. When choosing a truss design with the least number of elements, the amount of connections will be smaller which may be desirable in terms of costs.

3.3. Stability

First of all, the difference between the through- and half-through truss will be discussed. The through truss has horizontal braces at the top, connecting the two trusses. The bracing system ensures horizontal stability of the trusses as well as the transfer of lateral forces, like wind loads, to the supports (steelconstruction.info, 2022). This is done by ensuring portal frame action in the ends or by including braced frames (steelconstruction.info, 2022). When using a half-through construction, the global lateral torsional stability of the trusses should be checked as the trusses are not horizontally supported along the top-chord (steelconstruction.info, 2022). In addition, special attention needs to be given to the connections of the lower chord to the cross-members as these influence the stability of the truss (steelconstruction.info, 2022). For the lower chord, either a steel floor plate as deck ensures the transfer of lateral forces, or the addition of cross bracing in the deck (steelconstruction.info, 2022).

As the choice has been made to focus on half-through trusses, the check of the global lateral torsional stability of the trusses will be further explored.

In standard trusses with a top chord in compression, this member will tend to buckle outwards. So called ”U-frames” provide intermediate lateral restraints along the length of the top chord (steelconstruction.info, 2022). These U-frames consist of the cross-members of the bridge deck connected to the vertical or diagonal members of the trusses. In figure 3.7, the U-frame action is visualized and in figure 3.8 the U-frames of two types of trusses are shown.

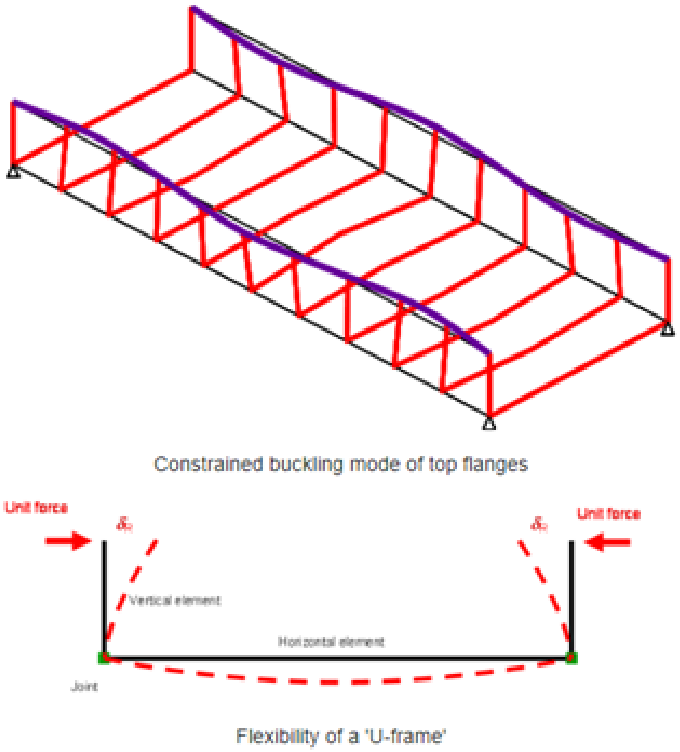


Figure 3.7: U-frame action (steelconstruction.info, 2022)

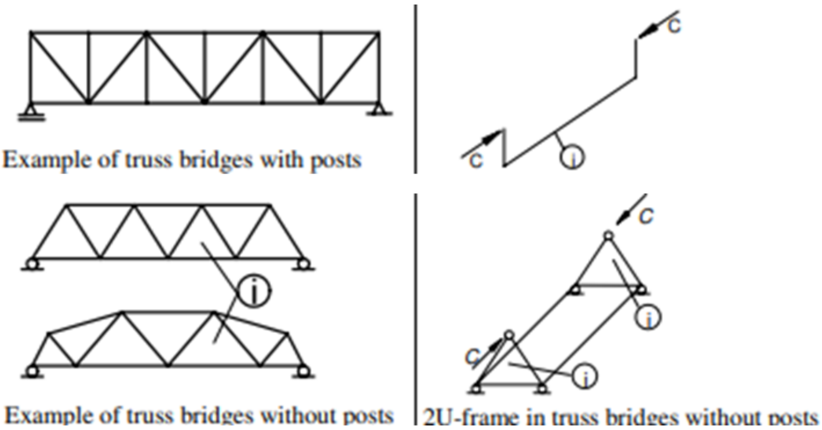

Example of truss bridges with posts
Example of truss bridges without posts
2U-frame in truss bridges without posts
C
C
C
C
i
i
i
i

Figure 3.8: U-frames of trusses with and without posts (Normcommissie 351001 'Technische Grondslagen voor Bouwconstructies', 2017)

In NEN-EN 1993-2, clause 6.3.4.2 a simplified method is given to determine the buckling load of an edge bar of a truss in compression, whereby lateral constraints (in this case the U-frames) are modelled as springs (Normcommissie 351001 'Technische Grondslagen voor Bouwconstructies', 2017). The stiffness of these springs, in case of U-frame action, can be calculated using table D.3 situated in appendix D, clause D.2.4 of the same code. The table can be found in appendix A of this thesis. The critical axial load can subsequently be calculated using the following formula's (Normcommissie 351001 'Technische Grondslagen voor Bouwconstructies', 2017):

$$N_{crit} = m\, N_E \tag{3.2}$$

Where

$N_E = \pi^2 \frac{EI}{L^2}$

$m = \frac{2}{\pi^2}\sqrt{\gamma}$ but not less than 1.0

$\gamma = \frac{cL^4}{EI}$

$c = \frac{C_d}{l}$

where

L is the span length between the rigid supports

l is the distance between the springs

C_d is the spring stiffness, calculated using table D.3 in NEN-EN 1993-2

Notice that in these formula's rigid supports are mentioned. This is of course not the case for the top chord of a truss, since the end U-frames will not be rigid (steelconstruction.info, 2022). PD 6695-2[6], section 9 gives an adjustment to the calculation of the m parameter to correct this shortcoming (steelconstruction.info, 2022).

Another shortcoming in this Eurocode calculation is that it is only applicable to standard trusses with a continuous top chord. When the geometry of a truss is optimized and no constraints for the top chord is set, it is highly possible that the truss will obtain a non-standard geometry and this Eurocode calculation method for lateral instability will not be useable.

To then correctly calculate the maximum load so that no lateral torsional buckling will occur, a finite element buckling analysis on a 3D-model may be carried out.

3.4. Dynamics

Dynamic loads originate not only from wind loads but also, and more importantly so, from pedestrians walking along the bridge (Heinemeyer et al., 2009). It is also for these pedestrians that uncomfortable vibrations should be restrained. To what extent vibrations should be restrained is not defined in any codes (Heinemeyer et al., 2009). However, in consultation with the client, the required comfort should be discussed.

To analyse the vibration behaviour of a bridge design, its lowest natural frequency needs to be calculated. The natural frequency of a structure depends on its mass, geometry and rigidity. It can be calculated using the finite element method or by using hand formulas for beams, cables and plates (Heinemeyer et al., 2009). Dynamic loads have their own frequency range. When this excitation frequency range is lower than the natural frequency of the structure, no dynamic problems will arise. However, when the frequency of excitation reaches the natural frequency of the structure, the structure will start to resonate, resulting in large displacements (ipv Delft, 2015). The displacements will increase when the source of vibration introduces more energy than the structure can absorb, resulting in uncomfortable conditions for the pedestrians and possible structural damage (ipv Delft, 2015). Different types of vibration can occur. Lateral vibration is the most uncomfortable type for pedestrians (R. Vernooij, lecture 14 of the course Steel Bridges (CIE5125) Technical University of Delft, June 2018). Moreover, once there is lateral vibration, this will only be increased by pedestrians starting to walk in the same frequency (Nakamura & Kawasaki, 2006). In contrast, pedestrians will dampen the movement when vertical vibration is present (Heinemeyer et al., 2009).

Due to the fact that footbridges are constructed with increasingly slender designs, resulting in lower stiffnesses, they will have a relatively low natural frequency (ipv Delft, 2015). This can be made clear using the most simple description of the natural frequency of a structure:

$$\omega_o = \sqrt{\frac{k}{m}} \tag{3.3}$$

Whereby k represents the stiffness of the structure and m the mass. When the stiffness decreases, the natural frequency will also decrease. The danger of the excitation frequency reaching the natural frequency is therefore present. Already in an early design stage, the vibration behaviour of the bridge should be analysed to check if the natural frequency of the bridge falls in a critical range (Heinemeyer et al., 2009). If this is the case, a more detailed dynamic analysis should be performed. This includes defining the present traffic classes and the required comfort class, which will set a maximum allowed acceleration (Heinemeyer et al., 2009). As stated before, no limits for accelerations are set in any codes, but the required comfort is discussed with the client. The National Annex A2 to NEN-EN 1990 does advise the following maximum accelerations:

- $0.7m/s^2$ for vertical vibrations
- $0.2m/s^2$ for horizontal vibrations during normal usage
- $0.4m/s^2$ for horizontal vibrations during exceptional situations of a crowd of people (Normcommissie 351001 'Technische Grondslagen voor Bouwconstructies', 2019a)

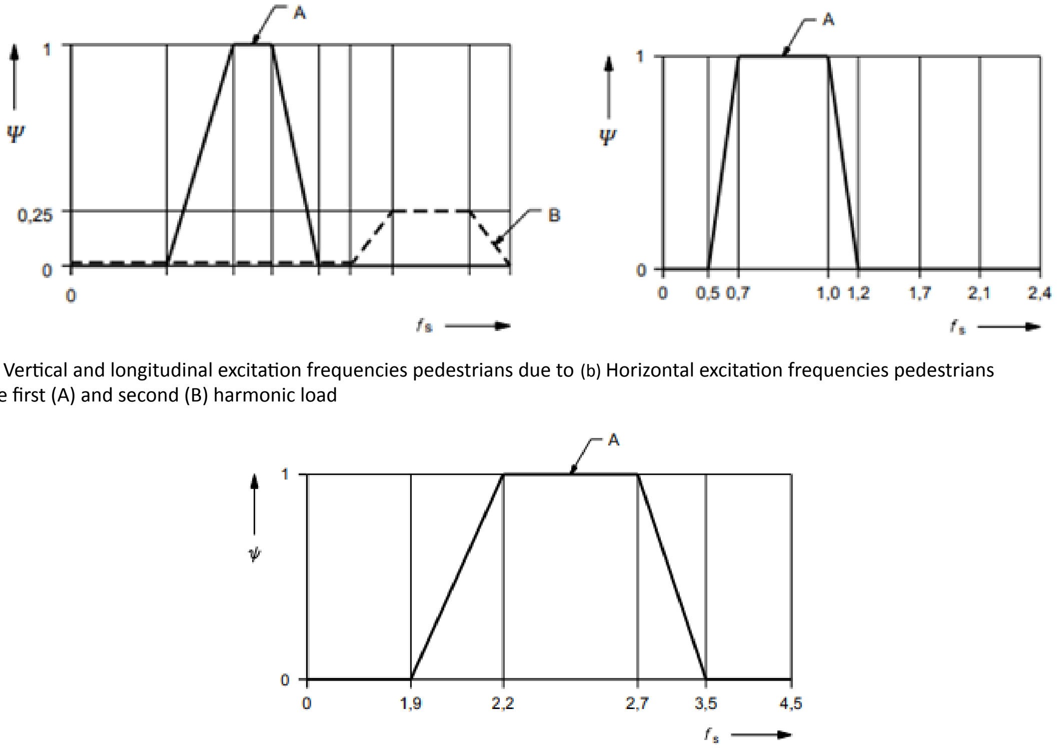
Furthermore, the structural damping parameters need to be calculated and finally, the structure needs to be checked for each design situation (Heinemeyer et al., 2009). If in the end the maximum acceleration is not reached, the bridge design is okay. If not, the structural design may be adapted to increase the natural frequency (Heinemeyer et al., 2009). A second option is to include dampers in the design, which is most often the easiest and most economical solution (R. Vernooij, lecture Steel Bridges, June 2018). As vibrations are sometimes difficult to predict and bridges can often act differently once constructed, one could also already include some extra holes in the bridge deck to make sure that if unforeseen vibrations occur, dampers can still be placed (ipv Delft, 2015)(R. Vernooij, lecture Steel Bridges, June 2018).

In this thesis truss bridges will be analysed. An advantage to these bridges is that they have a relatively high stiffness, which helps counteract the dynamic problem (steelconstruction.info, 2022).

To further reduce the chance of vibrations, the choice can be made to only allow designs in the optimization process that have a higher natural frequency than the critical range of excitation frequencies. In that case, no further analyses need to be performed and the bridge design will fulfill.

To define what this limit should be, it is necessary to go back to the critical ranges discussed above. The ranges of frequencies by different types of dynamic loads on footbridges that need to be taken into account, are given in the national annex NB.I to NEN-EN 1991-2 (Normcommissie 351001 'Technische Grondslagen voor

Bouwconstructies', 2019b). The corresponding graphs are shown in figure 3.9 and the values are summarized in table 3.1.



(a) Vertical and longitudinal excitation frequencies pedestrians due to the first (A) and second (B) harmonic load

(b) Horizontal excitation frequencies pedestrians

(c) Vertical excitation frequencies joggers

Figure 3.9: Excitation frequencies footbridge (Normcommissie 351001 'Technische Grondslagen voor Bouwconstructies', 2019b)

type of excitation frequency	critical range
Pedestrians: vertical and longitudinal	1.25 Hz - 4.6 Hz
Pedestrians: lateral	0.5 Hz - 1.2 Hz
Joggers: vertical	1.9 Hz - 3.5 Hz

Table 3.1: Excitation frequencies footbridge (Normcommissie 351001 'Technische Grondslagen voor Bouwconstructies', 2019b)(Heinemeyer et al., 2009)

In this table, the vibrations due to the first and second harmonic of vertical or longitudinal pedestrian loads are summarised in one critical frequency range. The lateral vibrations are not affected by the second harmonic of pedestrian loads (Heinemeyer et al., 2009).

From these values limit values for the natural frequency of the bridge design can be defined. The National Annex A2 to NEN-EN 1990 (application on bridges) states minimum natural frequencies that a structure needs to have to not require additional dynamic analysis:

- $5Hz$ for vertical vibrations
- $2.5Hz$ for horizontal and torsional vibrations (Normcommissie 351001 'Technische Grondslagen voor Bouwconstructies', 2019a)

The question is if a conservative value of 5Hz for vertical vibrations would eliminate too many sufficient designs. An option could be to allow designs with lower natural frequencies, and to mention them in a solution cloud of design options so that the possibility still exists to choose them when performing additional dynamic analysis.

3.5. Loads

In this paragraph, an overview will be given of the loads that need to be taken into consideration when designing a truss footbridge, according to NEN-EN 1991-2 (Traffic loads on bridges), NEN-EN 1991-1-4 (Wind actions) and the National Annex A2 to NEN-EN 1990 (application on bridges) (Normcommissie 351001 'Technische Grondslagen voor Bouwconstructies', 2015)(Normcommissie 351001 'Technische Grondslagen voor Bouwconstructies', 2011a)(Normcommissie 351001 'Technische Grondslagen voor Bouwconstructies', 2019a).

Load combinations and load factors

First of all, the following load combinations, as described in NEN-EN 1990+A1+A1/C2:2019 (eq. 6.10a 6.10b, p46), need to be considered in the calculations checking the Ultimate Limit State:

$$\sum_{j\geq 1}\gamma_{G,j}G_{k,j}+\gamma_{Q,1}\psi_{0,1}Q_{k,1}+\sum_{i>1}\gamma_{Q,i}\psi_{0,i}Q_{k,i} \tag{3.4}$$

$$\sum_{j\geq 1}\xi\gamma_{G,j}G_{k,j}+\gamma_{Q,1}Q_{k,1}+\sum_{i>1}\gamma_{Q,i}\psi_{0,i}Q_{k,i} \tag{3.5}$$

Whereby ξ is a reduction factor for unfavourable permanent actions G equalling $\xi = 0,89$. The corresponding load factors are:

Consequence Class	$\gamma_{G,unfavourable}$	$\gamma_{G,favourable}$	$\gamma_{Q,traffic}$	$\gamma_{Q,rest}$
CC1	1.2	0.9	1.2	1.35
CC2	1.3	0.9	1.35	1.5
CC3	1.4	0.9	1.5	1.65

Table 3.2: Load factors for footbridges

The ψ_0-factors to be included for footbridges are the following:

Load type	ψ_0
Traffic: uniformly distributed load q_{fk}	0.4
Traffic: uniformly distributed horizontal load q_{flk}	0.4
Traffic: concentrated load Q_{fwk}	0
Wind: uniformly distributed load q_{Wk}	0.3

Table 3.3: ψ_0-factors for footbridges

The stresses inside the structural elements are checked with the ULS design loads. Next to strength and stability, also vertical deflection needs to be checked. This is checked in Serviceability Limit State. The following load combination from NEN-EN 1990+A1+A1/C2:2019 (eq. 6.14b, p48) needs to be applied:

$$\sum_{j\geq 1}G_{k,j}+Q_{k,1}+\sum_{i>1}\psi_{0,i}Q_{k,i} \tag{3.6}$$

With this load combination the maximum vertical displacement of the bridgedeck may be calculated and checked against the maximum allowed displacement for bridge decks: $\frac{L}{800}$.

Loads

An overview of the loads that need to taken into account on truss footbridges are given in the table below.

Permanent Load	
self weight	to be calculated
Variable Load	
traffic: uniformly distributed load	$q_{fk} = 5kN/m^2$ when l > 10 m $q_{fk} = 2 + \frac{120}{(l+30)} kN/m^2 \geq 2.5kN/m^2$ and $\leq 5kN/m^2$ when l < 10 m
traffic: concentrated load	$Q_{fwk} = 10kN$ on $0.1 * 0.1m$
traffic: uniformly distributed horizontal load	$q_{flk} = 0.1 * q_{fk}$
Wind Load	
uniformly distributed load on truss	$q_{w,x} = c_s c_d c_f q_p(z_e)$ z_e: distance between the lowest ground level height to the centre of the bridgedeck $c_s c_d = 1$ when dynamic analysis is not necessary $c_f = c_{f,0}\psi_\lambda$ $c_{f,0}$: see figure 7.33 NEN-EN 1991-1-4, whereby $\phi = \frac{A}{A_c}$ $A =$ surface steel and $A_c = l * b$ ψ_λ: see figure 7.36 NEN-EN 1991-1-4, whereby $\lambda = min(1.4\frac{l}{b}, 70)$ when $l \geq 50$ and $\lambda = min(2\frac{l}{b}, 70)$ when $l \leq 15$ linear interpolation for values in between
uniformly distributed load on deck	$q_{w,x} = \frac{1}{2}\rho v_b^2 c$ $\rho = 1.25kg/m^3$ $v_b = c_{dir} c_{season} v_{b,0} = 1 * v_{b,0} = v_{b,0}^* = 23m/s$ when windload and traffic load are acting simultaneously c: see table 8.2 NEN-EN 1991-1-4 $q_{w,z} = c_s c_d c_{f,z} q_p(z_e)$ $c_s c_d = 1$ $c_{f,z} = 0.9$ $q_{w,y} = 0.5 * q_{w,x}$

Table 3.4: Loads on truss footbridges

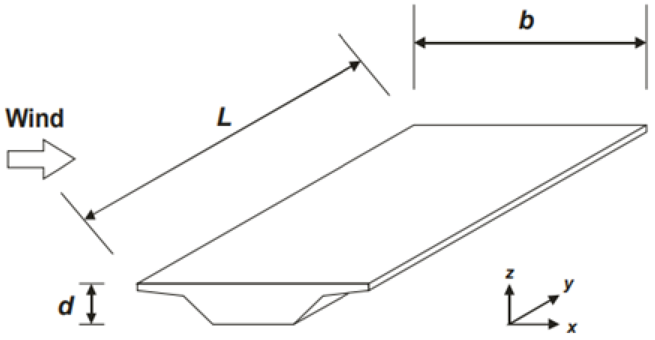


Figure 3.10: Coordinate system used for loads on bridge

Some remarks on the above loads are: the uniformly distributed traffic load in vertical and horizontal direction should be applied at the same time, whereby the horizontal load acts along the axis of the bridge, at the level of the wearing layer of the bridge deck. The concentrated traffic load is applied as a separate load case.

The uniformly distributed load has to be applied on its most unfavourable position. To define which surfaces this load needs to be applied on, influence lines of the bridge can be used. Influence lines show the effect of a unit load at a position y along the axis of the bridge, on the quantity of a force or displacement at a certain position a, wherefore the influence line is chosen (H. Welleman). In other words, for a couple of critical positions along the bridge, an influence line may be drawn to look at where the load should be applied to cause the largest stress or deflection in that position. These critical load distributions should all be applied separately on the bridge model to acknowledge the various loads that pedestrians can cause. An example of influence lines for forces in a truss is shown in figure 3.11. An influence line is drawn for 4 cases: the shear force in panel 2 (between L1 and L2) and the normal force of the diagonal in this panel and the same for panel 3 (between L2 and L3). The normalforces in the diagonals can be calculated from the shear force in the corresponding panels as their vertical component is equal to the shear force. Putting a unit load on the locations where the influence line is positive, will cause a positive force in the considered case. In contrast, a unit load positioned on a location where the influence line is negative, will cause a negative force in the considered case. The relative magnitude of the resulting force follows from the height of the influence line.

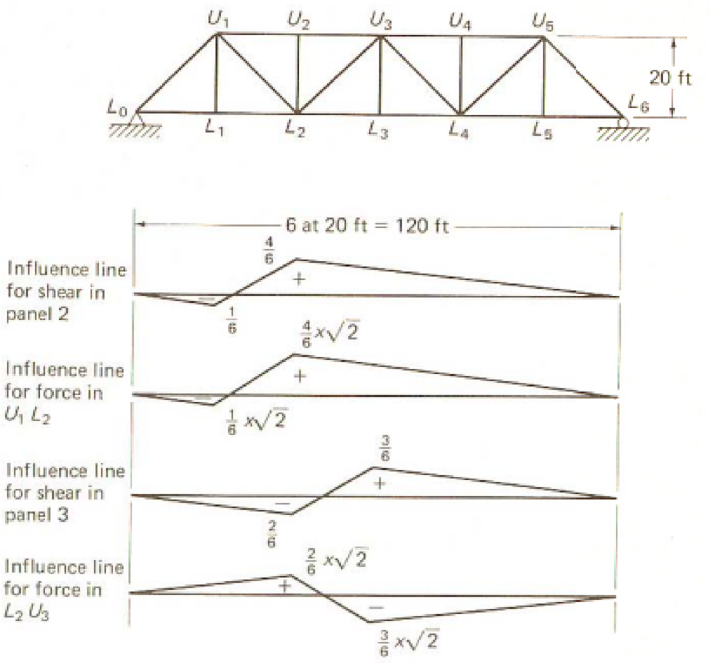


Figure 3.11: Influence lines of a truss (McCormac, 1984)

Furthermore, wind loads are applied in combination with the traffic loads using the load and ψ factors given in tables 3.2 and 3.3. The wind loads in every direction and part of the bridge are applied together.

Lastly, other loads like the weight of an unauthorized vehicle, collision loads by boats or cars and loads resulting from service vehicles may be taken into account when applicable.

Fatigue

Important to mention is that NEN-EN 1993-2 states that fatigue does not have to be considered in footbridges, as long as vibrations are not plausible (Normcommissie 351001 'Technische Grondslagen voor Bouwconstructies', 2017). As discussed above, only truss bridge designs which are not susceptible to vibrations will be explored in this thesis, which means fatigue may be neglected. In addition, in a master thesis by K. Bila (2019) it was proven that dynamic loading on a lively footbridge, even when having noticeable vibrations, did not produce high enough stresses for fatigue damage to become possible.

4

List of objectives and constraints for the new design approach

The next subquestion to answer is the following: What are the goals of the adapted design process and which constraints need to be taken into account?

Objectives

To begin with, the aim is to design a truss bridge where the percentage of included reclaimed steel elements is as high as possible. To do this, we could let the geometry of the design depend on the available stock of reclaimed elements. A second objective would be to aim for full utilisation of the available elements so that no cut-off waste is produced. When incorporating these two objectives, it is important to consider the capacity utilisation of the elements and the resulting total mass. Both aspects should also be optimised to a certain extent. An excessively overdimensioned truss bridge design has disadvantages in terms of transport logistics, aesthetics and most importantly, the magnitude of the environmental impact. Minimising the environmental impact is in the end the main objective. A final objective could be to minimise the number of elements in the design so that as few connections as possible need to be made. This helps reduce manufacturing costs and simplifies the disassembly of the structure for second reuse.

It should be noted that the objectives cannot all be optimised simultaneously. The client's priorities can be taken into account to indicate which objectives are more important than others. The aim is to create a design in which these objectives are balanced in the desired way. As an output, a solution cloud of possible designs can be given from which the best design can be selected based on the different objectives. To limit the size of this solution cloud and to include all objectives in a minimal way, limitations can be defined on, for instance, the minimum required capacity utilisation. This way, the objectives are also converted into constraints.

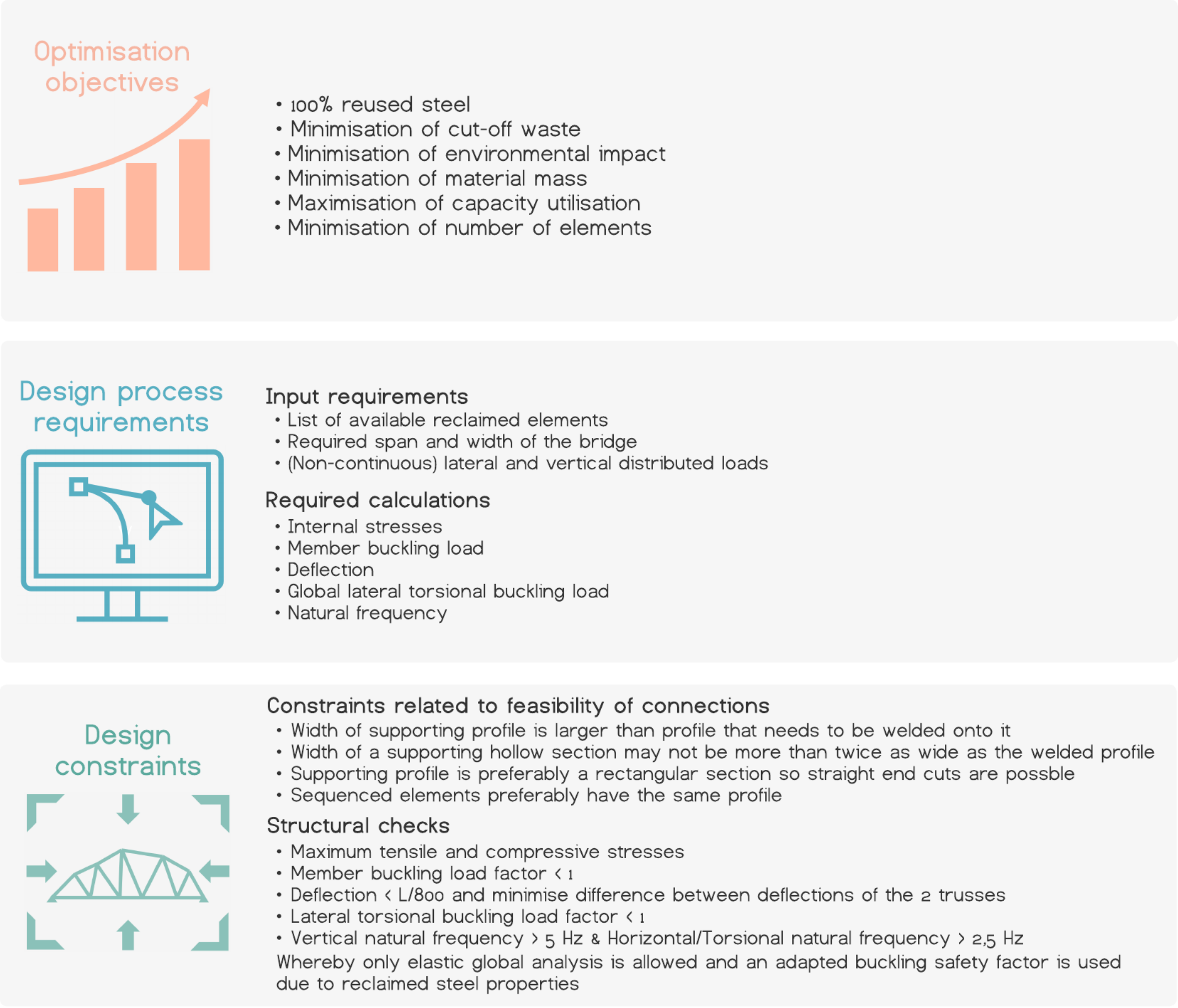
Constraints and requirements

Next to optimisation goals, certain constraints and requirements emerged from literature study that need to be taken into account. More specifically, there are a number of requirements that the design process must meet: firstly, certain input values that must be able to be entered and, secondly, certain calculations that must be able to be performed so that fully calculated truss bridge designs can be generated. Furthermore, there are a number of constraints that the designs must meet so that they can actually be built. These are constraints so that connections between all elements are possible and constraints so that the designs meet strength and stability requirements.

Important to note is that it is assumed that all reclaimed elements presented in the database will have been CE-marked. According to the Steel Construction Industry this is required and while the action team of CB'23 challenges this and it is also a possible that requirements to obtain a CE marking will be eased, for now, these standards will be assumed. With these standards, the reclaimed steel elements can be included and calculated like new elements, with the exception of an adapted buckling safety factor and the recommendation to not perform any global plastic checks.

Overview

In figure 4.1, an overview is given of the objectives, constraints and requirements that need to be included in the new design process. It summarises research done in previous chapters and gives an answer to the question on what the adapted design process should entail to efficiently incorporate reclaimed steel while satisfying design constraints. The next chapter will subsequently examine which optimisation method best fits this framework of goals and boundary conditions so that the new design process may be developed.



Optimisation objectives

- 100% reused steel
- Minimisation of cut-off waste
- Minimisation of environmental impact
- Minimisation of material mass
- Maximisation of capacity utilisation
- Minimisation of number of elements

Design process requirements

Input requirements

- List of available reclaimed elements
- Required span and width of the bridge
- (Non-continuous) lateral and vertical distributed loads

Required calculations

- Internal stresses
- Member buckling load
- Deflection
- Global lateral torsional buckling load
- Natural frequency

Design constraints

Constraints related to feasibility of connections

- Width of supporting profile is larger than profile that needs to be welded onto it
- Width of a supporting hollow section may not be more than twice as wide as the welded profile
- Supporting profile is preferably a rectangular section so straight end cuts are possble
- Sequenced elements preferably have the same profile

Structural checks

- Maximum tensile and compressive stresses
- Member buckling load factor < 1
- Deflection < L/800 and minimise difference between deflections of the 2 trusses
- Lateral torsional buckling load factor < 1
- Vertical natural frequency > 5 Hz & Horizontal/Torsional natural frequency > 2,5 Hz

Whereby only elastic global analysis is allowed and an adapted buckling safety factor is used due to reclaimed steel properties

Figure 4.1: Overview of the objectives, constraints and requirements to incorporate in the new design process.

5

Optimisation methods

Following the defined list of objectives, this chapter will now research the next subquestion: Which methods exist to optimise the structural design of a truss and which method best accommodates the inclusion of reclaimed elements?

5.1. Overview of the state of the art

Research into generating optimal truss designs have begun quite some time ago. Dorn et al. (1964) developed the ground structure method using linear programming as early as 1964. Since then a large amount of research has been conducted on different ways to find more accurate local or global optimal solutions. A global optimal solution being the solution with the highest possible objective value and local optimal solutions having the best objective value inside a certain region. Additionally, different starting points have been researched in generating various geometries. With the arrival of visual scripting environments like Grasshopper, algorithmic design has become more and more popular as engineers and architects can intuitively and easily design, while making use of smart algorithms to find optimal structures (Vierlinger & Bollinger, 2014). The users are motivated to use the power of the computer to calculate and compare many design variants to make well-considered choices. In addition, 3D printing has opened the way to construction of free-form design, so that the actual implementation of the non-conventional and often complex optimal designs is now not unthinkable anymore.

The parametric design platform Grasshopper includes already a lot of ready-to-use tools which can be used to smartly design and calculate structures. Like said above, the platform is already widely used among engineers and architects. The use of Grasshopper for the adapted design process of this thesis is therefore preferred.

As a single method or tool does not yet exist which includes most of the aspects and objectives listed in the last subquestion, inspiration will have to be found in theoretical research on truss layout optimisation. The goal is then to combine existing knowledge, developed tools and new creativity to develop a new algorithm which makes a full design process possible in Grasshopper which aligns to the objectives of this thesis.

The biggest gap in research that this thesis needs to bridge is the way that truss elements are designed in current algorithms. In current algorithms, the best truss layout is calculated by varying the dimension and length of the element, amongst others. A completely different approach is therefore needed when the availability of elements is set, namely our stock of reclaimed elements, and thus varying their dimensions cannot be a way to optimise the topology of the truss. Analysing whether an algorithm can be adapted to be applied on a stock-constrained design process is therefore the focus in the following literature review.

To start, a distinction can be made between the goal of the different algorithms needed to optimise a truss layout. A first kind of algorithm must be able to generate different kinds of truss layouts. Trusses can vary in many different ways, including the sizes of the elements, the topology of the truss and lastly the geometry or layout, also called shape optimisation (see figure 5.1). There is no simple parameter that can be varied, but instead a combination of parameters exist to describe a truss layout, and they all depend on each other. There is therefore no straightforward method of how to generate different variants, especially when wanting to cover all different possibilities, including irregular geometries, in order to find a global optimum. Various algorithms will be analysed that approach this challenge in different ways.

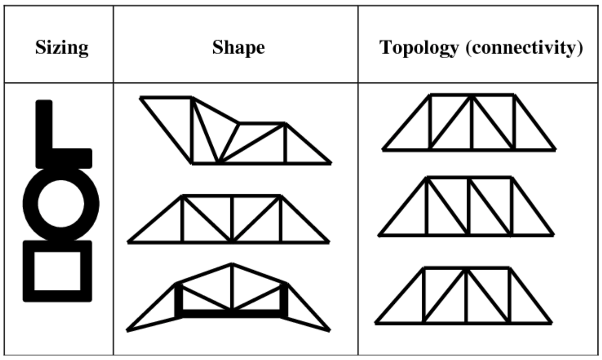


Figure 5.1: Types of optimisation (Shallan et al., 2014)

Subsequently, a second algorithm may be needed to solve the optimisation problem. Optimisation problems occur not only in the structural world, therefore a lot of different optimisation algorithms exist which all work in a general way and are applicable to all kinds of optimisation problems (Vierlinger & Bollinger, 2014). These algorithms require as an input a certain objective, described as a function that is dependent on different variables, and a couple of constraints. The algorithm then tries to find the local or global minimum of maximum value of this function, which satisfies the constraints. This is often not that straightforward, so different methods are developed to find this value. A couple of topology optimisation methods use such algorithms, which will be mentioned.

Before delving into the above algorithms, it is important to first understand what Grasshopper already has in place in terms of useful tools. This will give an idea on what is still missing.

5.2. Grasshopper tools

First of all, the finite element analysis tool Karamba may be mentioned. This tool has been developed so that structural calculation can be efficiently and quickly performed inside grasshopper, allowing a feedback loop so that a full optimisation can be performed on the basis of the load bearing performance of the design (Preisinger & Heimrath, 2014). This is important for this thesis as the desired output is a truss bridge which satisfies all structural mechanical checks. In addition, the calculated unity checks can be used in the optimisation loop to find the most efficient design.

Two other methods that perform extensive structural mechanical checks are the plugins GeometryGym and GSA-Grasshopper , the last one developed by Arup. These plugins make it possible to connect your parametric design flow from grasshopper, to a finite element program so that these can be used to perform thorough calculations. This can take more time than when using Karamba, especially when wanting to include calculation results in an optimisation loop.

For steel structures which include only 1D elements, like a truss bridge, Karamba can perform all necessary calculations accurately. Only when needing 2D or 3D elements and using other materials like concrete, may GeometryGym or GSA-Grasshopper be better alternatives.

Secondly, several optimisation tools exist to use in grasshopper. These are Galapagos, Octopus and Wallacei. These three tools all use an evolutionary solver to find the optimum solution. In such a solver the value that needs to be minimised or maximised (the objective) can be defined together with the parameters on which the objective depends and that may be varied. The solver can find a global optimum to this problem definition. The difference between the tools is that Galapagos is a single-objective solver while Octopus and Wallacei can be used for multi-objective problems. This means that when having multiple objectives, these can all be included in the optimisation algorithm. As an output, a solution cloud is given whereby the user can prioritize objectives. The corresponding design(s) will subsequently be shown.Another option would be to load all the solutions into a design explorer, e.g. Arup's 'Parameterspace'. This is an online platform which can show similar outputs as Wallacei, with the advantage that it is a website of which a link can be shared so that the solutions from the parametric design flow can easily be shared.

Karamba and the optimisation tools are plugins with a lot of potential to use in this thesis, they are well developed and perform a lot of the desired steps. The question is how these tools can be connected to additional steps to complete the new design process.

5.3. Generating truss geometry variants

The most challenging part of this thesis is how various truss geometries can be generated including only elements out of a prescribed stock. Research on optimising truss layouts can be divided into four categories:

- Ground structure methods
- Explicit topology optimisation methods
- Growth methods
- Voxel based topology optimisation methods

These methods vary in their approach to generate different topologies.

5.3.1. Ground structure methods

Most fitting is to start with the first topology optimisation method that was developed. This was the ground structure method, which was developed in 1964 by Dorn et al. The ground structure method has been further researched in many papers and adapted to different kinds of problems, but they all include the same idea. The ground structure method starts off with a grid of nodes, which are all connected vertically, horizontally and diagonally. The nodes can be connected with only the surrounding nodes or with all other nodes in the grid. The connecting lines prescribe the positions where truss elements may lie. Loads are applied on nodes of the ground structure. By varying the cross sections of the elements from zero to a certain maximum cross section, different topologies are generated, whereby members with zero cross section disappear in the truss layout. The topology with for example the least volume may then be calculated. An example of the steps in a standard topology optimisation process by Dorn et al. (1964) can be seen in figure 5.2.

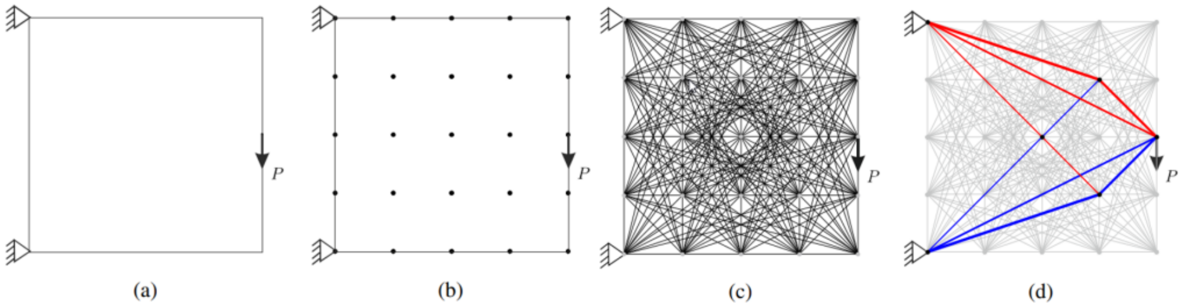


Figure 5.2: Topology optimisation process using the ground structure method (He et al., 2019)

The amount of nodes and connections in the ground structure determine the computation cost of the algorithm. An option is to decrease the amount of nodes and connections and create an additional step after the topology optimisation whereby the positions of nodes may shift. This step is called the layout optimisation step. Afterwards, one can reiterate to the topology optimisation step. This method allows more freedom to possible member positions, without making the ground structure too complex and heavy computation time-wise.

5.3.2. Ground structure methods with stock-assignment

Method by van Gelderen (2021)

In his master thesis van Gelderen (2021) develops an algorithm that is based on the adaptive member adding scheme by Gilbert and Tyas (2003) and later made available in a python script by He et al. (2019). The adaptive member adding scheme has the advantage that the same optimal solution can be found as with the general ground structure method, but with reduced computation time (He et al., 2019). This is because a reduced ground structure is used (see figure 5.3 (a)) together with a possible member list, the PML (5.3 (b)). The reduced ground structure has less connections between the nodes and during the iterations members from the PML will be added to the ground structure.

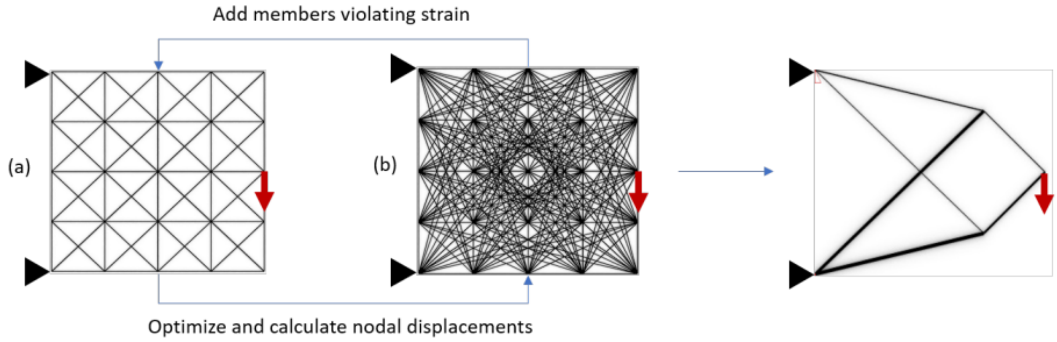

Add members violating strain
(a)
(b)
Optimize and calculate nodal displacements

Figure 5.3: Member adding scheme with a) reduced ground structure and b) possible member list (van Gelderen, 2021)

van Gelderen adapts the Python code from He et al., which consists of simple linear programming, to include a stock of reclaimed elements. The workflow of the method is as follows.

- The initial reduced ground structure is adapted such that the lengths of the connections between the nodes match the average length of the reclaimed elements. The PML only consists out of reclaimed elements and is thus named the possible stock member list (PSML). .
- A topology optimisation is performed on the reduced ground structure whereby the objective is to minimise the volume of the structure by varying the cross sections of the elements. Three constraints are defined:
  - there must be equilibrium,
  - the compressive resistance of the members must be sufficient and
  - the tensile resistance of the members must be sufficient.

  This optimisation problem is solved with the Dual Interior Point Method, which can be performed by using the cvxpy package in Python. This method is a class of algorithms to solve linear and nonlinear convex problems with one global optimum. Several solutions that are in equilibrium and fulfill all constraints are compared and the solution with the lowest volume is then chosen.
- Next, members from the ground structure are replaced by elements from stock whereby the element is chosen with the closest cross section, as calculated in the topology optimisation step. A minimum and maximum unity check limit is given, so that when no efficient stock element can be found, the element is regarded as a newly produced element.
- The nodal displacements of the truss with newly assigned cross sections is calculated. From the calculated displacements, the virtual strains of the PSML members are calculated. Members which violate the virtual strain limit are included in the ground structure. In addition a penalty system is set in place whereby penalties are given to inefficient and new members to avoid them from being present in next iterations.

Iterations are repeated until the virtual strain limit is not violated anymore and a minimum percentage of reuse and minimum average UC is reached. 100% reuse will not be easy to find as too many iterations will be needed. van Gelderen summarised his method in the following flowchart (figure 5.4)

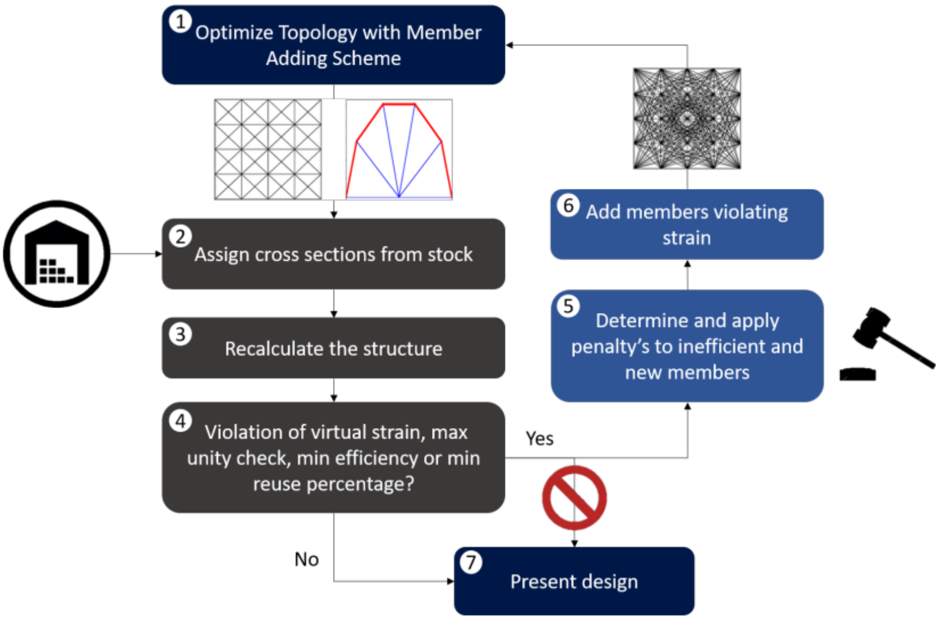
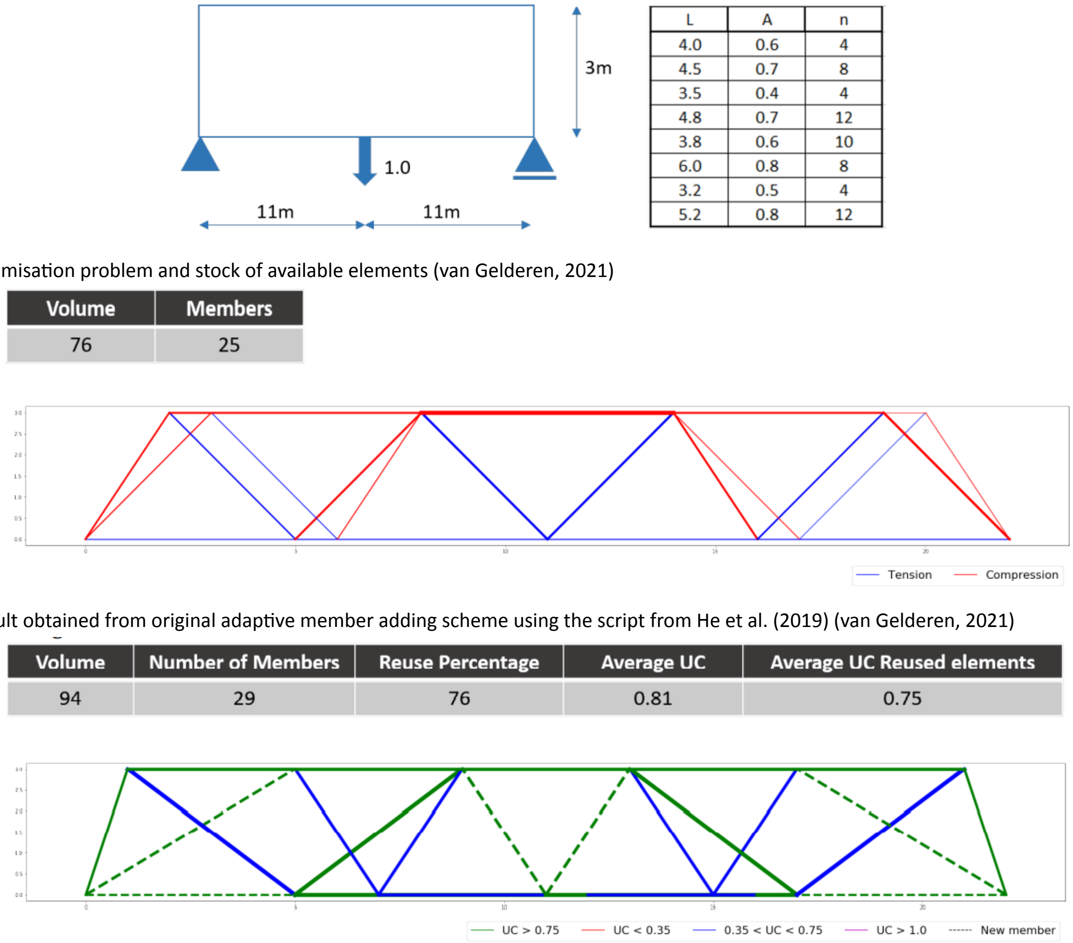

1 Optimize Topology with Member Adding Scheme
2 Assign cross sections from stock
3 Recalculate the structure
4 Violation of virtual strain, max unity check, min efficiency or min reuse percentage?
Yes
No
5 Determine and apply penalty's to inefficient and new members
6 Add members violating strain
7 Present design

Figure 5.4: Flowchart member adding scheme with reused elements (van Gelderen, 2021)

An example of a converged result of the method by van Gelderen is shown in figure 5.5. In the figure, the problem definition and the stock with available reclaimed elements are given. The first truss is the result of the adaptive member adding scheme using the script of He et al. The second truss is the result of the proposed algorithm by van Gelderen which includes reclaimed elements.



L	A	n
4.0	0.6	4
4.5	0.7	8
3.5	0.4	4
4.8	0.7	12
3.8	0.6	10
6.0	0.8	8
3.2	0.5	4
5.2	0.8	12

(a) Optimisation problem and stock of available elements (van Gelderen, 2021)

Volume	Members
76	25

(b) Result obtained from original adaptive member adding scheme using the script from He et al. (2019) (van Gelderen, 2021)

Volume	Number of Members	Reuse Percentage	Average UC	Average UC Reused elements
94	29	76	0.81	0.75

(c) Result obtained from proposed method by van Gelderen (2021)

Figure 5.5: Comparison of results in the master thesis by van Gelderen (2021)

There are a couple of limitations that hold back the usability of his method. First of all, a couple of essential adaptations to the script have to be made to avoid intersection and overlapping of members. Furthermore, some important checks like member buckling analysis and node instability still need to be included as well as the inclusion of self weight. Another limitation of this method is that the nodes of the ground structure grid are fixed, so that the incorporation of the exact lengths of the reclaimed elements is not possible and a lot of cut-off waste is produced. It does not include cut-off waste back into the element stock when they could still be usable.

A limitation of using the adaptive member adding scheme is the fact that reclaimed elements need to be fitted into the PSML. This neglects the possibility of reclaimed elements fitting into different positions in this grid, and therefore the eventual truss. The power of the adaptive member adding scheme lies in the variation of the cross sections to find the most optimal solution. By prescribing cross sections and their locations and using penalties to avoid problems, the algorithm becomes quite complex and finding an efficient structure can become challenging.

An advantage is that the script from van Gelderen is available so work does not have to be repeated and adaptations can directly be made.

Method by the Structural Xploration Lab of EFPL

The Structural Xploration Lab of the Swiss Federal Institute of Technology (EPFL) has published multiple papers on the subject of smartly including reclaimed elements in trusses and other kinds of structures. Their methods are based on the original ground structure method. In the paper of Brütting, Senatore, et al. (2018) the following workflow is followed.

- A ground structure is defined whereby its size and number of members is chosen so that a feasible assignment solution exist.

- An assignment and topology optimisation is performed using the simultaneous analysis and design (SAND) approach. This approach allows the possibility to regard the discrete assignment of the reclaimed elements and the topology optimisation as a mixed integer linear program (MILP) which can be solved to a global optimum. The YALMIP Matlab script is used for this optimisation step.

  A binary assignment matrix is used, whereby 1's and 0's define the locations of the stock elements. In the matrix it is possible to include an additional column where zero-elements may be assigned. These elements have a zero cross section and by assigning them to the ground structure, the topology of the truss may be changed as these members become invisible. To avoid unstable mechanisms from arising, it is possible to enforce the presence of non-zero-elements at certain locations. In addition, only the presence of one diagonal in each bay is allowed, so that overlapping elements cannot exist. The assignment matrix is included in the mathematical formulation of the goal and constraints of the optimisation problem. Therefore, the components of the matrix can be used as variables in the optimisation loop. Different topologies can be generated with different assignments of reclaimed elements, whereby the the structure will be chosen with the smallest mass. Figure 5.6 shows an example of the application of the assignment algorithm by Brütting, Senatore, et al.

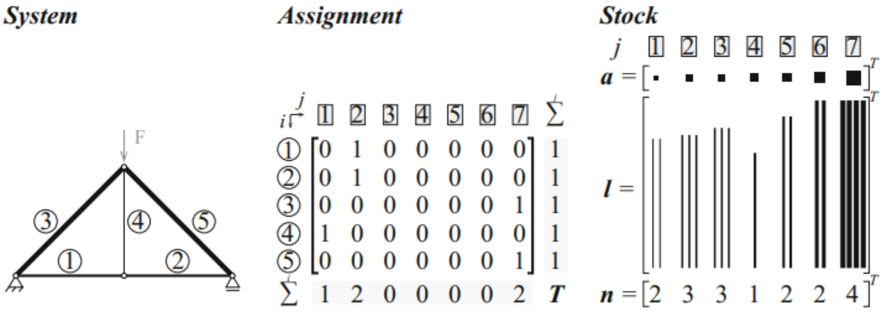


Figure 5.6: Assignment of available reclaimed elements to a truss design (Brütting, Senatore, et al., 2018)

- Furthermore, a geometry optimisation step is performed. In this step the aim is to minimise cut-off waste. The nodes of the truss from the previous steps may be shifted in a certain domain (to prevent overlapping) so that the reclaimed elements keep their original lengths, as can be seen in figure 5.7.

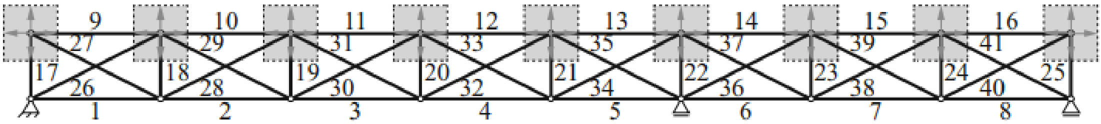


Figure 5.7: Shifting of nodes to incorporate geometry optimisation (Brütting, Senatore, et al., 2018)

- To not only increase lengths of elements a feedback loop is created to the topology and assignment optimisation step so that the structure can be recalculated to optimise on mass again. This is done by updating the coordinates of the nodes. The location of the nodes is included in the mathematical formulation of the topology and assignment optimisation problem so that all other variables depend on these values.

The algorithm is iterated until convergence is obtained whereby mass and cut-off waste are minimised. Important to note is that this sequential topology and geometry optimisation will result in a local optimum. An example of an outcome of the whole optimisation process is shown in figure 5.8.

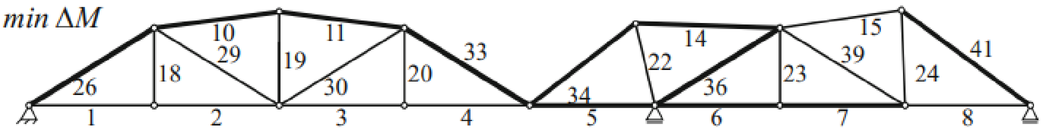


Figure 5.8: Result after convergence (Brütting, Senatore, et al., 2018)

In (Brütting et al., 2019), the above method is improved by including a deterministic combinatorial optimisation so that a global optimal element assignment can be found. The term combinatorial refers to the combination of topology optimisation and geometry optimisation. These are performed simultaneously instead of sequentially like in (Brütting, Senatore, et al., 2018). In their improved method they also calculate and subsequently minimise the embodied energy of the structure, which depends on the volume and cut-off waste which were in (Brütting, Senatore, et al., 2018) separately optimised. Furthermore, the geometry optimisation step now includes the objective of minimising compliance, which will result and the maximisation of the stiffness of the structure as compliance is defined as the inverse of stiffness. A stiffer structure will result in a smaller vertical displacement. Displacement limits of the structure to satisfy SLS constraints are checked in this method.

Discussion

Next to the fact that in Brütting, Senatore, et al.'s work a geometry optimisation step is included, the main difference between the method by van Gelderen and Brütting, Senatore, et al., is the way the topology optimisation step is performed. In van Gelderen's method, topology is optimised by varying the cross-sections of the members along a continuous range, whereafter a suitable reclaimed element is searched for. In Brütting, Senatore, et al.'s work, topology is optimised by varying the presence or absence of different reclaimed elements by varying 1's and 0's in the assignment matrix. The reclaimed elements are directly assigned in the ground structure and this seems like a more efficient and logical way to apply stock-constrained optimisation.

In overall, the following remarks may be made. Defining a prescribed web of positions where a stock of elements may be fitted in, may not be the most logical way to efficiently place them. It may be challenging to find the right positions for the elements while aiming to maximise material efficiency and minimise cut-off waste. The power behind the ground structure method is the fact that the topology of the truss is generated through variation of the members cross section, which is not possible with a discrete set of cross sections. However, Brütting, Senatore, et al. do find a smart way to handle this problem, by including the binary assignment matrix in the optimisation loop. They also find a way to implement the full length of a reclaimed element by varying the node positions and reiterating the topology optimisation step.

A downside to Brütting, Senatore, et al.'s method is that their script is not available. Therefore their algorithm would have to be rewritten in order to extend it to include more structural mechanical checks and incorporate the constraints related to the feasibility of connections.

Another downside could be that the ground structure method may not find a stock-constrained design solution when a limited number of reclaimed elements is available or when the dimensions of the elements vary a lot. There is a limited flexibility in rearranging the truss layout to the availability of elements. Brütting et al. therefore also allow cut-off waste and the implementation of new elements when no suitable reclaimed element can be found.

5.3.3. Ground structure methods in grasshopper tools

Important to mention in this paragraph are the existing Grasshopper tools Phoenix3D and Peregrine. These are two topology optimisation tools that are based on the ground structure method.

Phoenix3D

Phoenix3D is a tool developed by the Structural Xploration Lab of EFPL and explained in the paper by Warmuth et al. (2021). It includes the assignment optimisation step explained earlier in this paragraph. Next to the MILP it also presents a component to solve the optimisation problem using a Best-Fit heuristic, which is a more simplistic but much faster approach. The current tool needs an initial truss layout for which an optimal assignment of elements is found for. It does not perform a topology and geometry optimisation. It is also not possible to extend this tool in grasshopper to include more functionalities and fulfill our objectives. A short visualization of the possibilities of the tool is shown in 5.9.

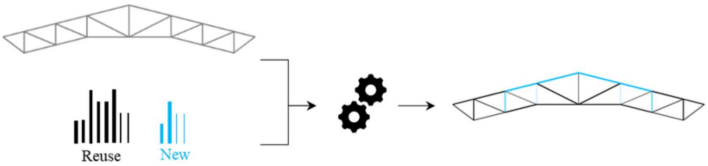


Figure 5.9: Example of a result using the Phoenix3D tool (Warmuth et al., 2021)

Peregrine

Peregrine uses the adaptive member adding scheme to find the optimal topology for a truss structure that is bounded by a certain design domain, with defined supports and loads, see figure 5.10. It does not provide the possibility to include a stock of elements or add own functionalities to the tool.

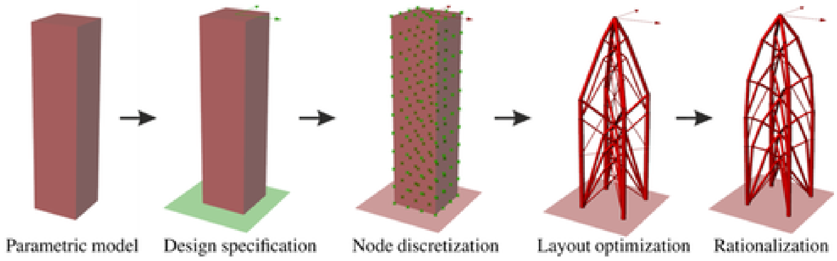


Figure 5.10: Topology optimisation using the plugin Peregrine (LimitState, 2019)

To conclude, these two tools offer already great functionalities that may be included in a parametric workflow, they do however not align with the objectives of this thesis and they are also not adaptable. These tools will therefore not be further considered in this thesis.

5.3.4. Other research projects using ground structure method

Lastly, some other papers researching truss topology optimisation using the ground structure method may be mentioned.

Sotiropoulos and Lagaros (2020) implement the ground structure using grasshopper components. This way, the ground structure is directly visualized and easily adjusted. Sliders can be implemented to let the user define the design domain and the complexity of the initial connectivity.

Achtziger (2007) presents a method to simultaneously perform a topology and geometry optimisation. He compares this method to optimisation methods where the steps are performed sequentially, like in the method of Brütting, Senatore, et al. (2018). When optimising sequentially, there is no information available on the quality of the obtained solution in terms of optimality. Therefore it is not known how far the local optimum lies from the global optimum. Additionally, he shows that the ground structure can stay rather simple when shifting and even merging of nodes is allowed . In that case, a lot of different geometries and topologies can be generated and a more optimal design can be found (Klemmt, 2021). Furthermore, he highlights the fact that using a sparse ground structure will result in more feasible design solutions as fewer connections will have to be made.

Fairclough and Gilbert (2020) present ways to make optimal truss layouts more buildable by limiting the number of joints, defining a minimum angle between elements and introducing dynamically generated crossover constraints that avoid cross over of members during layout optimisation.

Bukauskas (2020) develops a stock-constrained design method which is quite similar to the method by Brütting, Desruelle, et al. The difference lies in the choice of optimisation algorithm. He introduces an efficient assignment heuristic that is used instead of the SAND-approach. This new heuristic quickly generates approximately optimal solutions so that in an early design stage a real-time design exploration with feasible, stock-constrained and low-impact solutions is possible.

5.3.5. Explicit topology optimisation methods

In the explicit topology optimisation, members with a variable length, width, thickness, position and orientation are bounded by a design space. The design space is a FE mesh and by varying the member's characteristics, the members are floated across this mesh whereby the mesh is recalculated every iteration until a stable truss is formed with for example minimal volume. This is done by defining characteristic functions which describe the presence of elements (Coniglio et al., 2020). In the Moving Morphable Components (MMC) method by Guo et al. (2014) this is done using a Topology Description Function. The function is positive inside the area of the elements, 0 on the boundaries and negative outside the elements (Coniglio et al., 2020). Such a contour plot is shown in figure 5.11.

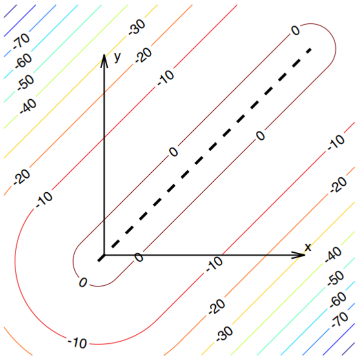
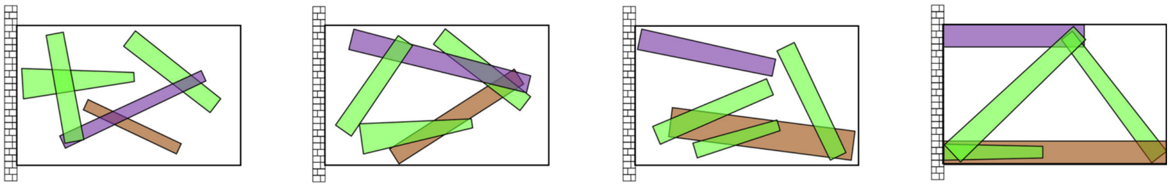


Figure 5.11: Contour plot of a geometric component (Coniglio et al., 2020)

In each iteration the distribution of material is translated to a uniform distribution of the Young's modulus and the density inside each element of the FE mesh (Coniglio et al., 2020). The members may overlap so that for example, elements may disappear in the final design if overlapped fully by another member. This is the case when you let elements merge, another possibility is to sum up the thicknesses of the overlapped parts(Coniglio et al., 2020).

In the optimisation problem formulation, the objective function can be written as a volume integral (Guo et al., 2014). The optimisation step is performed by using the Method of Moving Asymptotes (MMA), which was introduced by Svanberg in 1987 and improved in later publications (Svanberg, 2002) (Svanberg, 2007). MMA is a method to solve nonlinear programming in optimisation problems, whereby in each iteration a convex subproblem is generated which is again iteratively solved. The subproblems are generated using gradient information of the current iteration point, while controlled by ”moving asymptotes”, which are parameters that are updated every iteration (Svanberg, 1987).

A visualization of the MMC method is shown in figure 5.12, other explicit topology optimisation methods follow the same idea.



Components: the basic building blocks for MMC based topology optimization

Topology 1

Topology 2

Topology 3

Figure 5.12: Basic idea of the MMC method (Zhang et al., 2016)

The script of the MMC method by Zhang et al. (2016) is available. In this script it is possible to set the length, width and thickness of the elements so that these are not variables anymore and cannot be used in the optimisation loop. This way, a stock of elements can be included in a design space and float through this space until an optimal layout is found.

However, the algorithm is quite complex and when including too many elements or positioning them in a non-efficient way, the computation time increases drastically and no solution can be found. The way that the algorithm is set up, namely the use of a FE mesh which is recalculated every iteration and the use of the Method of Moving Asymptotes to solve the optimisation problem, is not a method that can be effectively simplified to make the computation time shorter.

5.3.6. Growth methods

In the growth method, new nodes and elements are generated and added to an initial layout. The layout can grow in dimensions or grow in complexity. During the iterations, elements and nodes may also be eliminated again. The manner that the geometries are grown differs per method.

Growth method by Klemmt (2021)

Klemmt (2021) has written a new structural growth algorithm in python and implements it in grasshopper, whereby the plugin Karamba is used to perform FEM-calculations on the elements in the truss. The python script is not available. The aim of his research was to avoid optimisation based on the description of mathematical rules but instead attempt to generate suitable trusses using a bottom-up growth simulation, whereby the result may not necessarily be the most optimal design. He explores cell-based growth and how this can be used to generate efficient truss structures. Next to being interested in exploring a cell-based growth process, he also chooses this direction because he concludes that the reason that research up to now has focused on 2D trusses is because of the difficulty of defining these mathematical rules for a 3D space. A growth method might overcome this obstacle.

In his algorithm an initial truss is given as input and its nodes will iteratively reposition and once every circa 10 iterations subdivide creating new connections. The nodes are moved and added so that forces are distributed efficiently, the structure hereby becomes more complex and refined. Note that this is a different optimisation goal than in most other methods discussed in this thesis, where the goal is to minimise mass or volume. In the optimisation algorithm, 3 forces are calculated for each node:

- The force required to align the beams that are both in compression or tension
- The force needed to pull beams where one is in compression and the other in tension into a certain angle
- The force required to equalize beam lengths

These three forces are clarified in figure 5.13.

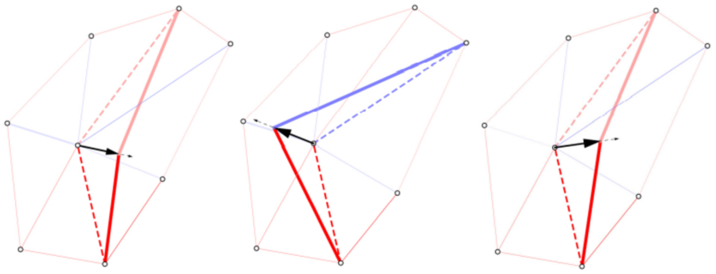


Figure 5.13: From left to right: force aligning elements, force pulling elements into an angle, force equalizing beam lengths (Klemmt, 2021)

These forces are then scaled and added to a movement factor. The node is moved and the elements are checked so that no intersection occurs. Once every circa 10 steps of node repositioning, the node connecting elements with the highest unity checks is chosen to be subdivided. The new node is positioned with a certain offset from the original node and is connected to all surrounding nodes. When an element is underutilised, during several iterations, it will be removed. Every iteration, the forces in the elements and the nodal displacements are calculated with Karamba, also the cross section optimiser component of Karamba is used to calculate optimal cross sections of the included members. The result of this algorithm is a growing triangular mesh. An example of the generation of a cantilever truss is shown in figure 5.14.

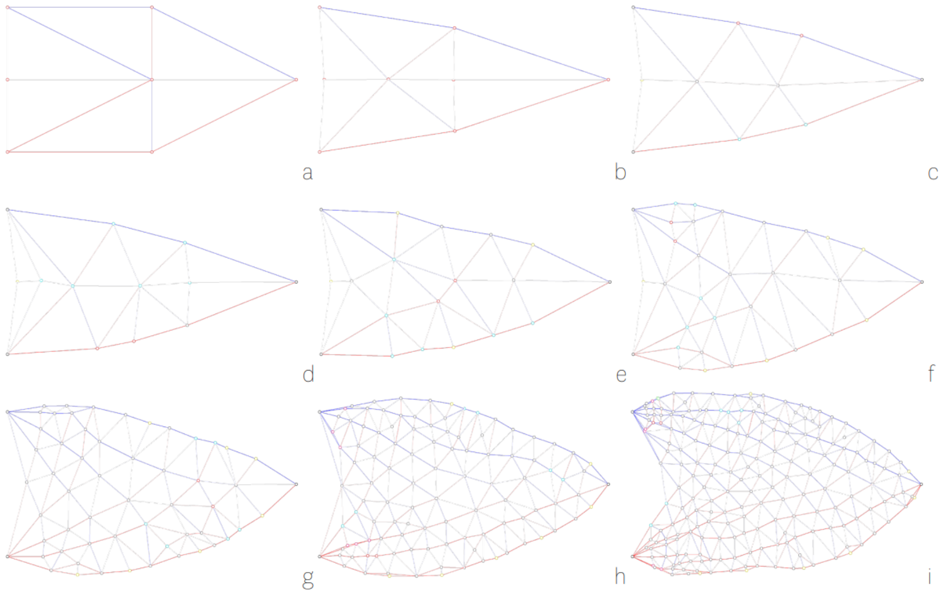


Figure 5.14: Development of the model during its growth (Klemmt, 2021)

Growth method by Cui and Jiang (2014)

Cui and Jiang (2014) propose a morphogenesis method to optimise truss layouts whereby elements are added or deleted and nodes are moved so that the strain energy of the elements is minimised. The minimisation of strain energy results in a maximisation of the stiffness of the structure. The strain energy is formulated as follows:

$$C = \frac{1}{2} F^T U \tag{5.1}$$

Whereby F is the nodal load vector and U is the nodal displacement vector in global coordinates (Cui & Jiang, 2014).

In the algorithm, elements are divided into three groups according to their strain energy sensitivity. Two limits are defined so that group A consists of elements with low strain energy sensitivity and group C has elements with a high strain energy sensitivity. Elements from group A may be deleted out of the structure as these elements will have little influence on the strain energy of the structure. In contrast, elements may be added to the truss at the location of the elements from group C as a large strain energy sensitivity indicates a large internal force, see figure 5.15.

Lastly, nodes may be repositioned as their position might not be the most rational one after the growing process. The nodes are shifted in opposite direction to the strain energy sensitivity.

Initially, all members have the same cross section. An optimisation of the cross sections may be performed at the end of the growth process. An example of the method by Cui and Jiang is shown in figure 5.16.

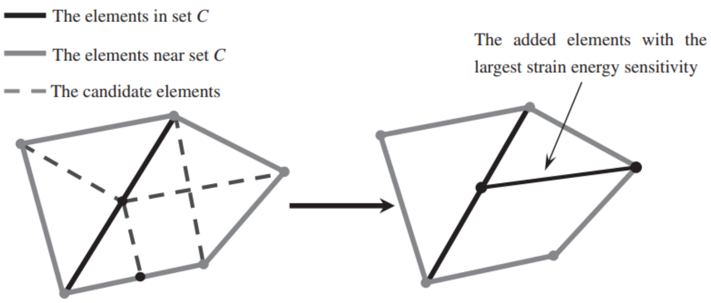


Figure 5.15: Addition near element of group C (Cui & Jiang, 2014)

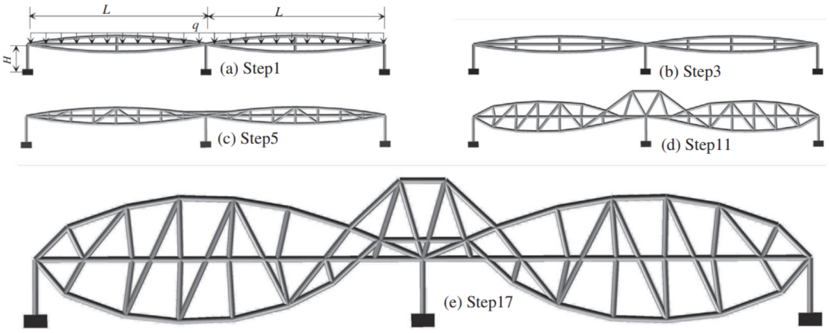


Figure 5.16: The optimisation process using the growth method by Cui and Jiang (2014)

Growth method by Martinez et al. (2007)

Another type of growth algorithm was proposed by Martinez et al. (2007). In his algorithm, members are generated between predefined supports and loads, whereby elements are removed that make the structure statically indeterminate (the topology optimisation step). This is done by minimising the mass of all possible members for a single load case. Subsequently, the node coordinates become the variables in the optimisation loop, whereby mass is again minimised, this can be seen as the geometry optimisation step. When a certain node moves a lot, it is possible that a different topology may be more optimal. In this case, the last added member is removed and the node is reconnected to the currently surrounding nodes. For the new structure, the topology optimisation step will again be performed and so on.

Furthermore, a last step is included whereby a new node is created where two members intersect or a new node is created bisecting one element. The new node is connected to its surrounding nodes and a local topology optimisation is performed. This step is performed for every possible node position, whereby the topology with the lowest mass is chosen. The growth process using the algorithm of Martinez et al. is shown in figure 5.17.

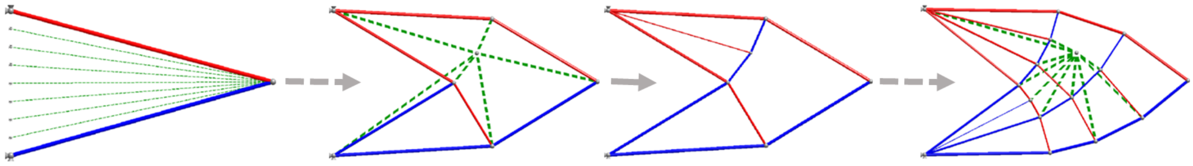


Figure 5.17: The optimisation process using the growth method by Martinez et al. (2007)

Discussion

The workflow of a growth method is quite intuitive and suits the goal of this thesis. When wanting to design a structure using available elements, a truss can be grown by adding members where forces need to be sustained and eliminating under-utilised members. This is a more intuitive and logical way than matching elements out of a stock to an initial design. However, as existing growth methods vary elements lengths and dimensions and bisect them, a different approach is needed to include reclaimed elements with fixed dimensions and lengths. An algorithm needs to be developed from the start. However, inspiration can be drawn from the existing growth methods.

From the method from Klemmt the same kind of node division and element addition can be used, whereby a triangular truss structure is generated. From the method of Cui and Jiang the method of dividing elements into three groups could be copied, but according to for example their unity checks. Elements which are very underutilised can be deleted while elements which are overutilised could be helped by adding new elements at these locations. The method by Martinez et al. whereby members are bisected is not an option when using available reclaimed elements as their whole lengths are preferred to be used. What can be copied is the selection of the best option based on minimal mass.

The downside of using the growth method and writing an own script is that finding a global optimum might be challenging.

5.3.7. Voxel based topology optimisation methods

A very popular method nowadays to generate optimal structural designs is by using a voxel based topology optimisation method (Klemmt, 2021). In these algorithms a FEM mesh is created and material is added or eliminated so that only the most utilised voxels are left and form an efficient structure (Klemmt, 2021). There are methods where voxels can be either solid or void or methods where voxels have a continuous density between 0 and 1 (Klemmt, 2021). The layout of the structure is shown by mapping the absence or presence of material (Coniglio et al., 2020). In figure 5.18 iterations are shown of a voxel-based optimisation algorithm by Querin et al. (1998) where voxels are either a void or solid. In figure 5.19 another algorithm is shown, developed by Bendsøe and Kikuchi (1988) where continuous voxels are used.

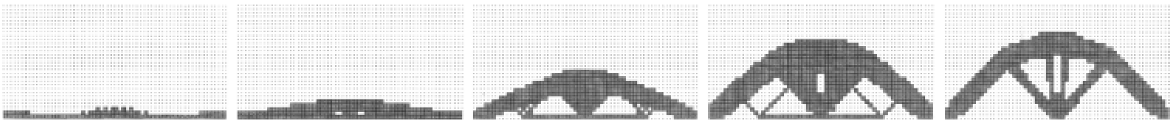


Figure 5.18: Voxel-based topology algorithm with solid and void voxels (Querin et al., 1998)

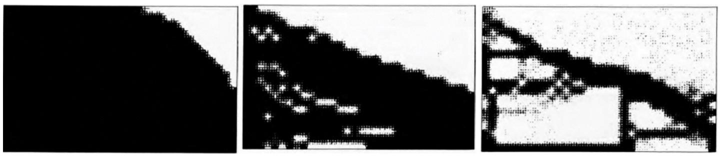


Figure 5.19: Voxel-based topology optimisation with continuous voxels (Bendsøe & Kikuchi, 1988)

The obtained free-form topology can become complex (Klemmt, 2021). These forms were for a long time not feasible to construct but with the arrival of 3D printing, this did become possible and therefore this method has become one of the most popular topology optimisation algorithms.

However, this method is not suitable when discrete elements with certain dimensions need to be implemented and the topology optimisation cannot be performed in a continuous or voxelised field. Therefore, these kinds of topology optimisations are left out of consideration in this thesis.

5.3.8. Other methods to generate truss layouts

Next to the widely used methods above, also other (creative) possibilities exist to generate different truss topologies to optimise. An example is the method by Vierlinger and Bollinger (2014) where they generate random directional fields that are parameterized by three points of simulated magnetic charge. The diagonals of the truss subsequently follow the orthogonal lines of these fields at the level of the mid-axis of the truss, see figure 5.20. A truss can be generated by addition of two magnetic fields, see figure 5.21. The magnetic fields are varied by the evolutionary optimisation tool Octopus in Grasshopper, so that the geometry can be found with the lowest mass.

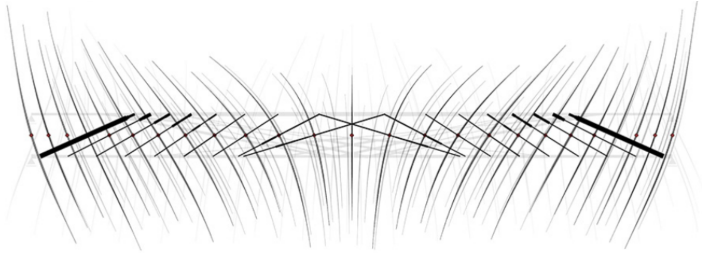


Figure 5.20: Diagonals defined by a point and a direction using magnetic fields (Vierlinger & Bollinger, 2014)

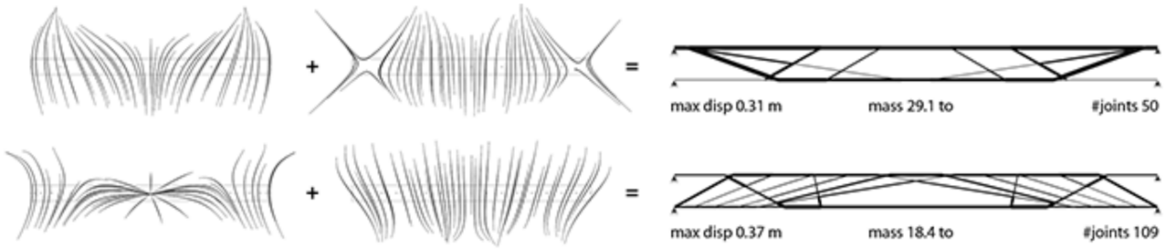
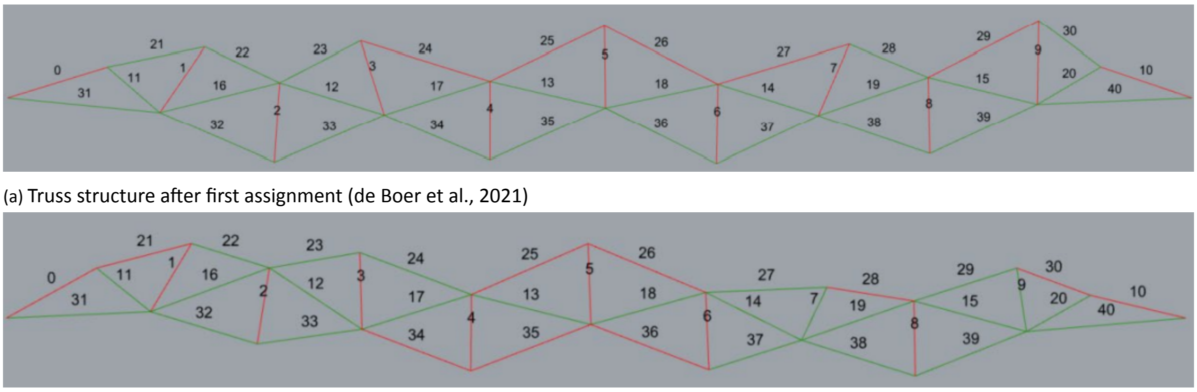


Figure 5.21: Generation of truss variants (Vierlinger & Bollinger, 2014)

Another example is the simple method used by de Boer et al. (2021) in his master thesis. He introduces an initial truss geometry and analyses it with Karamba in Grasshopper. He uses the cross section optimiser tool of Karamba to define the required cross sections of the truss members to resist the applied load. He subsequently matches the smallest sufficient reclaimed profile to each member in a linear manner. In the assignment step, a limit is given to the difference in length between the original member and the reclaimed element. When no suitable element is found, the element will be considered as a newly produced element. When a reclaimed element is selected, its whole length will be included in the truss. This is achieved by shifting the nodes to let the element fit in. The obtained truss structure after this first assignment step can be seen in 5.22a.

Subsequently, he analyses the truss a second time and checks whether any members are now over-utilised, due to the changed layout of the truss. Only the under-dimensioned members will again be swapped with new reclaimed elements, this time cutting them to the required length so that a third redistribution of forces is prevented. The final structure can be seen in 5.22b.



(a) Truss structure after first assignment (de Boer et al., 2021)

(b) Truss structure after second assignment, final structure (de Boer et al., 2021)

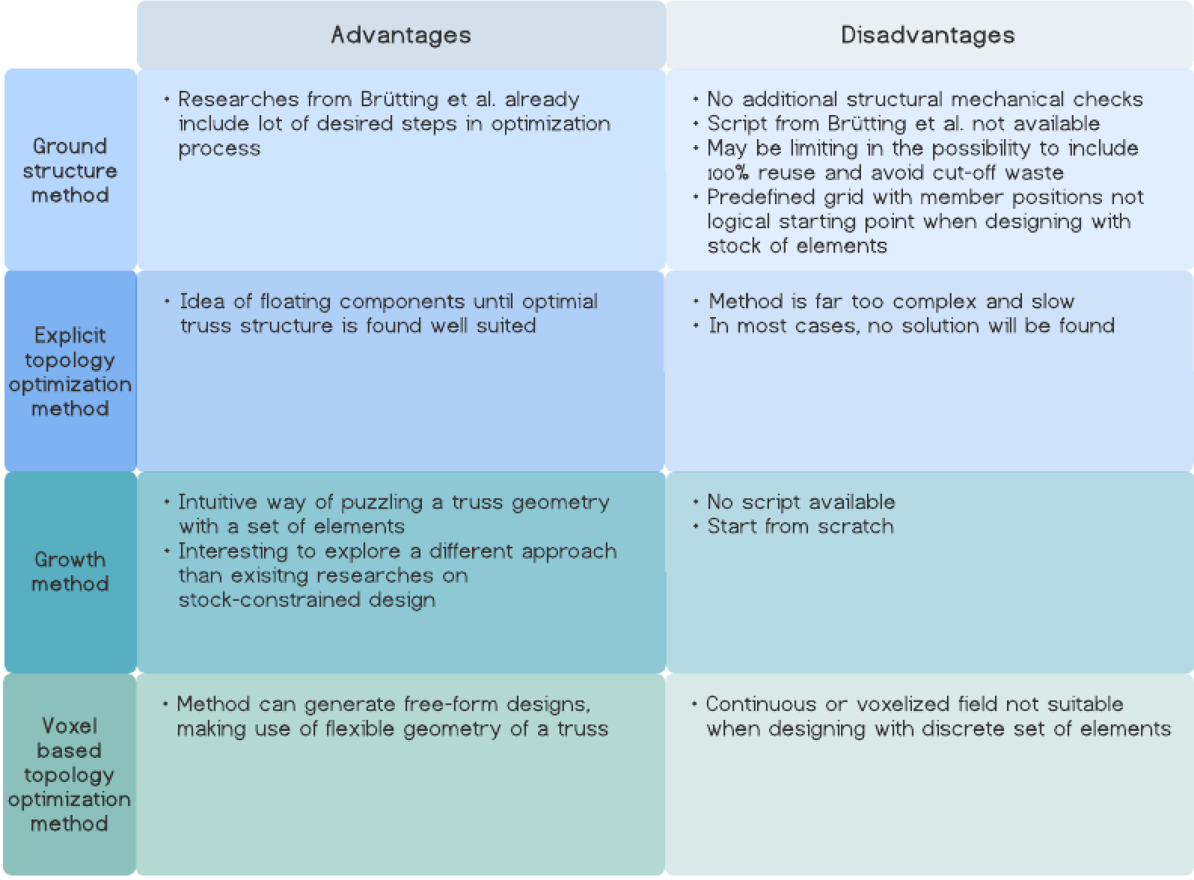
Figure 5.22: Optimisation process by de Boer et al. (2021) whereby green elements represent reclaimed elements out of a stock and red elements need to be newly produced

The method by de Boer et al. (2021) is quite simplistic whereby optimisation is performed in a linear way, neglecting the many possible alternatives of different stock assignments. It also neglects the fact that due to the changed layout, elements can become under-utilised. These elements are not swapped in the second iteration. Also new elements that were defined in the first step, are not swapped anymore with reclaimed elements in later iterations, even if their internal forces have changed due to a different lay-out. Furthermore, the method is restricted to an assignment step (with shifting of nodes), but leaves out a topology optimisation step.

The proposed method is however a good example of how a simple optimisation can be achieved using the opportunities that Karamba provides and using own simple steps, without getting caught in the complex truss topology algorithms described above. It delivers a fast way to find a truss design including reclaimed elements, even if it is not the most optimal solution.

5.4. Overview and discussion

The first step into designing an optimal truss structure is choosing a method to generate different alternatives so that these can be compared. Four methods have been described above. Their advantages and disadvantages in their applicability to the aim of this thesis are summarised in the overview below.



	Advantages	Disadvantages
Ground structure method	• Researches from Brütting et al. already include lot of desired steps in optimization process	• No additional structural mechanical checks • Script from Brütting et al. not available • May be limiting in the possibility to include 100% reuse and avoid cut-off waste • Predefined grid with member positions not logical starting point when designing with stock of elements
Explicit topology optimization method	• Idea of floating components until optimial truss structure is found well suited	• Method is far too complex and slow • In most cases, no solution will be found
Growth method	• Intuitive way of puzzling a truss geometry with a set of elements • Interesting to explore a different approach than exisitng researches on stock-constrained design	• No script available • Start from scratch
Voxel based topology optimization method	• Method can generate free-form designs, making use of flexible geometry of a truss	• Continuous or voxelized field not suitable when designing with discrete set of elements

Figure 5.23: Overview of existing methods to generate truss geometry variants and their applicability to the goal of this thesis

The choice has been made to use the growth method as inspiration for the new design approach that will be developed further in this thesis.

The research projects on stock-constrained truss design have until now used the ground structure method in their algorithms. They adapted and extended this method to include quite some objectives when designing with a stock of elements. However, as it still includes fitting elements into a prescribed regular grid of positions, it needs quite some steps and iterations to efficiently place the available elements. A consequence is that cut-off waste is produced as well as the inclusion of new steel elements when no suitable match can be found.

The growth method therefore provides a very interesting alternative way to tackle stock-constrained design as it has a more intuitive starting point. Elements with their full lengths can be added to (and eliminated from) the design until the full span has been reached, hereby creating a truss design with 100% reuse. This idea aligns with the goal of this thesis to design a truss with what is available.

A new algorithm will have to be developed from scratch as none of the existing growth methods are adaptable to include a stock of elements. However, it will be interesting to see in this thesis how far you can come in creating stable, efficient truss designs using this different approach compared to existing research.

6

Stock-constrained design of a truss bridge

6.1. Different growing methods

First of all various ideas were considered on how the truss could be 'grown'. The options will be shortly mentioned in this section along with their advantages and disadvantages.

6.1.1. Grow in density

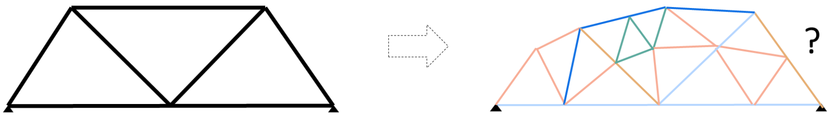


Figure 6.1: Example of growing in density

Inspired by existing growth methods, it was first explored whether the truss could be grown in density. Existing methods in general begin with a simple initial structure and let nodes and elements be added and shifted to create a smart design.

In the situation where a stock of elements needs to be included, an option would be to first create an initial structure and let reclaimed elements fit in by densifying and at the same time replacing the initial members so that a truss with only reclaimed elements is left over.

Existing methods split up elements or shift nodes and thereby change certain element lengths to find the best topology. However, like discussed earlier, this is not feasible when designing with elements with set lengths and when wanting to minimise trim loss. This way of growing then essentially becomes a fitting-in-method again. The aim is to look if a truss can be generated in a different way. Other growing methods will therefore be explored.

6.1.2. Grow from the middle

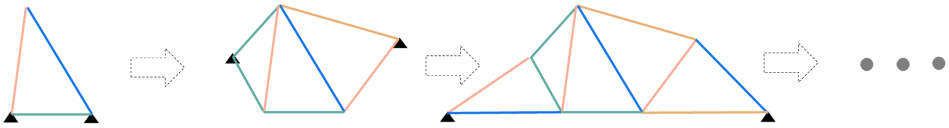


Figure 6.2: Example of growing from the middle

Other than growing in density, the truss can be grown in dimensions. An idea was to start from the middle and let triangles be added to the sides. That way, the process starts with a small truss, on two supports and will be lengthened in each iteration, while replacing the supports, until the full span has been reached. The advantage is that the truss can be grown in a manner that in each iteration a fully stable truss can be analysed using Karamba so that every iteration is checked on all structural requirements. Another advantage is that elements can be added with their full lengths whereby the extra node can simply be added where the two elements cross each other, hereby forming a new triangle. This way, elements do not have to be fit in, but the geometry will depend on the available lengths. By letting the truss grow symmetrically and letting the new triangles append to the most vertical side bars, the idea is that a realistic truss can be grown. In addition, a geometrical constraint could be added to make sure that the bottom elements are placed horizontally so that a straight bottom chord is created.

A downside to this method is the difficulty to model the truss and its loads in such a way that when growing the truss in length, the initially calculated middle bars still suffice. A problem could be that the vertical deflection would become too large as this cannot be checked correctly during the growing process. The forces that the middle of the truss will need to carry will also increase exponentially when the length increases. A couple of workarounds could be to:

- Replace middle elements during the growth process according to the changing required capacity.

  A problem that arises using this workaround is that when replacing elements, the lengths cannot be changed anymore as this would require all the nodes of the truss to shift. Either elements would need to be cut to fit in or the truss will need to regrow from the changed geometry onwards. Growing an irregular geometry according to picked lengths and afterwards replacing them with other elements which have to be cut creates an illogical truss design which does not have any advantage. It is however unavoidable that elements need to be updated and regrowing the truss every time will put the algorithm in an unsolvable loop.

- Add elements so that the middle of the truss is densified during the growth process.

  The danger here is that the final geometry could get a very illogical geometry and force distribution. Additionally, elements would again need to be fit into the design.

- Adapt the loads so that the final required capacity is directly calculated.

  The loads could be recalculated as normal forces that the elements would need to withstand in the final geometry. However, as the final geometry is not defined yet, calculating the final forces would include a lot of guesswork so the risk remains that the elements will not meet unity checks and will have to be replaced.

- Adapt the unity checks in the calculation so that larger profiles are picked in the beginning.

  Adapting the unity checks could become quite complex and in addition the same problem arises as with changing the loads. Calculating what the reduced unity check would need to be would include a lot of guesswork.

The conclusion after analysing these problems is that the truss should be grown depending on the final forces that the elements will need to resist. Growing the truss from the middle makes calculating and implementing these forces quite complicated.

6.1.3. Start with the bottom chord

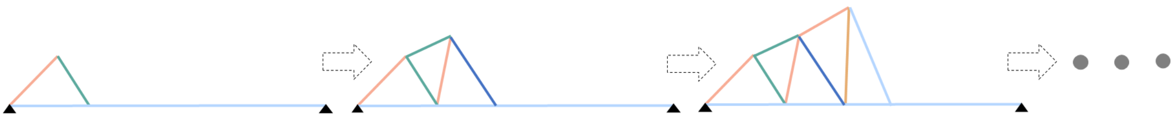


Figure 6.3: Example of growing from the start

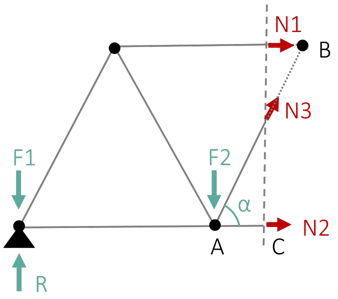
This third option starts off with calculating the required bottom chord capacity, using a standard truss design, whereafter elements are added on top of this bottom chord to generate a new truss design. The advantage of starting off with a bottom chord is that you directly implement the boundary condition that a bridge deck needs to be able to be connected. It also allows for the design of a bottom chord with the least welds as possible as the longest elements (with preferably the same profiles) may be chosen from the stock, resulting in a more logical bottom chord than when growing it in several steps during the growth process.

After defining the bottom chord, diagonals can be added beginning at the start of the span. The downside to this method is that the intermediate geometry cannot be globally checked using Karamba as only at the end of the growth process a truss will be generated that is stable and includes the final forces. This results in the need for a different calculation method. The problem that needs to be overcome is the fact that elements need to be structurally checked, without having knowledge on the rest of the geometry.

The answer to this question was found in the method of section, a method to analyse truss forces. A constraint of this method is that it does require the truss to be statically determinate, so that the reaction forces are known a priori. Subsequently, the forces in the truss elements are calculated by cutting the truss at a certain location along the span whereby only the forces and geometry on one side of the truss have to be known. By growing the truss from the start of the span, the section moves from beginning to end whereby only the already defined geometry is relevant to the decision of the next to be placed elements. In the calculations the stresses in the elements as well as the member instability is taken into account. After the truss is fully grown, it can be globally calculated using Karamba, so that aspects like vertical deflection and global lateral torsional buckling may be checked.

6.2. Analysis of trusses by method of section

The method of section section works by defining an imaginary section at the location of the elements whose axial forces are unknown and wished to be calculated. By performing moment equilibrium checks around smartly chosen points the unknown axial forces may be calculated. The points are chosen so that only one unknown force is present in the equation. The points are therefore usually located on the intersection points where other unknown forces pass through so that they do not need to be considered in the moment equilibrium. An example is shown below.



$$\begin{cases} \Sigma M_A = 0 = R * x_A - F1 * x_A + N1 * h \rightarrow N1 \\ \Sigma M_B = 0 = R * x_B - F1 * x_B - F2 * \ (x_B - x_A) - N2 * h \rightarrow N2 \\ \Sigma M_C = 0 = R * x_C - F1 * x_C - F2 * \ (x_C - x_A) + N1 * h + N3_V * (x_C - x_A) \rightarrow N3_V \\ N3 = \frac{N3_V}{\sin(\alpha)} \rightarrow N3 \end{cases}$$

Figure 6.4: Example of the method of section

In the example it can be seen that only point loads, positioned at nodes, are taken into account and that all nodes all modelled as hinges. This is a simplification of the reality where a distributed load needs to be supported by the bottom chord and the bottom chord will be a continuous beam. This causes an irregular moment distribution that needs to be supported by the bottom chord. However, this moment distribution has negligible effect on the axial forces in the other elements of the truss. The moment distribution is governing for the selection of the bottom chord profiles, but this is already taken into account when selecting the bottom chord at the beginning of this method.

The connections between the diagonals and the bottom chord will be welded so in reality they will take up some moment next to the axial forces acting onto them. For the weld design this is important to recognize, however for the element design, modelling the connections as fully hinged will simplify calculations while guaranteeing a more conservative solution.

Therefore, after calculating the required bottom chord, the hand calculations will be performed on the truss modelled with hinges and point loads.

6.3. Structural mechanical checks

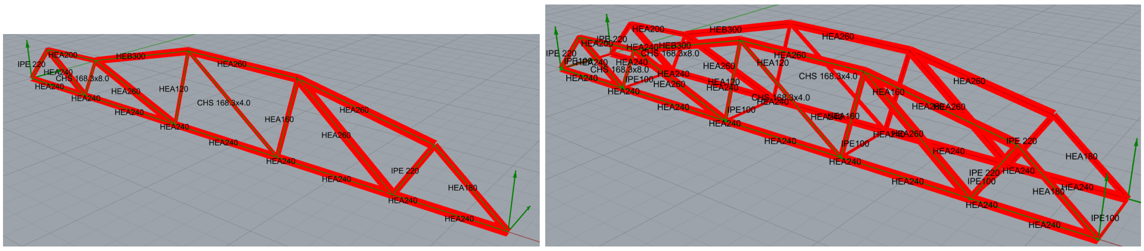
As mentioned above, during the growth process, elements will be tested on their capacity considering stresses that need to be resisted and member instability. Maximum allowed unity check is set on a reduced value of 0,9 and 0,8 for the top chord elements (more susceptible to instability) to acknowledge that additional global checks that need to be calculated afterwards may be governing.

When the full truss geometry has been generated it is required to check the truss on lateral torsional stability, which can be done in Grasshopper using Karamba. Karamba can calculate the second order normal forces in the final truss bridge design, whereafter the geometric and elastic stiffness matrices of the structure may be calculated. These are then used to find the buckling load factors for linear elastic buckling. By multiplying the smallest buckling load factor with the applied loads, the linear elastic buckling load is found. This means that as long as the smallest buckling factor is larger than 1, the structure will not become unstable.

As the output of the growth method is a 2D truss it should be considered how to check global instability in a realistic way. The half-open truss bridge consists out of two trusses, with a fixed connection to the bridge deck. The stiffness of the bridge deck will counteract lateral buckling of the two trusses. The stiffness and mass of the bridgedeck will follow from input values, which will be further explained in chapter 8.1. To model the stiffness of the bridge deck appropriately, a 3D model is generated from the 2D truss output from the growth method. The truss is copied and shifted horizontally according to the desired bridge deck width. Later in the design process, 2 unique trusses will be generated to together form a full truss bridge design. In that stage the global checks can be done more accurately. For this earlier stage, a truss still needs to be checked on global performance to be included into a solution cloud of valid truss options. In this case, the 3D model with a copied second truss will suffice.

The bridgedeck is modelled by adding cross beams with a certain profile. In the 3D model, the loads on the bridgedeck will be modelled as lineloads on these crossbeams. In the 2D model, pointloads at the locations of the crossbeams will represent the forces to be transferred to the bottom chord. The cross beams have a moment resisting connection to the two bottom chords. This way lateral torsional instability of the trusses will be counteracted by a rotational stiffness along the length of the truss. This will be the most realistic and parameterized way to check the global instability of the generated trusses.

Furthermore, next to global instability, the minimal required natural frequency of the truss bridge may be calculated and checked using Karamba so that dynamic issues are avoided. Lastly, also the vertical deflection of the truss bridge is checked.



(a) 2D truss generated in growth algorithm

(b) 3D model after copying truss

Figure 6.5: Generating 3D model

6.4. Scripting environment of the algorithm

The design process will be developed in Grasshopper. A reclaimed element database can be imported as an excel file into Grasshopper. In Grasshopper, a choice has to be made on how the algorithm is written. The options are:

- Using only grasshopper components

  Already quite quickly it became clear that grasshopper components alone would not suffice to perform all the steps and loops in the algorithm. Grasshopper components can become quite slow quite quickly. Certain tasks like loops are much quicker to run in python then by connecting multiple grasshopper components to each other.

- Using the GHPython component

  This is a readily installed component in Grasshopper wherein python scripts can be written. On the outside of the component the inputs and outputs can be defined and connected to other Grasshopper components.

  A downside to this component is that only RhinoScriptSyntax can be imported, with which you can generate and modify geometry like in Rhino or Grasshopper. No other python packages like numpy and pandas are able to be imported, which makes the possibilities of scripting quite limited. This became especially clear when deciding on how to store the information of the reclaimed steel element stock and update this information during the algorithm. The most structured way to do this is by using a pandas dataframe. This is a clear way to store data in a 2D dataframe whereby data manipulation can be easily carried out. It makes searching for information on elements, choosing elements and restoring spare parts much more accessible during the algorithm.

- Using the GH_CPython component

  This is a component developed by AbdelRahman (2017) to make the import of python packages possible. This is a big advantage, however, the downside to this component as that RhinoScriptSyntax is not available in this case. So in this component it would be possible to write the algorithm using pandas dataframes, but it is not possible to create geometry during the algorithm and the only solution would be to output x,y,z-coordinates of the final nodes so that the truss geometry can be generated afterwards using grasshopper components. This is however not an ideal and efficient workflow.

- Using GHPython Remote + GHPython component

  A solution to the problems faced in the previous options is solved in this last option. The GHPython Remote by Cuvilliers et al. (2017) connects the GHPython component to an external Python interpreter, making it possible to use all python packages, while still being able to make use of the RhinoScriptSyntax.

To conclude, the algorithm will be set up in Grasshopper, where a reclaimed element stock can be imported as an excel file. This file is subsequently turned into a pandas dataframe where additional information may be added like the elements capacities.

Furthermore, the algorithm will be scripted in python using the GHPython Remote and GHPython components. The algorithm will consist out of multiple steps and loops. As it is desired to visualize most of the steps, the python script will be split up into several parts, so that multiple GHPython components can be run after each other while the output in between them can be visualized using Grasshopper components. This way, the computation time will also be divided so that the whole algorithm can be run in parts, making it easier for Grasshopper. It also ensures that steps like turning the data from the excel file in a pandas dataframe can be run separately in the beginning or that specific steps can be run multiple times while other components keep their calculated output.

As an example, a part of the final grasshopper file is shown in figure 6.6. It can be seen that several GHPython components are connected to each other. These are the grey components which are labeled with 'Python'. The inputs and outputs of the components are connected with lines, letting information flow from one python script to another. Besides connections between the python components, certain outputs are also connected with non-visible grasshopper components which are used to visualize the intermediate results.

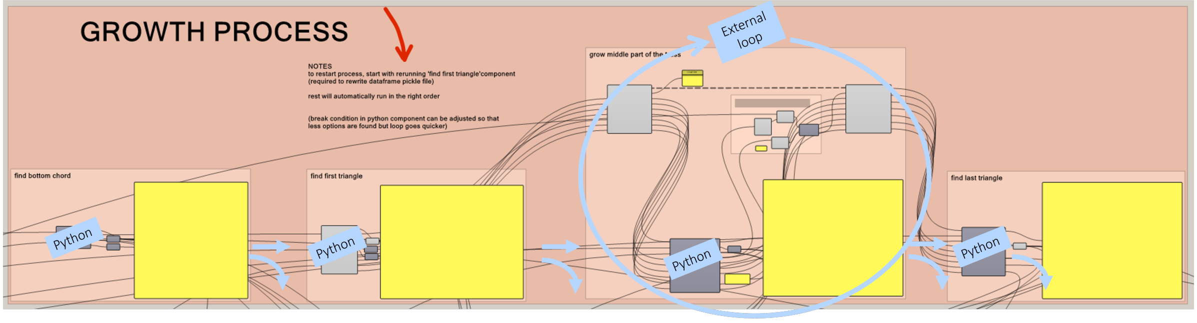
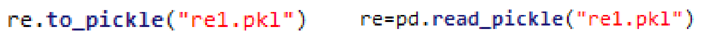

GROWTH PROCESS
NOTES
to restart process, start with rerunning 'find first triangle'component
(required to rewrite dataframe pickle file)
rest will automatically run in the right order
(break condition in python component can be adjusted so that
less options are found but loop goes quicker)
find bottom chord
find first triangle
grow middle part of the truss
find last triangle
External loop
Python
Python
Python
Python

Figure 6.6: Connection of Python components in Grasshopper

Two additional problems needed to be overcome to apply the desired workflow in Grasshopper. The first one is the way that the dataframes can be sent from one component to the other, while being updated. The GHPython Remote makes it possible to script with the pandas package inside the GHPython component, but it lacks the possibilities to input or output anything that has to do with additionally imported python packages. The solution to this problem was to pickle the dataframe. This way the dataframe is temporarily stored inside a pickle file and can be easily opened in another GHPython component without loosing information and without needing a long computation time.



```
re.to_pickle("re1.pkl")
```

```
re=pd.read_pickle("re1.pkl")
```

(a) Storing the reclaimed element database in one python component

(b) Opening the reclaimed element database in another python component

Figure 6.7: Transferring pandas dataframes in Grasshopper

The second problem is the way loops can be carried out inside Grasshopper. Inside the various GHPython components for and while loops can be scripted to perform some of the loops required in the algorithm. However, because the algorithm is now split into several GHPython components it is necessary to find a way to perform looped calculations outside of these components. The Anemone plugin by Zwierzycki (2015) makes this possible by providing grasshopper components that can be connected to all other types of components, creating a loop. For and while loops but also nested loops can be achieved through this system. In figure 6.6 such an external Anemone loop can be seen.

6.5. General remarks about the algorithm

In this paragraph the overall setup of the algorithm is explained, before diving deeper into the specific steps in the next chapter.

The aim of the algorithm is to grow a truss out of reclaimed elements, whereafter a 3D model of a truss bridge is generated and calculated. The algorithm should be applicable to different sizes of databases and different sets of lengths and profiles. It should also be applicable to different bridge spans.

In an algorithm , a balance should be found between making smart steps and the amount of iterations needed to find a solution. The smarter the algorithm, e.g. the smarter a suitable element is picked out of the database, the less iterations needed to find the best solution. However, the downside to scripting very calculated steps is that it may result in cases where the seeked for option is not found and an alternative is needed. More specifically, to make an algorithm applicable to all kinds of stocks, all possible alternatives need to be considered so that every time, a correct element may be picked out of the stock. This results in a very elaborate script with a lot of debugging.

This is why it was chosen in this thesis to keep the generation of the truss quite random, so that lots of iterations are needed to find a suitable and efficient design, but the algorithm will be able to provide a solution to all types of stocks

More specific, during the growth process, elements will be picked out of the database randomly (considering a couple of constraints), whereafter the resulting geometry is found. Subsequently, the axial forces in the elements may be calculated. Next, the elements are tested on their capacity. If their capacity is sufficient, the considered configuration is stored as a valid option. This process will be run multiple times until a certain amount of valid options are found whereafter the most efficient option can be chosen. This way the algorithm can be kept quite simple but by iterating, a relatively efficient option can be found before going on to the next step. This process is visualised in figure 6.8.

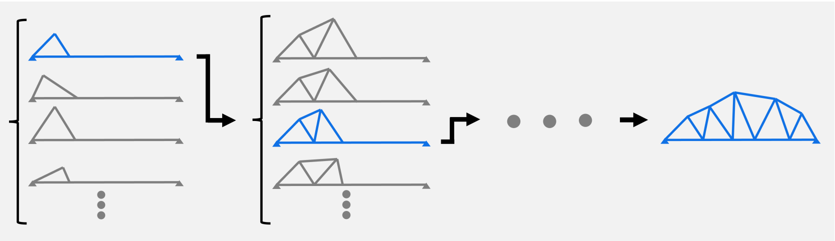


Figure 6.8: General steps in the algorithm

By setting up the algorithm in this manner, it is also not possible to make a much more calculated decision on the next to be picked element. The geometry will change each time different elements are chosen. The changed geometry results in changed forces in the elements. These forces cannot be predicted on forehand. Therefore, the choice of element will keep a random aspect.

To improve the final truss design, the algorithm may be performed several times. As the algorithm contains random steps, a different truss design will be generated every time the algorithm is run. This way, a solution cloud of valid truss options can be created. These may may be compared to each other to select the best truss design according to different objectives:

- The truss with the lowest environmental impact

  the main goal of this thesis

- The truss with the lowest mass

  to make transportation easier

- The truss with the highest capacity utilisation

  also the 'smartest' truss

- The truss with the least number of elements

  to reduce the number of connections

- The trusses without hollow sections welded under an angle lower than 30 °.

  to be able to apply standard Eurocode formulae to check the resistance of the connections

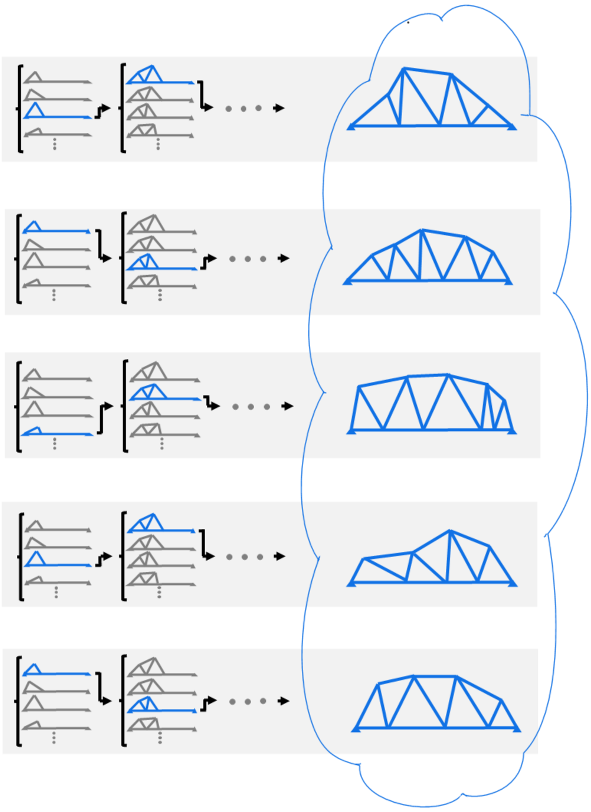


Figure 6.9: Generating a solution cloud

Besides comparison of trusses, an additional step is provided where truss bridges are generated consisting of two separately generated unique trusses. These truss bridges may thereafter be compared according to the same objectives.

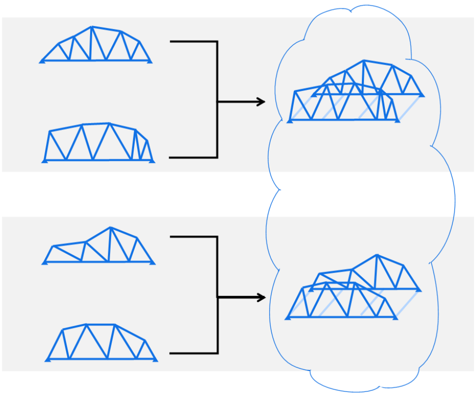


Figure 6.10: Generating a solution cloud

7

Steps in the algorithm

In the screenshot below an overview is given of the whole grasshopper file.

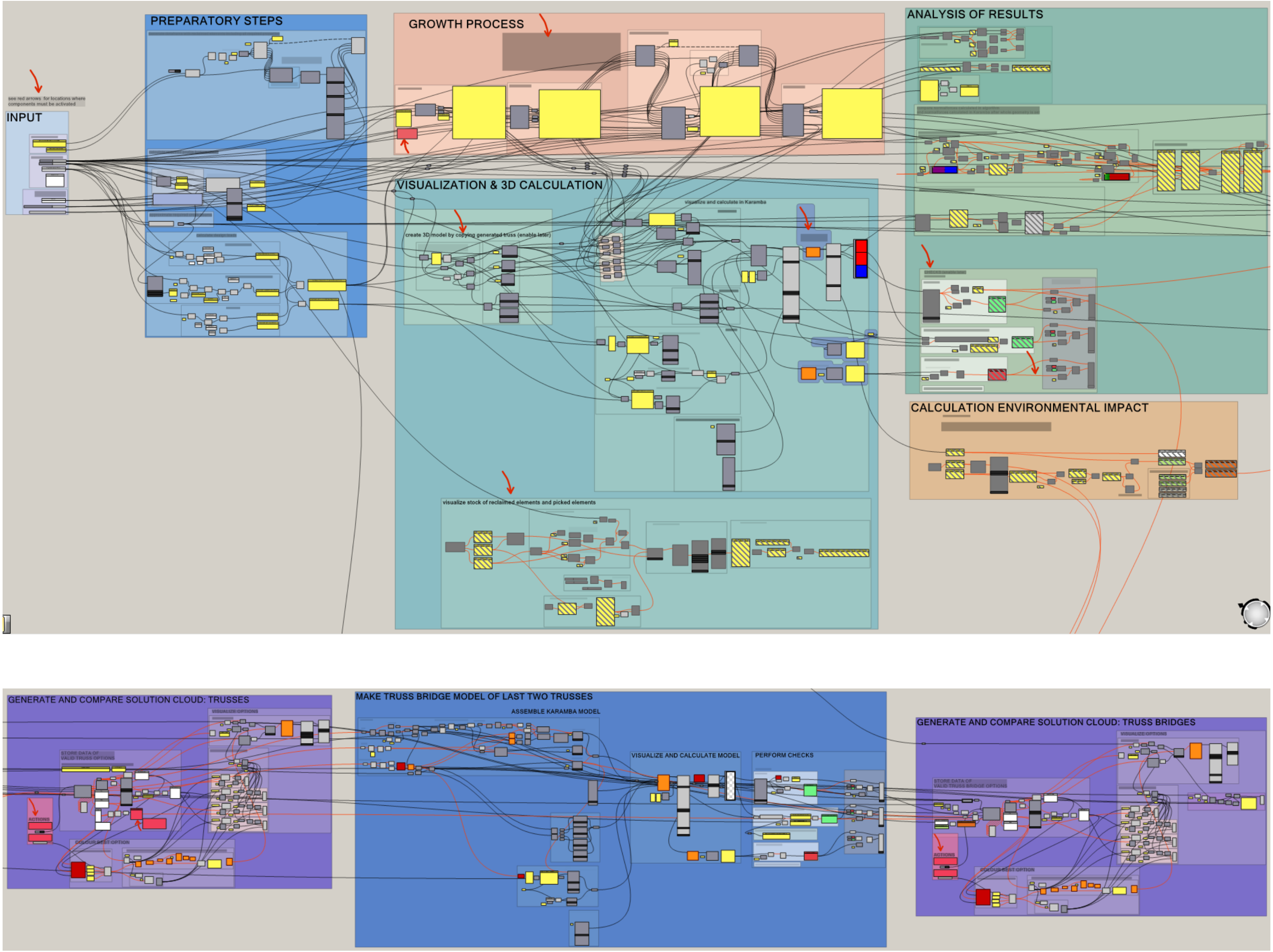

see red arrows for locations where components must be activated
INPUT
PREPARATORY STEPS
GROWTH PROCESS
ANALYSIS OF RESULTS
VISUALIZATION & 3D CALCULATION
create 3D model by copying generated truss (enable later)
visualize and calculate in Karamba
visualize stock of reclaimed elements and picked elements
CALCULATION ENVIRONMENTAL IMPACT
GENERATE AND COMPARE SOLUTION CLOUD: TRUSSES
MAKE TRUSS BRIDGE MODEL OF LAST TWO TRUSSES
ASSEMBLE KARAMBA MODEL
VISUALIZE AND CALCULATE MODEL
PERFORM CHECKS
GENERATE AND COMPARE SOLUTION CLOUD: TRUSS BRIDGES

Figure 7.1: Grasshopper file

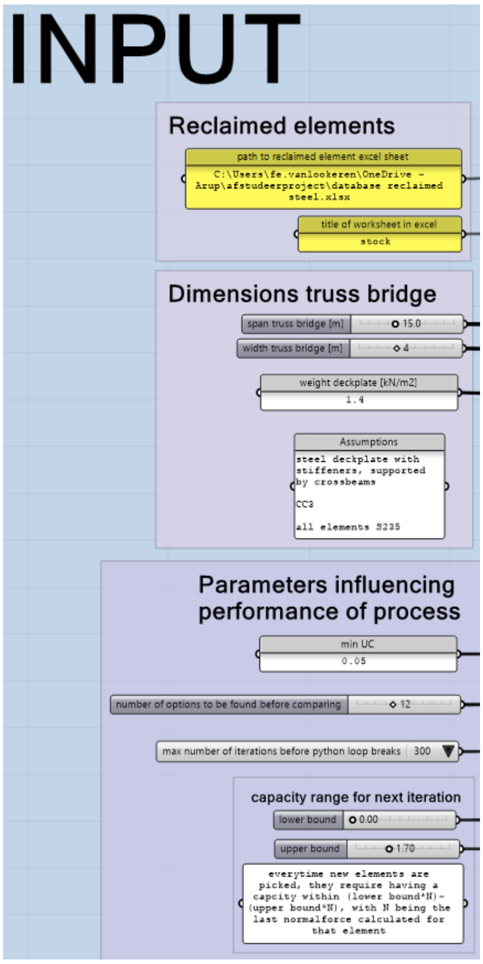
The design process starts off with a group of input requirements. The rest of the algorithm will then run automatically, except for the locations where red arrows are positioned. These are intermediate steps where something needs to be activated so that the algorithm can run smoothly. After the input group, a blue group is made where various preparatory steps are performed. Afterwards, the growth process algorithm can start off, which is grouped inside the orange box. During the growth process, intermediate steps are visualized with Karamba. In the visualization group (in teal) also options are provided to make a 3D model or to visualize the stock of elements. In the same group. Karamba will also calculate the model using various types of analyses. In the green group of components, the results of the calculations are analysed. The orange group will calculate the environmental impact factors of the truss design. The purple groups contain components to generate the solution clouds: first the truss solution cloud and then the truss bridge solution cloud. The blue group in between contains Karamba components to model and calculate the truss bridge model in order to decide if it is a valid option that may be included in the last solution cloud.

The algorithm will be explained by zooming into the consecutive groups whereby the workflow is visualized in flowcharts in corresponding colours and the content of the python scripts will be explained with figures and flowcharts. The full scripts can be found in appendix C. Screenshots of each Grasshopper group can be found in appendix D.

7.1. Inputs

The inputs shown in figure 7.2 are required to perform the design process. The main input parameters are the width and length of the desired truss bridges and the available stock of reclaimed steel elements. The weight of the deckplate can be changed but the current value is suitable to design a truss bridge using a steel deckplate with stiffeners, supported by crossbeams. This value is elaborated in chapter 8.1.

Lastly, some additional parameters are included to control the performance of the design process. These parameters influence the speed and the result of the design process. Hereby the option is given to the user to perform a quick design process whereby the results may be less optimal, or have the design process run more thoroughly and therefore longer. This will result in more optimal designs. The parameters are clarified in the following paragraphs.



INPUT

Reclaimed elements

path to reclaimed element excel sheet: C:\Users\fe.vanlookeren\OneDrive - Arup\afstudeerproject\database reclaimed steel.xlsx

title of worksheet in excel: stock

Dimensions truss bridge

span truss bridge [m]: 15.0

width truss bridge [m]: 4

weight deckplate [kN/m2]: 1.4

Assumptions

steel deckplate with stiffeners, supported by crossbeams

CC3

all elements S235

Parameters influencing performance of process

min UC: 0.05

number of options to be found before comparing: 12

max number of iterations before python loop breaks: 300

capacity range for next iteration

lower bound: 0.00

upper bound: 1.70

everytime new elements are picked, they require having a capcity within (lower bound*N)-(upper bound*N), with N being the last normalforce calculated for that element

Figure 7.2: Input parameters

7.2. Generating the reclaimed elements database

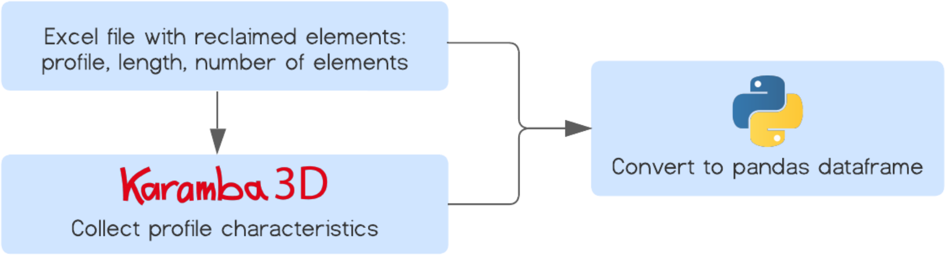

Excel file with reclaimed elements: profile, length, number of elements
Karamba 3D
Collect profile characteristics
Convert to pandas dataframe

Figure 7.3: Grasshopper steps

7.2.1. Initial database

The excel file including the available reclaimed elements must contain the following columns: profile (e.g. HEA300), length (in meters) and number (of elements), with those names as titles. The data is then converted into a pandas dataframe.

7.2.2. Finding additional element properties

In order to calculate the capacity of the elements, the cross sectional area, the area moment of inertia and the buckling coefficients of the profiles need to be included in the database. In addition, the width and height of the profile is needed for further calculation purposes. Luckily, Karamba offers a component that provides all characteristics of all possible components. By sending in a list of profile names, a list of corresponding characteristics is given as output. This information can be collected using an anemone loop and inserted back into the python component so that it may be added to the pandas dataframe.

7.2.3. Calculation of element capacities

Subsequently the capacities of the elements are calculated using the following formulas:

$$F_{t,d} = \frac{f_y * A}{\gamma_M} \tag{7.1}$$

$$F_{c,d} = \xi \frac{f_y * A}{\gamma_{M1}} \tag{7.2}$$

Whereby:

$$\xi = \frac{1}{\phi_z + \sqrt{\phi_z^2 - \bar{\lambda}_z^2}} \leq 1$$

$$\phi_z = 0.5[1 + \alpha_z(\bar{\lambda}_z - 0.2) + \bar{\lambda}_z^2]$$

$$\bar{\lambda}_z = \frac{L_{cr}}{i} \frac{1}{\lambda_1}$$

$$\lambda_1 = \pi \sqrt{\frac{E}{f_y}}$$

$$i = \sqrt{\frac{I_{zz}}{A}}$$

Whereby:

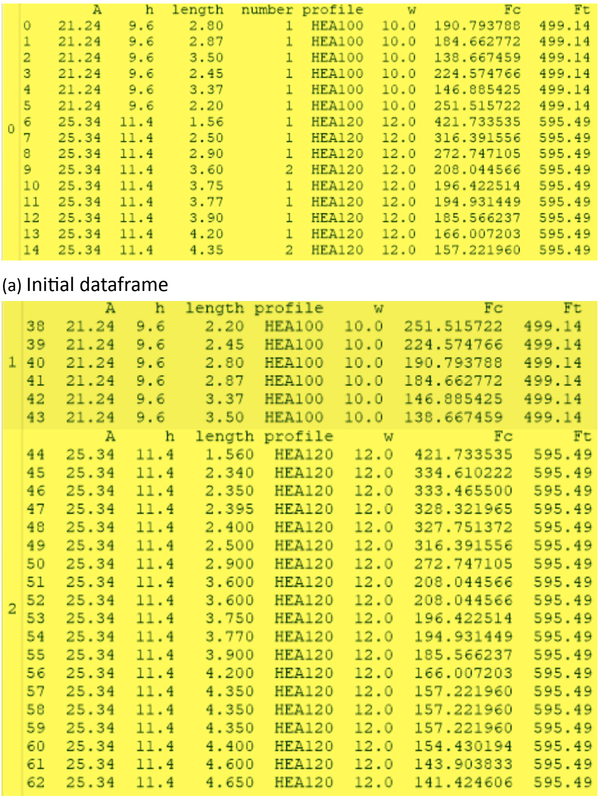
- $\gamma_M = 1$
- $\gamma_{M1} = 1.15$ (due to designing with reclaimed elements)
- $f_y = 23.5kN/cm^2$ (all elements assumed to be S235, using Karamba units)
- $E = 21000kN/cm^2$

The capacities ($F_{c,d}$ and $F_{t,d}$) are added as two additional columns in the database.

7.2.4. Reorganization of the database

Lastly, some alterations are made to the dataframe so it will be more efficient to use. The dataframe is sorted first according to profile name and secondly to increasing length. Additionally, the column with the number of elements is removed while all elements that are available multiple times are each added to the dataframe. The final dataframe includes the following columns: profile, length, cross sectional area, compressive and tensile capacities, width and height.

Lastly, the indices are redefined. These indices will be important in the rest of the algorithm. All the elements now have their own unique 'element id'. In subfigure 7.4a the first 15 items of the initial dataframe can be seen. In subfigure 7.4b the dataframe has been reorganized. The profiles HEA100 and HEA120 obtained new indices and are shuffled so that they are sorted by length. The elements that were present multiple times in the dataframe have each individually been added to the new reorganized dataframe.



```
            A     h  length  number profile     w          Fc      Ft
  0   21.24   9.6    2.80       1  HEA100  10.0  190.793788  499.14
  1   21.24   9.6    2.87       1  HEA100  10.0  184.662772  499.14
  2   21.24   9.6    3.50       1  HEA100  10.0  138.667459  499.14
  3   21.24   9.6    2.45       1  HEA100  10.0  224.574766  499.14
  4   21.24   9.6    3.37       1  HEA100  10.0  146.885425  499.14
  5   21.24   9.6    2.20       1  HEA100  10.0  251.515722  499.14
0 6   25.34  11.4    1.56       1  HEA120  12.0  421.733535  595.49
  7   25.34  11.4    2.50       1  HEA120  12.0  316.391556  595.49
  8   25.34  11.4    2.90       1  HEA120  12.0  272.747105  595.49
  9   25.34  11.4    3.60       2  HEA120  12.0  208.044566  595.49
  10  25.34  11.4    3.75       1  HEA120  12.0  196.422514  595.49
  11  25.34  11.4    3.77       1  HEA120  12.0  194.931449  595.49
  12  25.34  11.4    3.90       1  HEA120  12.0  185.566237  595.49
  13  25.34  11.4    4.20       1  HEA120  12.0  166.007203  595.49
  14  25.34  11.4    4.35       2  HEA120  12.0  157.221960  595.49
```

(a) Initial dataframe

```
             A    h  length profile     w          Fc      Ft
  38  21.24  9.6    2.20  HEA100  10.0  251.515722  499.14
  39  21.24  9.6    2.45  HEA100  10.0  224.574766  499.14
1 40  21.24  9.6    2.80  HEA100  10.0  190.793788  499.14
  41  21.24  9.6    2.87  HEA100  10.0  184.662772  499.14
  42  21.24  9.6    3.37  HEA100  10.0  146.885425  499.14
  43  21.24  9.6    3.50  HEA100  10.0  138.667459  499.14
             A     h  length profile     w          Fc      Ft
  44  25.34  11.4   1.560  HEA120  12.0  421.733535  595.49
  45  25.34  11.4   2.340  HEA120  12.0  334.610222  595.49
  46  25.34  11.4   2.350  HEA120  12.0  333.465500  595.49
  47  25.34  11.4   2.395  HEA120  12.0  328.321965  595.49
  48  25.34  11.4   2.400  HEA120  12.0  327.751372  595.49
  49  25.34  11.4   2.500  HEA120  12.0  316.391556  595.49
  50  25.34  11.4   2.900  HEA120  12.0  272.747105  595.49
  51  25.34  11.4   3.600  HEA120  12.0  208.044566  595.49
  52  25.34  11.4   3.600  HEA120  12.0  208.044566  595.49
2 53  25.34  11.4   3.750  HEA120  12.0  196.422514  595.49
  54  25.34  11.4   3.770  HEA120  12.0  194.931449  595.49
  55  25.34  11.4   3.900  HEA120  12.0  185.566237  595.49
  56  25.34  11.4   4.200  HEA120  12.0  166.007203  595.49
  57  25.34  11.4   4.350  HEA120  12.0  157.221960  595.49
  58  25.34  11.4   4.350  HEA120  12.0  157.221960  595.49
  59  25.34  11.4   4.350  HEA120  12.0  157.221960  595.49
  60  25.34  11.4   4.400  HEA120  12.0  154.430194  595.49
  61  25.34  11.4   4.600  HEA120  12.0  143.903833  595.49
  62  25.34  11.4   4.650  HEA120  12.0  141.424606  595.49
```

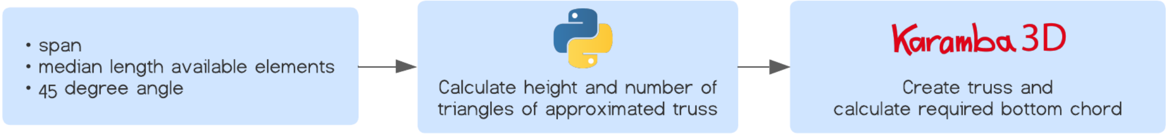
(b) Reorganized dataframe

Figure 7.4: Reclaimed element dataframe

7.2.5. Storing of database

The dataframe is pickled, and thereby stored as re1.pkl. By calling the following function, the dataframe can be reopened in other GHPython components: re=pd.read_pickle(”re1.pkl”)

7.3. Calculating the required capacity of the bottom chord



- span
- median length available elements
- 45 degree angle

Calculate height and number of triangles of approximated truss

Create truss and calculate required bottom chord

Figure 7.5: Grasshopper steps

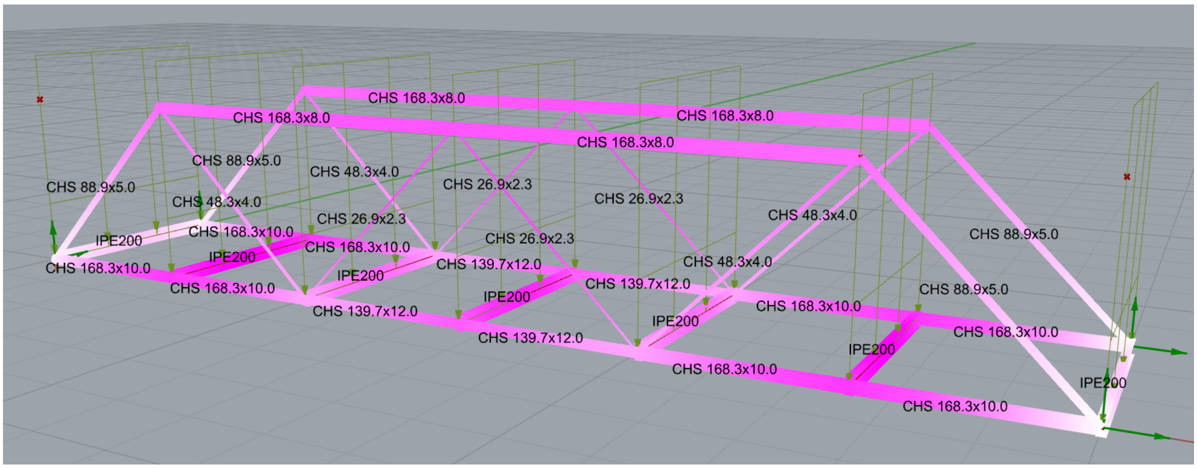


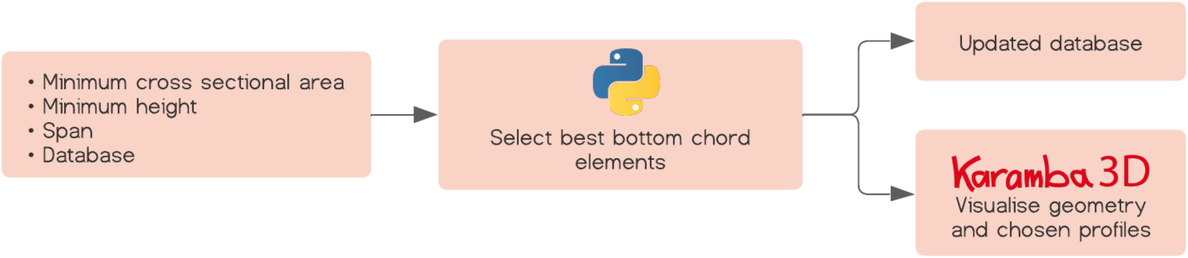
Figure 7.6: Standard Warren truss with optimised cross sections

To calculate the required capacity of the bottom truss, a virtual standard Warren truss with the same span is calculated whereby the cross section optimiser component from Karamba is used to determine the minimum required profiles. The dimensions of the representative truss will influence the final bottom chord profile. The lower the height of the truss and the lower the amount of triangles in the truss, the larger the bottom chord will need to be to resist tensile forces. Assuming the truss that will be generated by the growth method will not be the most optimal design in terms of material efficiency, the representative truss should not have too optimised dimensions either. However, considering a worst case scenario will let the bottom chord be too overdimensioned in the final design. A balance thus needs to be found to approximate the required bottom chord profile.

The decision was therefore made to use the median length of the available reclaimed elements and use a standard angle of 45° to generate a standard warren truss which will approach the final design as good as possible.

The calculated height and number of triangles, together with other inputs to generate a representative 3D model, are the inputs to a cluster of Grasshopper components which will generate the truss bridge geometry and calculate it using Karamba. In the model, additional cross beams are added when the width of the triangles is more than 3.5 meters. The reason for this is that steel deck plates usually do not span larger distances. The bottom chord will in this case need to be able to carry an additional pointload, situated in the middle of each triangle. The output is the largest cross sectional area of the bottom chord of the calculated truss. This cross sectional area will be used as a minimum value for the bottom chord element to be chosen in the next step.

7.4. Choosing the bottom chord elements


- Minimum cross sectional area
- Minimum height
- Span
- Database

Select best bottom chord elements

Updated database

Karamba 3D Visualise geometry and chosen profiles

Figure 7.7: Grasshopper steps

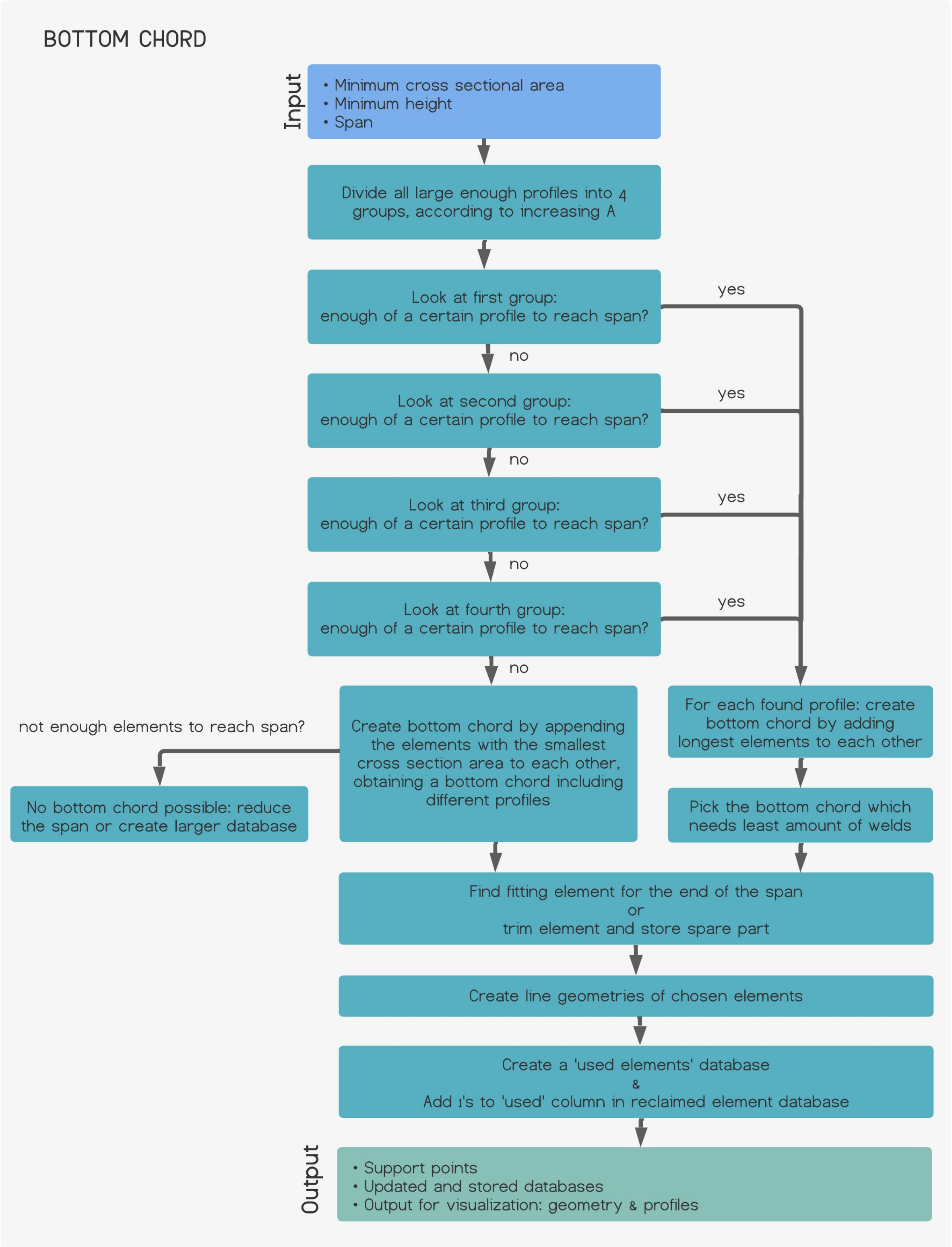

BOTTOM CHORD
Input
• Minimum cross sectional area
• Minimum height
• Span
Divide all large enough profiles into 4 groups, according to increasing A
Look at first group: enough of a certain profile to reach span?
yes
no
Look at second group: enough of a certain profile to reach span?
yes
no
Look at third group: enough of a certain profile to reach span?
yes
no
Look at fourth group: enough of a certain profile to reach span?
yes
no
not enough elements to reach span?
Create bottom chord by appending the elements with the smallest cross section area to each other, obtaining a bottom chord including different profiles
For each found profile: create bottom chord by adding longest elements to each other
No bottom chord possible: reduce the span or create larger database
Pick the bottom chord which needs least amount of welds
Find fitting element for the end of the span or trim element and store spare part
Create line geometries of chosen elements
Create a 'used elements' database & Add 1's to 'used' column in reclaimed element database
Output
• Support points
• Updated and stored databases
• Output for visualization: geometry & profiles

Figure 7.8: Flowchart algorithm bottom chord

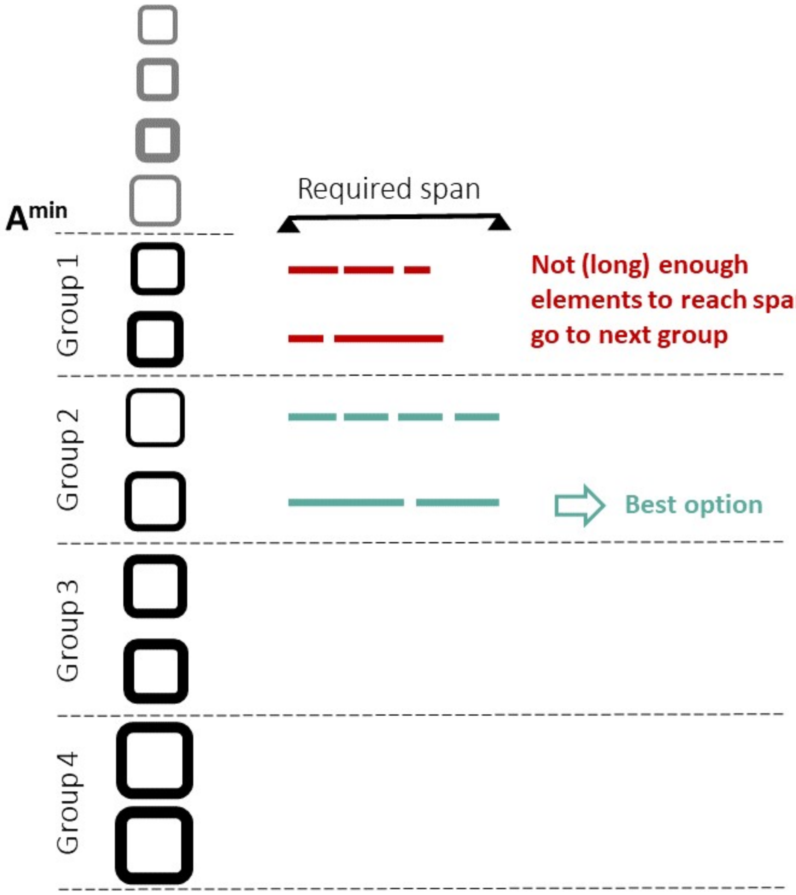


Figure 7.9: Visualization of the division of the profiles into groups

In figure 7.7 the main Grasshopper steps are shown. A python script is included wherein the algorithm, described in figure 7.8 is scripted. Figure 7.9 clarifies the steps in the algorithm.

The inputs to the script are the span of the truss bridge, the minimum required cross sectional area of the bottom chord and the height of the crossbeams and therefore the minimum required height of the bottom chord so that the crossbeams may be connected.

The main idea in this algorithm is to seek the best profile for the bottom chord whereby three aims are kept in mind. The first one is to select the profile with a cross sectional area closest to the minimum required cross sectional area. The second aim is to select a profile which can reach the required span with a relatively low number of elements, so that a less welds are needed. The last goal is to try and form a bottom chord with elements with the same profile, so that connections can easily be made. Because we have these three aims, the following method is followed to find the best option for the bottom chord.

First the elements are selected which comply to the minimum cross sectional area and minimum height constraint and which are not circular hollow sections (so that only straight end cuts are needed to weld diagonals onto this bottom chord). These are then divided into 4 groups, beginning with smallest cross sectional areas in the first group to the largest in the last group. First, the first group is checked on the existence of a profile in this group which has (long) enough elements to reach the whole length when connected to each other. If this does not exist, the second group is analysed and so on. When at least one profile is found in a certain group, the bottom chord is created by adding the longest elements to each other of the chosen profiles. The option which has the least welds is chosen for the final design

If in all the groups no single profile exist with (long) enough elements, the second option is considered: creating a bottom chord with different profiles. All elements with a cross sectional area above the minimum value are now put in a single group and the bottom chord is generated through sequencing the elements with the smallest profiles.

If there are not enough profiles to reach the span, an error message displays informing that the bottom chord cannot be constructed. The database is in that case too small or does not have large or long enough elements.

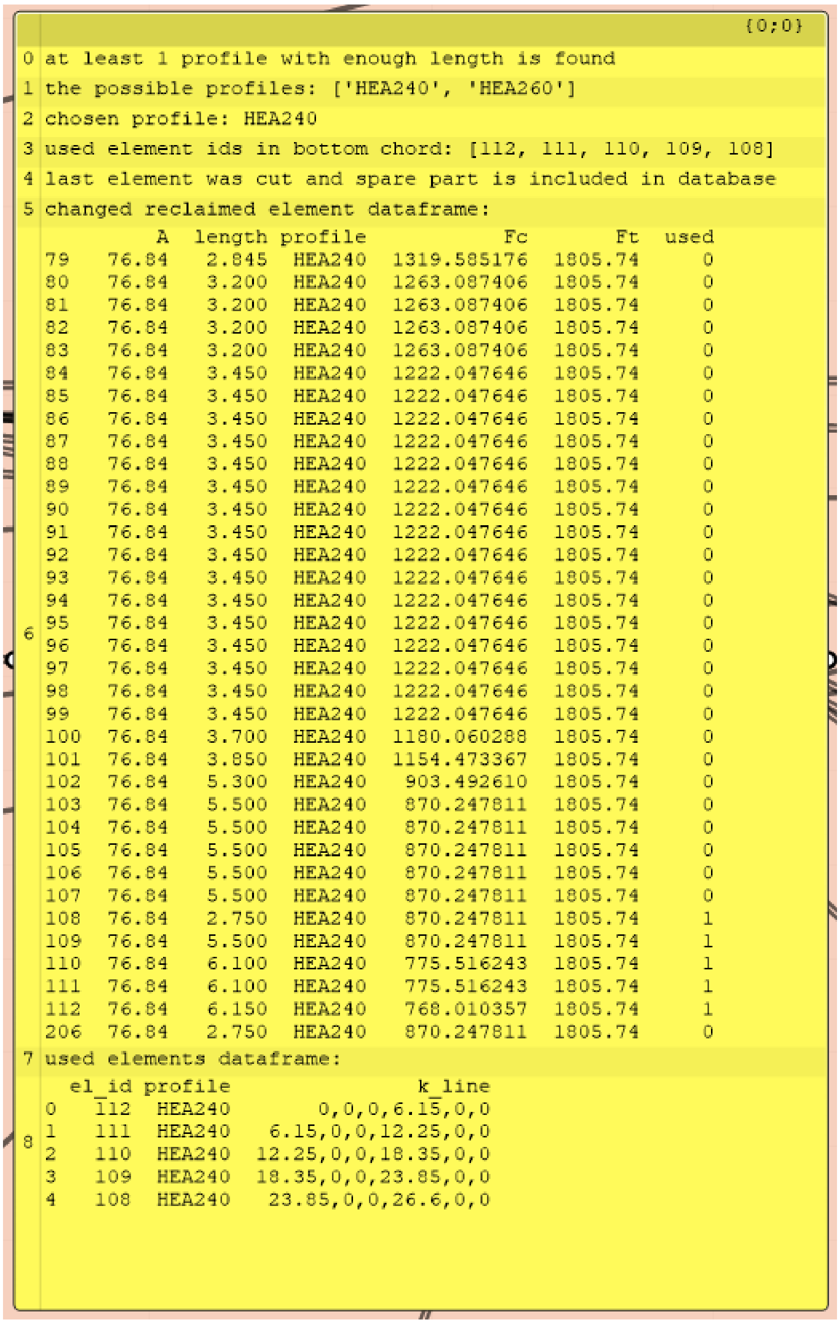
If the bottom chord can be constructed, the creation of the bottom chord ends with searching for a last element that exactly fits into the required span of the truss. If this element does not exist, a longer element is chosen and trimmed. The spare part is again included in the reclaimed element database.

The last steps of the algorithm include creating the Rhino geometry of the elements in the bottom chord and creating a new database which includes the chosen elements. The following information is gathered: element ID, profile and line geometry. This way a structured overview of the final truss geometry and the corresponding elements is made and can easily be exported to Karamba to visualize and calculate the design. This database will be called the 'used element database'.

In addition, an extra column is made in the reclaimed element database which is called 'used'. In this column 1's will signify elements that are already picked and 0's elements that are still available. In the next steps, random elements will always be chosen which satisfy the condition of having a 0 in the 'used' column.

Lastly, the support points of the truss are defined and the adapted reclaimed and used element databases are stored in a new pickle file, which can be opened in the next step.

The reclaimed element database and used element database are printed out onto the panel next to the python component, as can be seen in figure 7.10. In the reclaimed element database only the profile chosen for the bottom chord is printed. Elements 108-111 now have a 1 in the 'used' column as they have been picked for the bottom chord. These elements are also included in the new used element database. In addition, in the reclaimed element database, a spare part has been included at the end of the database. This spare part has been cut from the original element 108. The original element had a length of 5.5 m and has now been cut in half.



```
                                                                  {0;0}
0 at least 1 profile with enough length is found
1 the possible profiles: ['HEA240', 'HEA260']
2 chosen profile: HEA240
3 used element ids in bottom chord: [112, 111, 110, 109, 108]
4 last element was cut and spare part is included in database
5 changed reclaimed element dataframe:
           A  length profile           Fc       Ft  used
  79   76.84   2.845  HEA240  1319.585176  1805.74     0
  80   76.84   3.200  HEA240  1263.087406  1805.74     0
  81   76.84   3.200  HEA240  1263.087406  1805.74     0
  82   76.84   3.200  HEA240  1263.087406  1805.74     0
  83   76.84   3.200  HEA240  1263.087406  1805.74     0
  84   76.84   3.450  HEA240  1222.047646  1805.74     0
  85   76.84   3.450  HEA240  1222.047646  1805.74     0
  86   76.84   3.450  HEA240  1222.047646  1805.74     0
  87   76.84   3.450  HEA240  1222.047646  1805.74     0
  88   76.84   3.450  HEA240  1222.047646  1805.74     0
  89   76.84   3.450  HEA240  1222.047646  1805.74     0
  90   76.84   3.450  HEA240  1222.047646  1805.74     0
  91   76.84   3.450  HEA240  1222.047646  1805.74     0
  92   76.84   3.450  HEA240  1222.047646  1805.74     0
  93   76.84   3.450  HEA240  1222.047646  1805.74     0
  94   76.84   3.450  HEA240  1222.047646  1805.74     0
6 95   76.84   3.450  HEA240  1222.047646  1805.74     0
  96   76.84   3.450  HEA240  1222.047646  1805.74     0
  97   76.84   3.450  HEA240  1222.047646  1805.74     0
  98   76.84   3.450  HEA240  1222.047646  1805.74     0
  99   76.84   3.450  HEA240  1222.047646  1805.74     0
  100  76.84   3.700  HEA240  1180.060288  1805.74     0
  101  76.84   3.850  HEA240  1154.473367  1805.74     0
  102  76.84   5.300  HEA240   903.492610  1805.74     0
  103  76.84   5.500  HEA240   870.247811  1805.74     0
  104  76.84   5.500  HEA240   870.247811  1805.74     0
  105  76.84   5.500  HEA240   870.247811  1805.74     0
  106  76.84   5.500  HEA240   870.247811  1805.74     0
  107  76.84   5.500  HEA240   870.247811  1805.74     0
  108  76.84   2.750  HEA240   870.247811  1805.74     1
  109  76.84   5.500  HEA240   870.247811  1805.74     1
  110  76.84   6.100  HEA240   775.516243  1805.74     1
  111  76.84   6.100  HEA240   775.516243  1805.74     1
  112  76.84   6.150  HEA240   768.010357  1805.74     1
  206  76.84   2.750  HEA240   870.247811  1805.74     0
7 used elements dataframe:
    el_id profile               k_line
  0   112  HEA240        0,0,0,6.15,0,0
  1   111  HEA240     6.15,0,0,12.25,0,0
8 2   110  HEA240    12.25,0,0,18.35,0,0
  3   109  HEA240    18.35,0,0,23.85,0,0
  4   108  HEA240     23.85,0,0,26.6,0,0
```

Figure 7.10: Output python component

7.5. Choosing the truss elements

In this paragraph a general explanation is given on how the next elements in the truss will be selected.

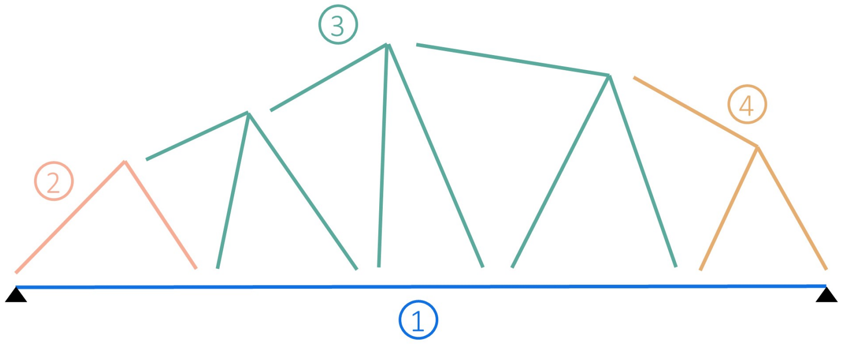


Figure 7.11: Order of choosing the elements

7.5.1. First two elements

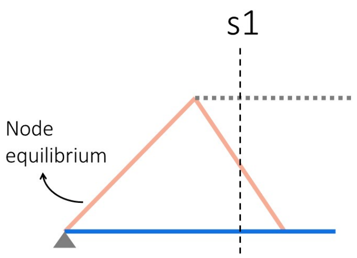


Figure 7.12: First two elements

After generation of the bottom chord, the growth process will start by picking the first two elements. These two elements are positioned so that they form a triangle and the two lower angles have a mean of 45°. A certain constraint needed to be set to define their positions and by defining the angle instead of certain height or width, the script will be applicable to all kinds of databases with variant element dimensions. The first two elements will now always be positioned in a logical way.

Other than needing two elements to determine their orientation in the resulting triangle, a second element is also needed to calculate the force in the first element. The width of the resulting triangle will determine the pointload occurring in the first node. The force in the first element can then be found by setting up the force equilibrium in the first node.

The force in the second element is more complicated to define. As mentioned above, the method of section is used to determine the forces in the rest of the elements. However, to make a section through this second element, the section will also need to pass through a third element which is not yet defined. The orientation of this element can only be known when also a fourth element is chosen, as the geometry depends on the length of the chosen elements. The required capacity of these last elements again depend on the next elements, and so on. It is thus required to make some assumptions at a certain point so that the truss can be grown step by step. It is not possible to generate a large part of the geometry and only then check capacities. In that case the whole geometry will need to be changed every time an element is not sufficient, which will always be the case. This results in an unsolvable loop.

In the algorithm, a minimal amount of elements is selected whereafter these are checked with some assumptions on the position of the not yet chosen elements. Only after elements are found which comply to the required capacities, will the next elements be chosen. The choice of the orientation of the virtual element will have to be conservative so that the option will still fulfill when the final geometry is set and results in different forces than first assumed. To decide on the orientation of the virtual element, an analysis of forces in trusses is performed. More specific, the effect of the orientation of the top chord on the force in the right facing diagonals needs to be understood.

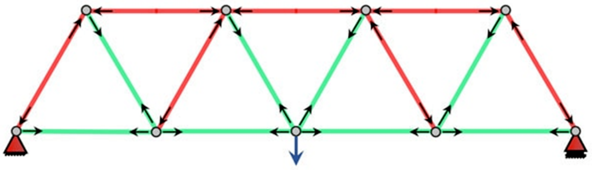


Figure 7.13: Forces in a standard Warren truss (Carroll, 2019)

In a standard warren truss these right-facing diagonals will have a tensile force in the first half of the truss and compressive force in the second half. In figure 7.13 the tensile forces are shown in blue while the compressive forces are shown in red. From left to right, the tensile force will decrease and subsequently the compressive force will increase. When in the first half the top chord inclines upwards instead of staying horizontal, the tensile forces in the analysed diagonals will decrease. In the second half of the truss, when the top chord inclines downwards instead of staying horizontal, the compressive forces in the analysed diagonals will decrease as well.

When introducing the boundary condition that the top chord may only be orientated horizontal or upwards in the first half of the truss and horizontal or downwards in the second half of the truss, the most conservative orientation of the top chord along the whole truss will be horizontal. This way the analysed diagonals will have the highest tensile or compressive force possible. This boundary condition will also be beneficial to the resulting stiffness of the final truss, as the larger the height in the middle of the truss, the stiffer the structure and the lower the vertical deflection.

One problem that may occur with this decision is that in the middle, where the tensile and compressive forces in the diagonals are the smallest, the force may change direction when becoming smaller. This way, a diagonal may be tested on tensile capacity while it will ultimately need to transfer a compressive force in the final design. The unity checks will be recalculated in the final global calculation of the truss using Karamba. In case guessed elements are not sufficient in the final design, the truss will not be considered as a valid option.

To conclude, the second element of the initial triangle will be calculated with the method of section , whereby a 'virtual' horizontal top chord element is assumed so that the calculation can be performed. This will in most cases be the most conservative assumption. The generation of the first triangle will be repeated until a certain amount of valid options is found. From these options the most efficient one is chosen and the algorithm can continue to the next step.

7.5.2. Middle elements

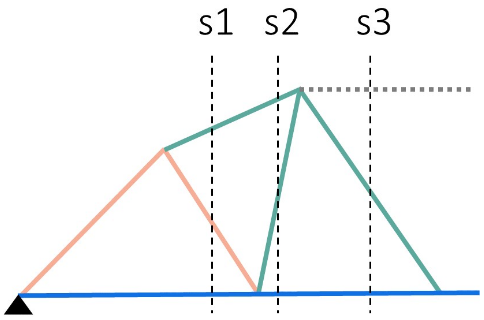


Figure 7.14: Middle elements

The next step is to generate the elements next to the first triangle. This will be done by each time adding three elements, which will form the top chord and the next triangle. This step can then be repeated until the end of the span is nearly reached.

The reason that in every step three elements will be chosen is the following. When first picking two new elements and appending them to the last two nodes, their position will be defined by searching for the intersection point where the ends of the elements meet. The resulting top chord element may be calculated with the current information. However, the force in the second element will depend on the point load in the second node. This point load depends on the width of the second triangle, which is not formed yet. We arrive at the same problem as encountered in the generation of the first triangle. The problem is solved by adding a third element, of which the position is calculated by connecting it to the last node and finding the intersection point with the bottom chord. In addition, a horizontal virtual top chord element is again assumed so that a conservative guess can be made on the required capacity of this third element. This step can now be repeated until the end of the span is nearly reached. In each step, several iterations will be run until again a certain amount of valid options is found and the most efficient one can be chosen.

7.5.3. Last three elements

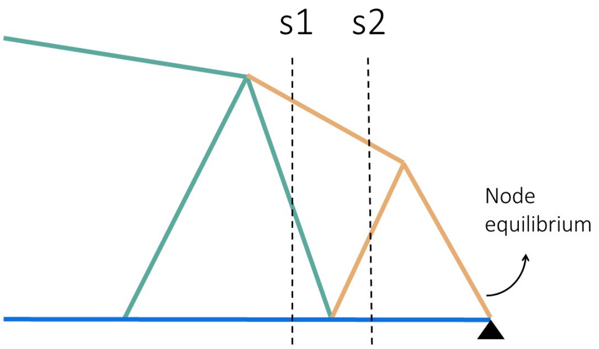


Figure 7.15: Last three elements

When the end of the span is nearly reached, the algorithm will switch over towards finding the last triangle. This step differs slightly from the generation of the previous triangles as the very last element needs to end exactly at the last node.

Like in the generation of the middle part of the truss, first two random elements are chosen and connected to each other. The third element needs to fit exactly in between the intersection points of the first two elements and the end of the span. If a fitting element is not found, a longer element is chosen and trimmed to the correct length. The spare part will again be included in the reclaimed element database. The axial force of the last element can be calculated using a simple node equilibrium equation.

Now the overall idea behind the selection and calculation of the truss elements is explained, the next paragraphs will describe more into detail how the python scripts are structured.

7.6. Choosing the first elements

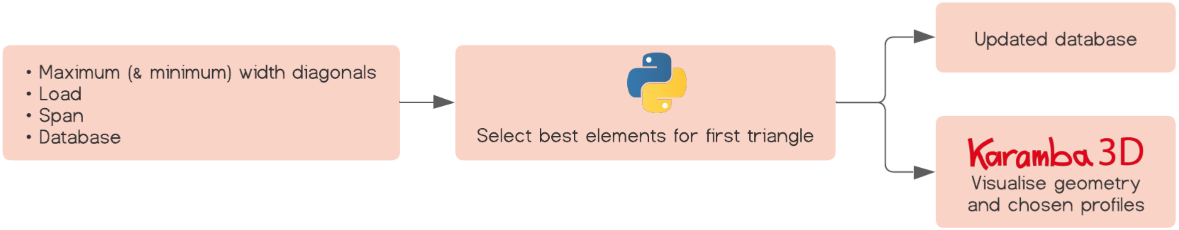

• Maximum (& minimum) width diagonals
• Load
• Span
• Database
Select best elements for first triangle
Updated database
Karamba 3D
Visualise geometry and chosen profiles

Figure 7.16: Grasshopper steps

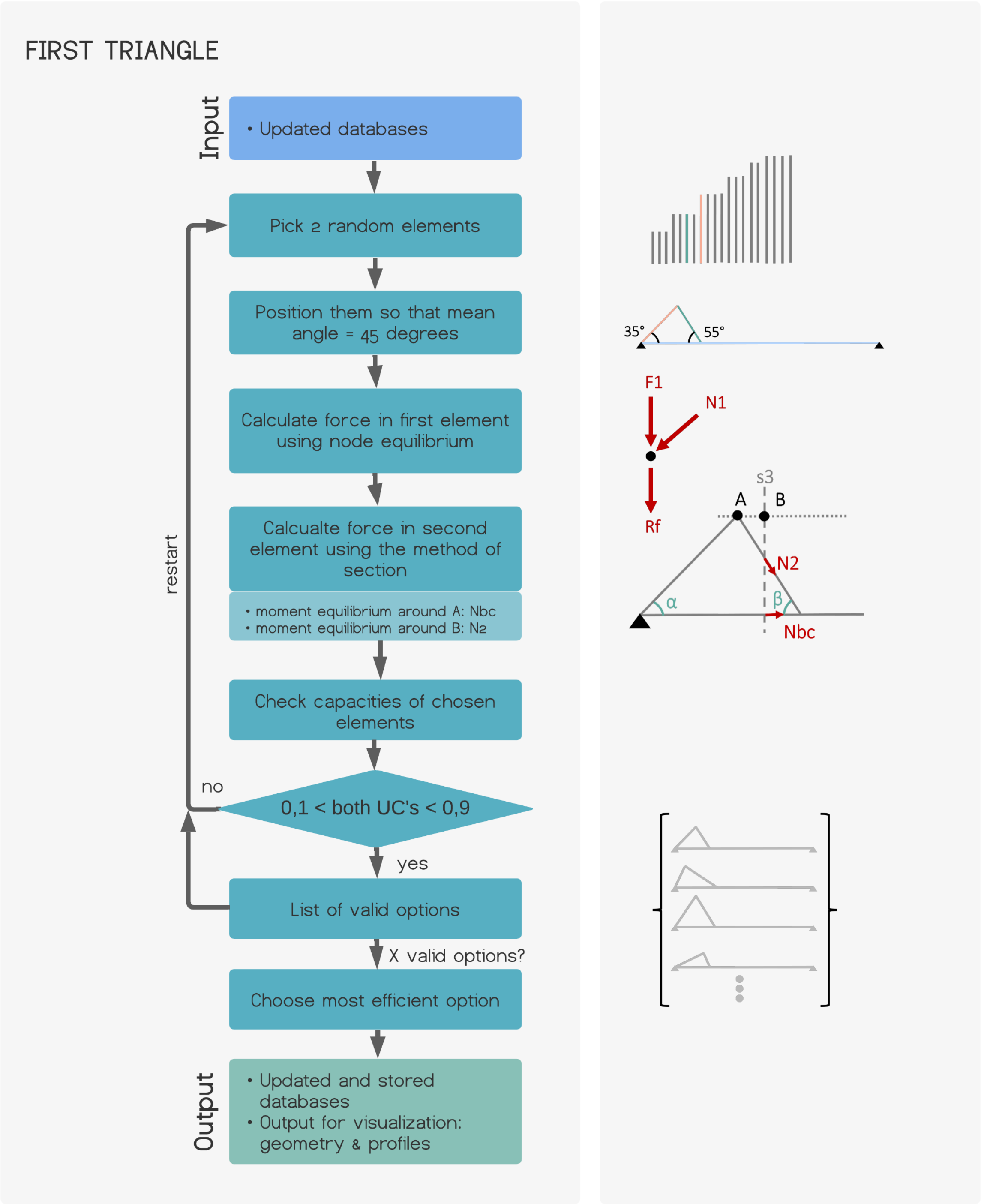

FIRST TRIANGLE
Input
• Updated databases
Pick 2 random elements
Position them so that mean angle = 45 degrees
Calculate force in first element using node equilibrium
Calcualte force in second element using the method of section
• moment equilibrium around A: Nbc
• moment equilibrium around B: N2
Check capacities of chosen elements
0,1 < both UC's < 0,9
no
yes
restart
List of valid options
X valid options?
Choose most efficient option
Output
• Updated and stored databases
• Output for visualization: geometry & profiles
35°
55°
F1
N1
Rf
s3
A
B
N2
α
β
Nbc

Figure 7.17: Flowchart algorithm first triangle

7.6.1. Generating the geometry

First two random elements are chosen out of the reclaimed element database. Two constraints are applied to the selection of elements. The first one is applied to the value in the 'used' column of the database. Only elements may be picked which have a '0'in this column, meaning that they are still available and not already chosen for the bottom chord. The second constraint relates to the constructability of the connections with the bottom chord. The elements may not have larger widths than the bottom chord elements as they will need to be welded onto them.

To position them into a triangle with the lower corners having a mean angle of 45°, the following calculation was made:

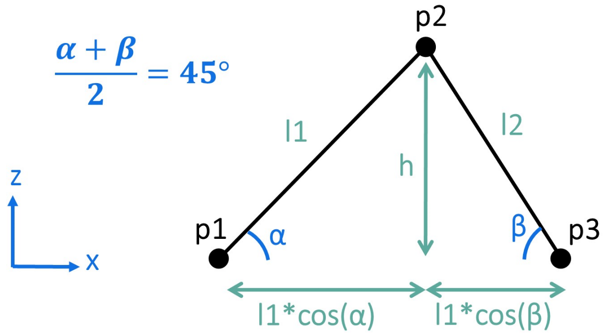


Figure 7.18: Calculating the positions of the first two elements

$$h = sin(\alpha) * l_1 = sin(\beta) * l_2 \Leftrightarrow \alpha = arcsin(sin(\beta)\frac{l_2}{l_1}) \tag{7.3}$$

$$\alpha + \beta = 90° \Leftrightarrow arcsin(sin(\beta)\frac{l_2}{l_1}) + \beta = 90° \Leftrightarrow \beta = arctan(\frac{sin(90°)}{cos(90°) + \frac{l_2}{l_1}}) \tag{7.4}$$

With the angles of the orientation of the two elements found, the coordinates of the two new nodes may be calculated:

$$p2 = (l_1 * cos(\alpha)\ ;\ 0\ ;\ l_1 * sin(\alpha))$$

$$p3 = (l_1 * cos(\alpha) + l_2 * cos(\beta)\ ;\ 0\ ;\ 0)$$

7.6.2. Calculating the axial forces

Next, the force in the first element is calculated by node equilibrium:

$$\sum V = 0 \Leftrightarrow N_{1,V} + F_1 + R_f = 0$$

$$N_{1,V} = N_1 * sin(\alpha)$$

$$\Rightarrow N_1 = \frac{-R_f - F_1}{sin(\alpha)} \tag{7.5}$$

Then the force in the second element is calculated by method of section. The section is positioned halfway between the top node and the end node of the triangle. First moment equilibrium is calculated around point A (see figure in flowchart) to determine the force in the bottom chord:

$$\sum M_A = 0 = (R_f + F_1)x_A - N_{bc}z_A$$

$$\Rightarrow N_{bc} = \frac{(R_f + F_1) * x_A}{z_A} \tag{7.6}$$

Then moment equilibrium is taken around point B so that the force in the second element can be found:

$$\sum M_B = 0 = (R_f + F_1) * x_B - N_{bc}z_A - N_{2,V}(x_B - x_A)$$

$$N_{2,V} = N_2 * sin(\beta)$$

$$\Rightarrow N_2 = \frac{(R_f + F_1) * x_B - N_{bc} * z_A}{sin(\beta) * (x_B - x_A)} \tag{7.7}$$

7.6.3. Finding the best option

Now the forces in the elements are calculated, they can be tested against their capacities. If both Unity Checks are above a specified lower UC limit and below 0.9, they are considered as a valid option. The steps are repeated until a specified amount of valid options are found, whereafter the option with the highest UC's (closest to 0.9) is chosen.

The amount of valid options to be found and the UC limits are parameterized so that they can easily be adapted at the input stage of the design process. Increasing the amount of valid options to be found and compared or increasing the lower UC limit will lead to more efficient truss designs. However, the computational cost will also increase.

The output of this step is again an updated reclaimed element and used element database and re-stored in a new pickle file. Also lists of the used profiles and the line geometries of the elements are made so that they can be transferred to Karamba to visualize the intermediate results.

7.7. Choosing the middle elements

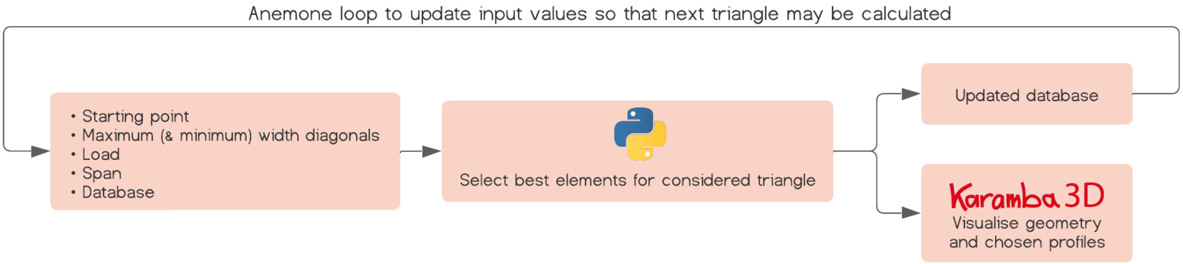

Anemone loop to update input values so that next triangle may be calculated
• Starting point
• Maximum (& minimum) width diagonals
• Load
• Span
• Database
Select best elements for considered triangle
Updated database
Karamba 3D
Visualise geometry
and chosen profiles

Figure 7.19: Grasshopper steps

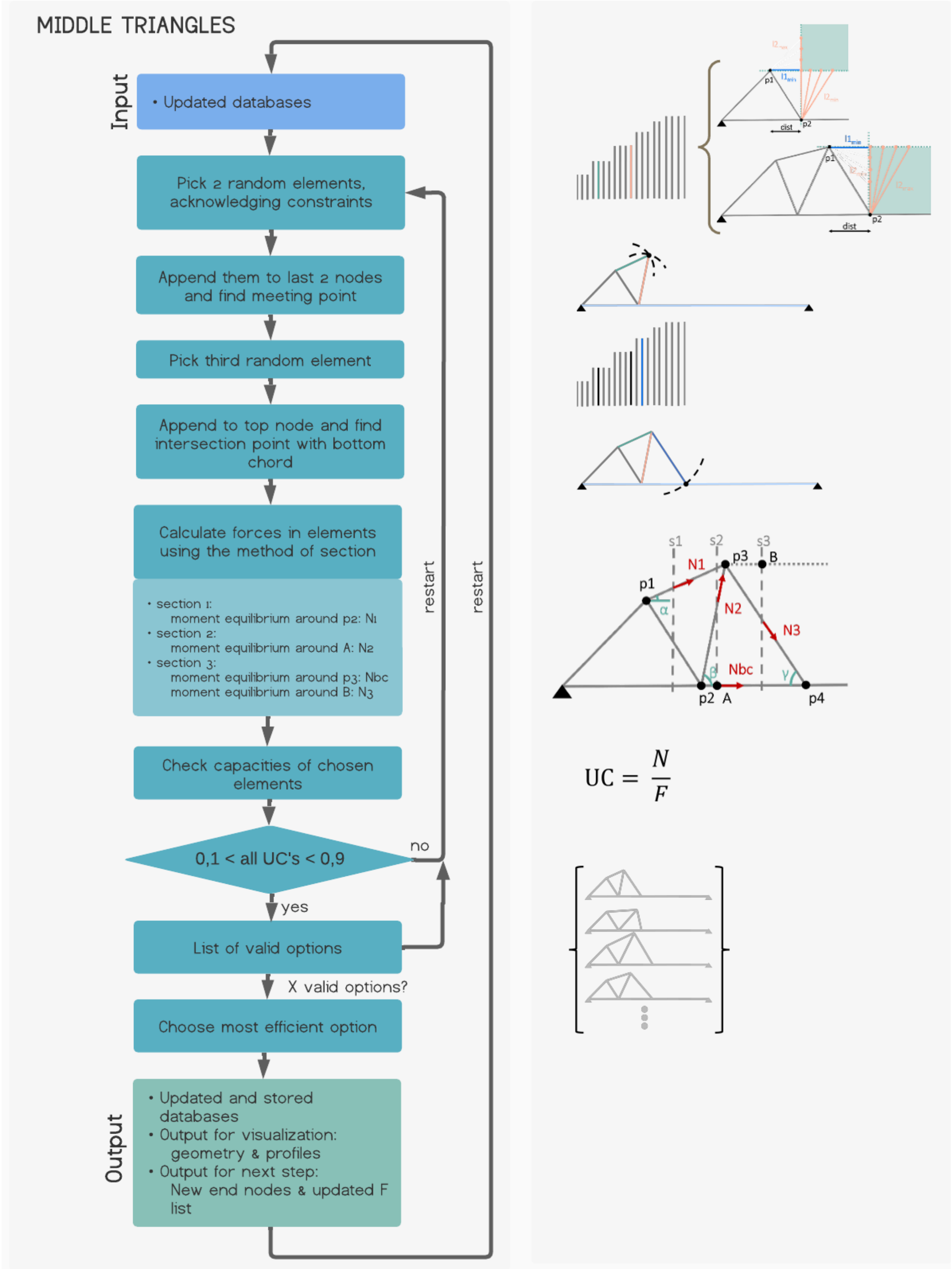

MIDDLE TRIANGLES
Input
• Updated databases
Pick 2 random elements, acknowledging constraints
Append them to last 2 nodes and find meeting point
Pick third random element
Append to top node and find intersection point with bottom chord
Calculate forces in elements using the method of section
• section 1: moment equilibrium around p2: N1
• section 2: moment equilibrium around A: N2
• section 3: moment equilibrium around p3: Nbc moment equilibrium around B: N3
Check capacities of chosen elements
0,1 < all UC's < 0,9
no
yes
List of valid options
X valid options?
Choose most efficient option
Output
• Updated and stored databases
• Output for visualization: geometry & profiles
• Output for next step: New end nodes & updated F list
restart
restart
p1
p2
dist
s1
s2
s3
p3
B
N1
N2
N3
Nbc
α
β
γ
A
p4
$UC = \frac{N}{F}$

Figure 7.20: Flowchart algorithm middle triangles

7.7.1. Anemone loop

In figure 7.19, it can be seen that the python component is connected to an anemone loop. Inside the python component the next three elements are generated, by following steps explained in the flowchart in figure 7.20.

These steps are further described below. When the best triangle option is chosen, the Anemone components will loop the information of this triangle back as new input for the python component so that the next three elements may be chosen. For example the last two nodes are registered and specified as begin points again in the next round so that the next elements can be appended to these last nodes. Furthermore, the list of point loads and their positions is updated in the anemone loop.

A break in the Anemone loop stops the generation of new triangles when the span is nearly reached. The loop will stop when the last node falls into a certain distance from the end of the bottom chord. The current distance is set on 1,5 times the mean width of the generated triangles. This will make sure that the remaining space to generate the last triangle will be large and small enough to find fitting elements. This distance can easily be adjusted.

Now the steps inside the python component will be explained more clearly.

7.7.2. Generating the geometry

The three elements are chosen randomly but with certain constraints applied to it so that their chances of suitability is increased. These are summarised in the following list and afterwards explained into more detail:

1. Constraints on the required length of the element
2. Constraints on the required capacity of the elements
3. Constraint on the maximum width of the profile
4. Condition that the element is not already positioned in the truss
5. Condition that the resulting geometry is not too close to the end of the span

Length of the element

The first constraints are certain minimum and maximum lengths that the elements may have. These constraints need to make sure that the resulting triangle will be applicable. For a triangle to be a valid option, six conditions need to be met, which are summarised below and visualized in figure 7.21.

1. The two elements need to be long enough so that an intersection point is found and a triangle can be formed.
2. The two elements may not differ too much in length, so that again an intersection point is found and a triangle can be formed.
3. The resulting triangle may not face backwards.
4. The top chord element needs to have the correct orientation: inclined upwards or horizontal in the first half of the truss and inclined downwards or horizontal in the second half.
5. The intersection point of the two elements needs to lie before the end node of the last triangle, so that each time a section is made to calculate forces, the same elements are cut.
6. The resulting triangle may not lie below the bottom chord.

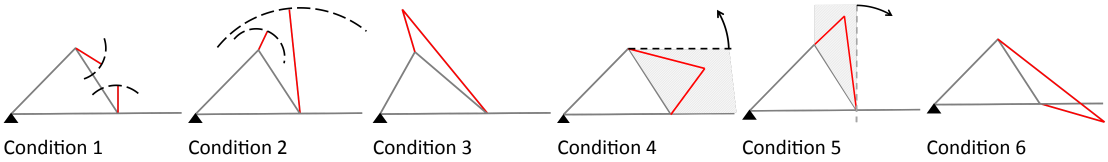
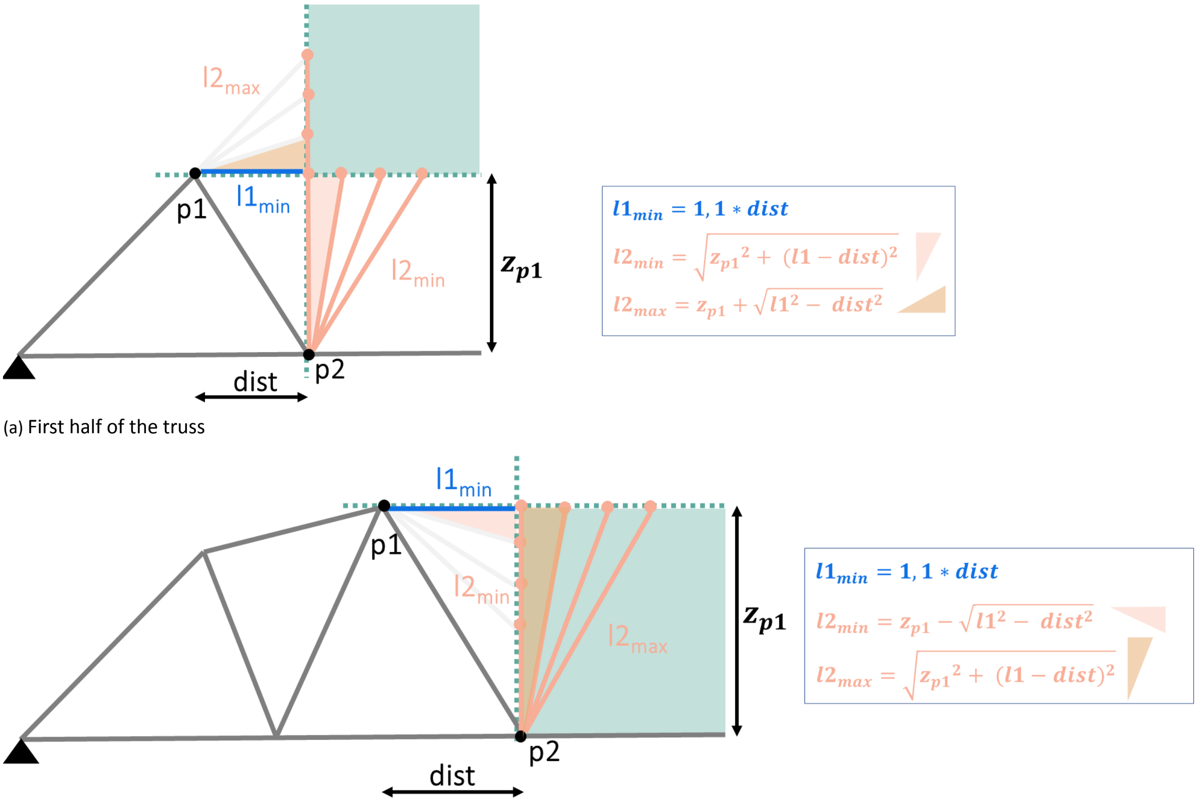


Figure 7.21: Conditions to obtain an applicable triangle

These conditions can be summarised in one green area where the intersection point may lie. This area is different for the first and for the second half of the truss. When analysing this green area, a minimum and maximum length for the to be chosen elements can be derived, so that their intersection point will lie in this area. To make this possible, the constraint on the length of the second element is made dependent on the length of the first picked element. The elements are thus selected one after the other. In figure 7.22 these green areas are pictured, including the resulting length constraints. The equations are derived below.


l2max
p1
l1min
l2min
z_{p1}
dist
p2
$l1_{min} = 1,1 * dist$
$l2_{min} = \sqrt{z_{p1}^2 + (l1 - dist)^2}$
$l2_{max} = z_{p1} + \sqrt{l1^2 - dist^2}$
l1min
p1
l2min
l2max
z_{p1}
p2
dist
$l1_{min} = 1,1 * dist$
$l2_{min} = z_{p1} - \sqrt{l1^2 - dist^2}$
$l2_{max} = \sqrt{z_{p1}^2 + (l1 - dist)^2}$

(a) First half of the truss

(b) Second half of the truss

Figure 7.22: The green areas where the intersection point of the first two elements needs to lie, including the resulting length constraints of these elements.

Starting with the green area of the first half of the truss. The minimum length that the first element will need to have is the horizontal distance from p1 to the green area. As the intersection point of the triangle will need to lie at least a bit further than p2 to make a section between these points possible, the minimum length of the first element is set to 1.1 times this horizontal distance.

The minimum length of the second element depends on the length of the first element. The second element needs to be long enough so that the intersection point will not be drawn lower than the horizontal boundary line of the green area. The various pink lines in the figure show some of the locations where the second element may lie depending on how long the first element is. The length of the second element can then be calculated using the Pythagorean theorem.

Setting a maximum length for the second element is needed so that the intersection point will not be pushed in front of the vertical boundary line of the green area. Again, the maximum allowed length depends on the length of the first element. Its value can be calculated using the Pythagorean theorem.

For the second half of the truss, the same manner of thinking is used, but now the green area lies below the horizontal boundary line. This results in the equations being swapped and slightly different. An additional limit is set to the minimum required length of the second element. In case the length of the first element is exactly equal to the diagonal distance between p1 and p2, the length of the second element may with the current equations be equal to 0. To prevent this, the minimum required length of the second element must always be larger than $0.2 * z_{p1}$. This makes sure that there will always be a second element and the first element does not fall exactly onto the position of the last element of the previous step.

After picking the first two elements, these will be connected to the last top and end node. Their intersection point is found by creating two circles with radii equal to the lengths of the two elements. Using RhinoScriptSyntax, the intersection points of the two circles can be found and the intersection point is chosen with the highest x-coordinate. This will be the node where the two elements meet.

Lastly, a third element is picked with the only length constraint being that it needs to be longer than the minimal distance to the bottom chord. The third element is then positioned using a similar technique as earlier. A circle is created which has a radius equal to the length of the third element and a centre located at the intersection point of the first two elements. The intersection points of the circle and the bottom chord are calculated and the point is picked with the largest x-coordinate. The third element is positioned in between.

Capacity of the element

The second constraint set on the random choice of element is based on the required capacity. When the range in available capacities is large, the resulting unity checks will vary a lot. To narrow down the range in capacities that can be chosen to more a realistic one, an upper and lower bound is set. Like explained before, because the geometry will change every time new elements are picked and therefore also the resulting normalforces, it is impossible to exactly calculate the required capacity of the next to be picked element. By looking at the normal forces calculated in the last iteration and considering a large enough range around these normal forces, still a more considered capacity range can be defined for the next to be chosen elements. The range has to be large enough that elements can be found which comply to all constraints and that the required capacity is not made too specific, as it may differ quite a bit when the geometry is changed. In addition, the range has to be small enough so that still a calculated direction can be given to the choice of element. In the script, a backup option is provided when no option exists in between the defined boundaries. In this backup option, an element is chosen without the capacity constraints applied to it. An illustration of the approach is shown in figure 7.23.

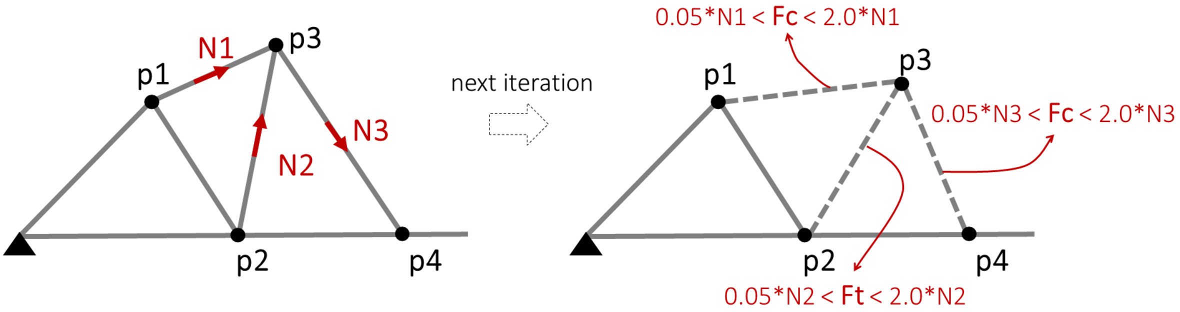


Figure 7.23: Constraint on the required capacity of the next to be chosen element, based on the normalforces calculated in the last iteration

When applying this constraint on the required capacity, the number of iterations needed to find valid options reduces significantly. More options can then be compared and better unity checks can be found.

A small sensitivity study was carried out to decide on the best range to be applied. Every time a different range was applied the number of iterations and the number of found valid options was recorded. The maximum number of iterations was always set on 100 and the maximum number of valid options needed to be found was set on 3. Furthermore, the mean unity check of the best option was saved as well as the number of iterations where the capacity range was too small, causing an element to be chosen without the capacity constraint applied to it. The sensitivity study does have some inaccuracies as the results depend on the geometry of the first triangle of the truss. This first triangle is regenerated every time the algorithm is rerun. In addition, the results also depend on the applied database and the dimensions of the required truss. The results can therefore not be compared in a too strict sense, but they can provide some useful insights. The results are shown in the graph below.

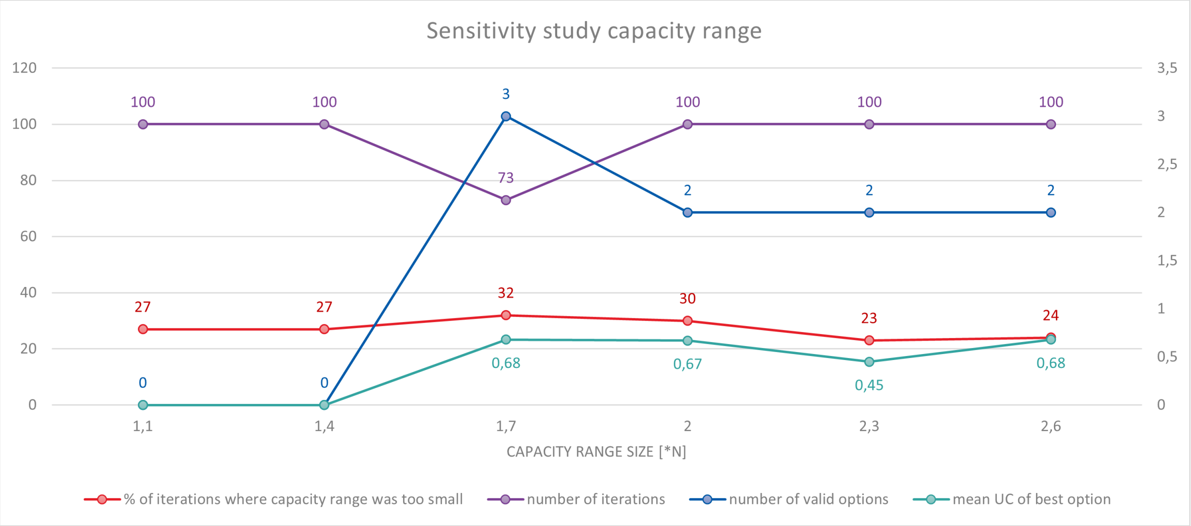


Figure 7.24: Sensitivity study of the capacity constraint range

capacity range	range size
0.4N - 1.5N	1.1
0.3N - 1.7N	1.4
0.2N - 1.9N	1.7
0.1N - 2.1N	2
0.1N - 2.4N	2.3
0.1N - 2.7N	2.6

Table 7.1: The accompanying capacity ranges

In the graph it can be seen that the percentage of iterations where the range was too small for an element to be found, does not increase or decrease significantly when varying the range size. In first instance it would be expected that the number of iterations where no element option can be found becomes smaller when the applied range becomes larger. The percentage of these kind of iterations is thus more influenced by the randomness of the process than the range size. So for the tested database, a lower bound on the required range size does not have to consider the risk of it being too small to find enough element options.

What is influenced by the range size is the possibility of finding a valid triangle option whereby the unity checks lie in between 0.3-1.0. When the range is too small, no valid options are found (see capacity range size 1.1 and 1.4 in the graph). This could mean that the range size is too specific in these cases, so that capacities are chosen which comply with the previous geometry but differ too much from the required capacities of the new geometry. Providing a larger capacity range will make the difference in resulting unity checks larger, but at least the right capacity will lie somewhere in between and can be found with enough iterations. This conclusion is visualized in figure 7.25. In this figure it can be seen that the capacity range taken around the normal force calculated in the first iteration (in red) is too specific. The next iteration has a different geometry and therefore different normalforces (in blue). The required capacity for the first element lies outside of the considered capacity range as it is defined too specific around the normalforce calculated in the last iteration. Therefore, the chosen element will probably not have a unity check inside the allowed limits as it differs too much from the required capacity. The considered capacity range makes it in this case more difficult to find a valid option instead of the other way round.

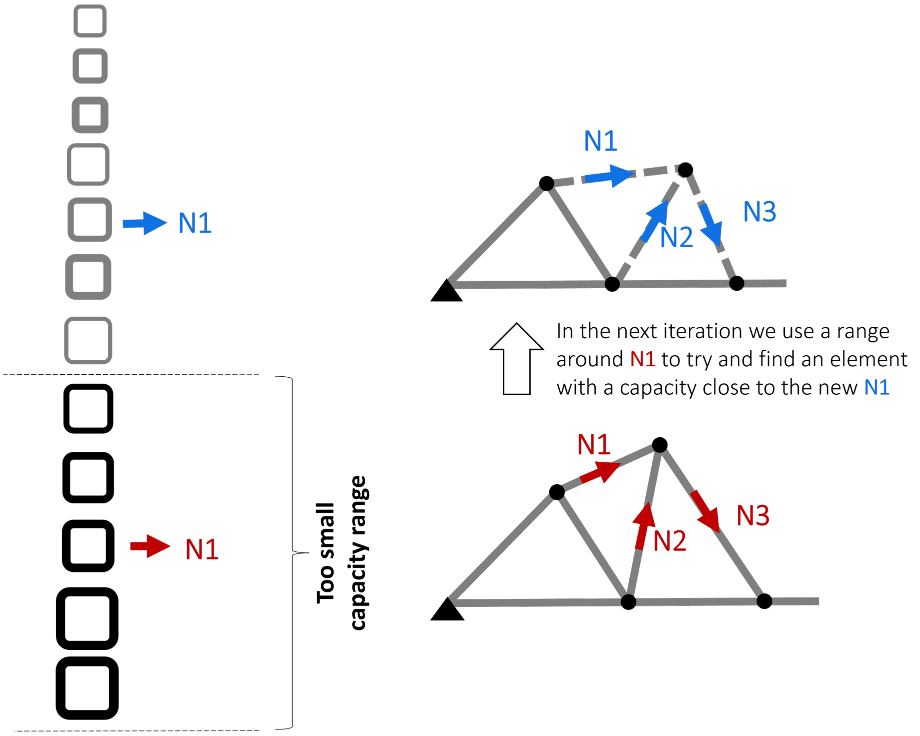

N1
N1
N2
N3
In the next iteration we use a range around N1 to try and find an element with a capacity close to the new N1
N1
N1
N2
N3
Too small capacity range

Figure 7.25: Visualization of the situation where the capacity range is too specific

Lastly, it can be seen that the resulting mean UC when valid triangle options are found (range sizes 1.7 to 2.6) does not increase or decrease significantly when varying the range size. The same applies to the number of iterations needed to find at least 3 valid options, or the number of valid options that are found under 100 iterations. These values thus also rely more on the randomness of the process than the range size.

The conclusion is that the range size has to be large enough so that valid options may be found, but that the exact range size can not be derived with certainty due to the random aspect of the process and the variability of the available stock. In the output file of the generation of the middle triangles, the amount of options in the chosen capacity range for each iteration is reported. This way it may be checked how specific the range is. In addition, the percentages are reported of the amount of UC's that were too small or too large. Depending on these values, the capacity range may be shifted so that the algorithm has the tendency to choose more smaller or more larger profiles. For example in the current stock, the profiles are relatively large. By setting the capacity range from 0-1.8, more small profiles will be included in the considered range, enlarging the chance to find a suitable element for the diagonals that do not transfer large forces.

The considered capacity range is included in the performance parameters made accessible in the input group. This way, the capacity range may be shifted according to output results, so that the algorithm can be adapted to find solutions more efficiently.

Width of the element

The third constraint has to do with the constructability of the connections with the bottom chord. These constraints resulted from literature study about possible connections and their influence on the truss design, summarised in paragraph 3.2.2. The so called second and third elements, which are the diagonals of the truss and connected to the bottom chord, may not have larger widths than the bottom chord elements. The diagonals will need to be able to be welded onto the upper flange of the bottom chord. In case the bottom chord consists of rectangular hollow sections, an extra constraint is applied so that the width of the diagonals is not smaller than half the width of the RHS, to avoid chord face failure.

Not used element

Furthermore, a fourth constraint is set so that only elements may be picked which do not have a '1' in the 'used' column. The database has been updated since the last step and only elements that have not already been positioned are available.

End of the span
Lastly, there is one additional condition that the geometry needs to comply with, if not, the loop is restarted. This condition is that the end node of the third element may not be too close to the end of the bottom chord. If this is the case, it will be too challenging to find elements to fit in the last triangle. The boundary is set on half the distance between points p2 and p4, i.e. the width of the last triangle. If this distance is not available anymore after the last node, the last node is too close to the end of the bottom chord.

Overview
To summarise, the following constraints need to be satisfied when randomly picking the three elements:

- The length of the element needs to be inside the minimum and maximum length range, so that the resulting geometry is applicable.
- The capacity of the element needs to lie inside the capacity range, so that a valid solution with correct capacities can be found more efficiently.
- The width of the element needs to comply with constraints dependent on the profile of the bottom chord so that a connection can be manufactured.
- The element may not already be positioned in the truss design.
- The choice of three elements may not result in a geometry that comes too close to the end of the span, so that a last triangle can still be fitted in.

7.7.3. Calculating the axial forces

Now that the geometry is found, the forces may be calculated using the method of section.

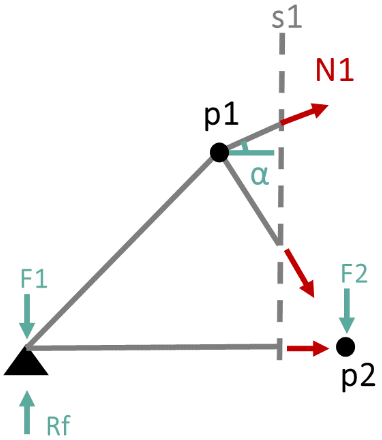


Figure 7.26: First section

The first section (s1) is positioned halfway between points p1 and p2 (see figure 7.26). Moment equilibrium is calculated around p2 to determine the force in the first element. Because this algorithm applies to all the elements placed between the first and last triangle, a general method to calculate the moments in this moment equilibrium resulting from the pointloads and reaction force is provided as follows:

$$M_1 = R_f * x_{p2} + \sum_{i=1}^{n} F_i * (x_{p2} - x_{F_i}) \tag{7.8}$$

with n = the number of point loads, situated before p2. This is the reason a list with already calculated pointloads and their positions are updated through out the anemone loops. The moment equilibrium can then be written as follows:

$$\Delta x = x_{p2} - x_{s1}$$

$$\alpha = tan^{-1}(\frac{z_{p3} - z_{p1}}{x_{p3} - x_{p1}})$$

$$\sum M_{p2} = 0 = M_1 + N_1 sin(\alpha)\Delta x + N_1 cos(\alpha)(z_{p1} + \frac{\Delta x}{x_{p3} - x_{p1}}(z_{p3} - z_{p1}))$$

$$\Rightarrow N_1 = \frac{-M_1}{sin(\alpha) * \Delta x + cos(\alpha) * (z_p 1 + \frac{\Delta x}{x_{p3} - x_{p1}} * (z_{p3} - z_{p1})} \quad (7.9)$$

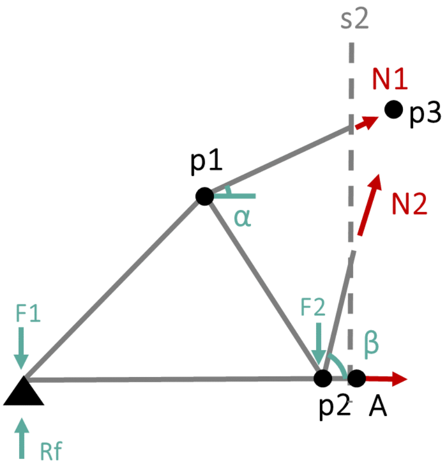


Figure 7.27: Second section

The second section (s2) is positioned halfway between points p2 and p3. Moment equilibrium is calculated around point A (see figure 7.27) to determine the force in the second element. The moment due to the reaction force and pointloads up to point A now includes an additional pointload, situated at point p2, calculated with:

$$F_{new} = q * \frac{x_{p4} - x_{F,last}}{2}$$

$$x_{F,new} = x_{p2}$$

These two values are added to the list of previous pointloads and point load positions so that the new moment can be calculated:

$$M_2 = R_f * x_A + \sum_{i=1}^{n} F_i * (x_A - x_{F_i}) \quad (7.10)$$

with n = the number of point loads, situated before point A. The moment equilibrium can then be written as follows:

$$\beta = tan^{-1}(\frac{z_{p3}}{x_{p3} - x_{p2}})$$

$$\sum M_A = 0 = M_2 + N_1 cos(\alpha)[z_{p1} + \frac{x_A - x_{p1}}{x_{p3} - x_{p1}}(z_{p3} - z_{p1})] + N_2 sin(\beta)(x_A - x_{p2})$$

$$\Rightarrow N_2 = \frac{-M_2 - N_1 cos(\alpha)[z_{p1} + \frac{x_A - x_{p1}}{x_{p3} - x_{p1}}(z_{p3} - z_{p1})]}{sin(\beta)(x_A - x_{p2})} \quad (7.11)$$

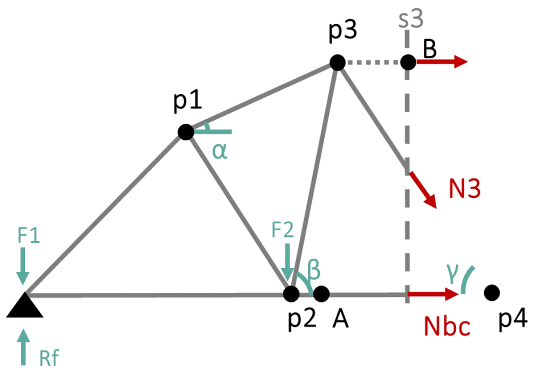


Figure 7.28: Third section

Finally the axial force in the last element is calculated. Like for the first triangle, a virtual horizontal top chord element is assumed so that a section can be made through this last element. The section is made in between p3 and p4. First moment equilibrium is calculated around p3 to determine the axial force in the bottom chord. Then, moment equilibrium is calculated around point B to calculate the force in the third element.

Calculations for moment equilibrium around p3:

$$M_3 = R_f * x_{p3} + \sum_{i=1}^{n} F_i * (x_{p3} - x_{F_i}) \tag{7.12}$$

with n = the number of point loads, situated before point p3.

$$\gamma = tan^{-1}(\frac{z_{p3}}{x_{p4} - x_{p3}})$$

$$\sum M_{p3} = 0 = M_3 - N_{bc} * z_{p3}$$

$$\Rightarrow N_{bc} = \frac{M_3}{z_{p3}} \tag{7.13}$$

Calculations for moment equilibrium around point B:

$$M_4 = R_f * x_B + \sum_{i=1}^{n} F_i * (x_B - x_{F_i}) \tag{7.14}$$

with n = the number of point loads, situated before point B.

$$\sum M_B = 0 = M_4 - N_{bc} * z_{p3} - N_3 sin(\gamma)(x_B - x_{p3})$$

$$\Rightarrow N_3 = \frac{M_4 - N_{bc} * z_{p3}}{sin(\gamma)(x_B - x_{p3})} \tag{7.15}$$

7.7.4. Finding the best option

Now all three axial forces have been found (N_1, N_2, and N_3), they can be tested against their capacities. The capacity which is included in the Unity check has the same direction as the force in the element, i.e. in tension or compression. If all three Unity Checks are above a specified lower UC limit and below 0.9, the configuration is considered as a valid option. Again, the best option out of a list of valid options is chosen.

The output of this step is once more an updated reclaimed element and used element database. The updated databases are stored under the same pickle file name so that when reopening the file at the beginning of the python component in the next anemone loop, the new information will be included. Also lists of the used profiles and the line geometries of the elements are made so that they can be transferred to Karamba to visualize the intermediate results. For the next step, an updated new top node and end node is exported as well as the updated lists including information about the pointloads.

Lastly, the mean width of the generated triangles is computed. This value as well as the x coordinate of the last end node are connected to the break function of the anemone loop. With these values, it can be calculated whether the last generated triangle is in the critical range of the end of the bottom chord. This critical range is set as 1.5 times the mean width of the generated triangles. Entering this critical range means that the anemone loop will stop and the algorithm will go onto the next step: the generation of the last triangle.

7.8. Choosing the last three elements

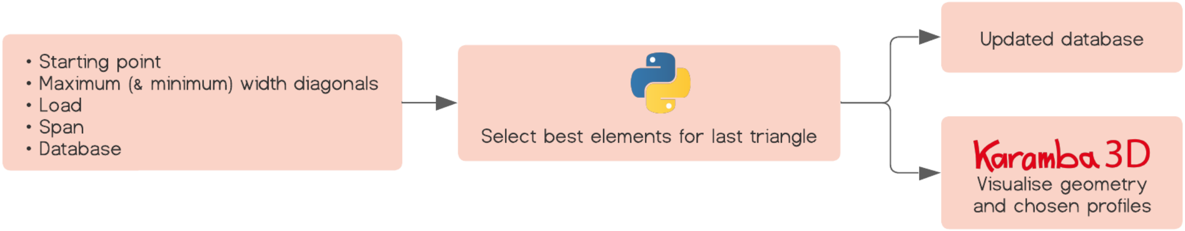

• Starting point
• Maximum (& minimum) width diagonals
• Load
• Span
• Database
Select best elements for last triangle
Updated database
Karamba 3D
Visualise geometry and chosen profiles

Figure 7.29: Grasshopper steps

The choice of the last three elements is quite similar to the choice of the middle elements. As described in 7.5, the main difference lies in the choice of the third element, which needs to fit exactly in between the last top node and the end of the bottom chord. The changes in the python script are mentioned below.

7.8.1. Generating the geometry

For the first element, an additional constraint on its length is set, this time about the maximum length it may have. Its length may not be longer than 0.9 times the horizontal distance left between the last top node and the end of the span. This way, the last triangle will not pass the end of the span, while there is still space for the last diagonal to be situated between the last top node and the end of the bottom chord

Another difference to the script for the middle elements is that no third element is picked until its required capacity is calculated. This is because the last element will need to fit in between the last top node and the end of the bottom chord, so an element will need to be cut to this length. Since an element has to be cut anyway, it is better to reason in reverse for choice of the last element: we first calculate the normal force that will be present in this last member and then find an element from the database with a capacity closest to this value. This element is then chosen and cut to the correct length. The spare part is included in the database.

7.8.2. Calculating the axial forces

The axial forces in the first two elements are calculated in the same manner as for the middle elements. The axial force in the third and last element is calculated performing a simple node equilibrium calculation, like was done for the very first element in the truss. The difference with the third elements in the rest of the truss is that no assumed top chord element needs to be used as there will not be any present. The calculation of the force in the third element can be written as follows:

$$\begin{aligned} F_{last} &= q * \frac{x_{p4} - x_{p2}}{2} \\ \sum V = 0 &= N_{3,V} + F_{last} + R_f \\ N_{3,V} &= N_3 * sin(\gamma) \\ \Rightarrow N_3 &= \frac{-R_f - F_{last}}{sin(\gamma)} \end{aligned} \tag{7.16}$$

7.8.3. Finding the best option

Here, the same steps are once more followed as explained in the previous sections.

The last output consists of the final updated reclaimed and used element databases, stored in a final pickle file. But also the final lists with profile names and line geometries to transfer to Karamba.

7.9. Generating a 3D model and visualizing and calculating the model in Karamba

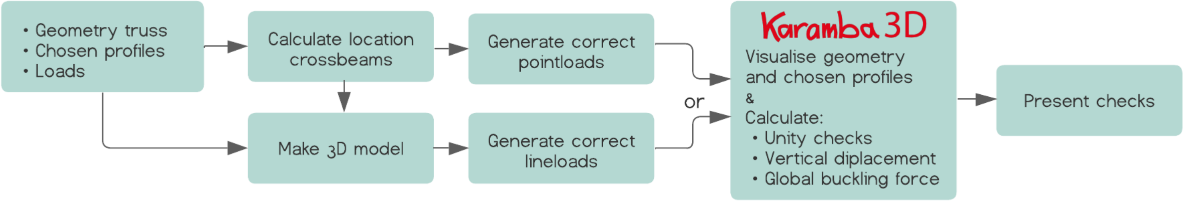


Figure 7.30: Grasshopper steps

2D model

In between iterations, the 2D truss is visualized using Karamba components. By importing the profile names and the line geometries, Karamba can visualize the truss with according profiles. Also the location of the supports are defined. The joints between the diagonals are defined as hinges. An intermediate step that has to be made is the definition of intersection points along the bottom chord, where diagonals and extra crossbeams are connected. Otherwise, the connections will not be acknowledged in Karamba and the bottom chord will respond separately to the applied loads. The positions of the cross beams are calculated so that one is positioned at every diagonal connection and additional beams are located in between these nodes when the width of the triangle is larger than 3.5 meters. Point loads representing the traffic load and the weight of the bridgedeck and crossbeams, including load factors, are located on the bottom chord at every crossbeam position. An example of a truss model in Karamba is shown in figure 7.31.

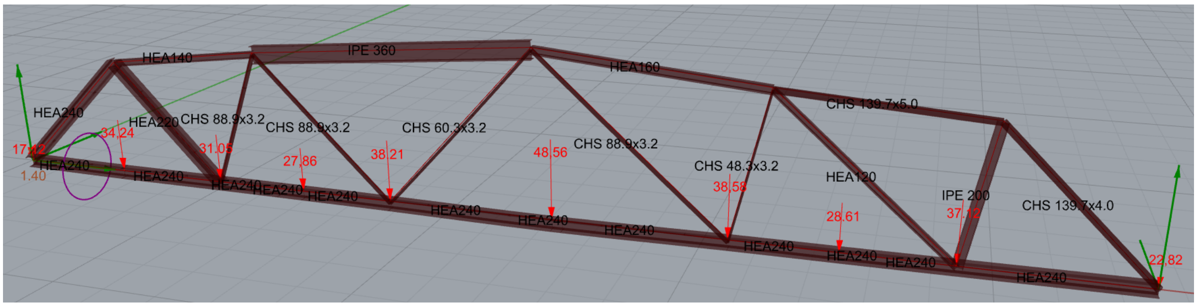


Figure 7.31: Truss model

3D model

After the full truss has been generated, a temporary 3D model of the truss bridge can be made by copying the generated truss and moving it over the defined width of the bridge, like explained in 6.3. The purpose of generating this temporary model is to perform the global checks on the generated truss design more accurately, as the effect of the bridge deck stiffness can be included in this case. In the general input of the whole design process, the weight of the assumed bridge deck is specified. With this value, required crossbeams are calculated. Paragraph 8.1 will elaborate on these input values. In the Karamba calculation of the 3D model, the selfweight of the cross beams will be included so that the distributed line loads on the crossbeams will only represent the weight of the deck plate and the variable traffic load, including load factors.

After the truss has been copied, the geometry of the cross beams have been specified and the additional supports are defined, the Karamba component which assembles the whole model will receive this new information and change the 2D model to the 3D one. The 3D model of the truss presented in figure 7.31 is presented in figure 7.32.

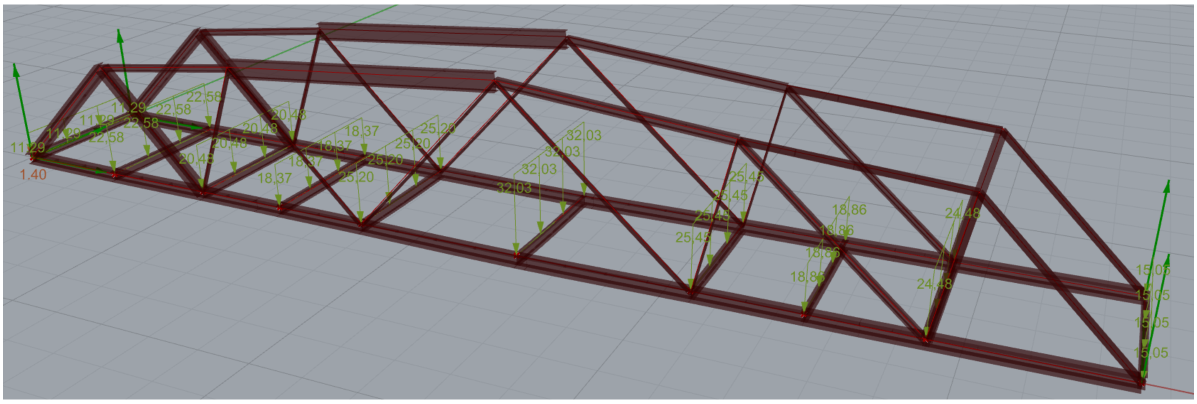


Figure 7.32: Temporary truss bridge model

Subsequently, two analyses are performed: a first order analysis and a second order analysis including calculation of the buckling modes and corresponding buckling factors. By performing these analyses we can check the truss bridge design on the following aspects (see figure 7.33):

- Whether all the elements have enough capacity to resist the applied forces

  check: no UC's above 1

- Whether the maximum allowed vertical displacement for footbridges is not exceeded

  check: $\frac{max\ displacement}{\frac{span}{8}} \leq 1$

- Whether no global instability will occur

  check: the smallest buckling factor should be larger than 1

In figure 7.33, these checks are shown. The components are turned green when the check is satisfied and red when not. It can be seen that the current 3D truss bridge model that is analysed does not fulfill global buckling requirements. The model is unstable for the current loads.

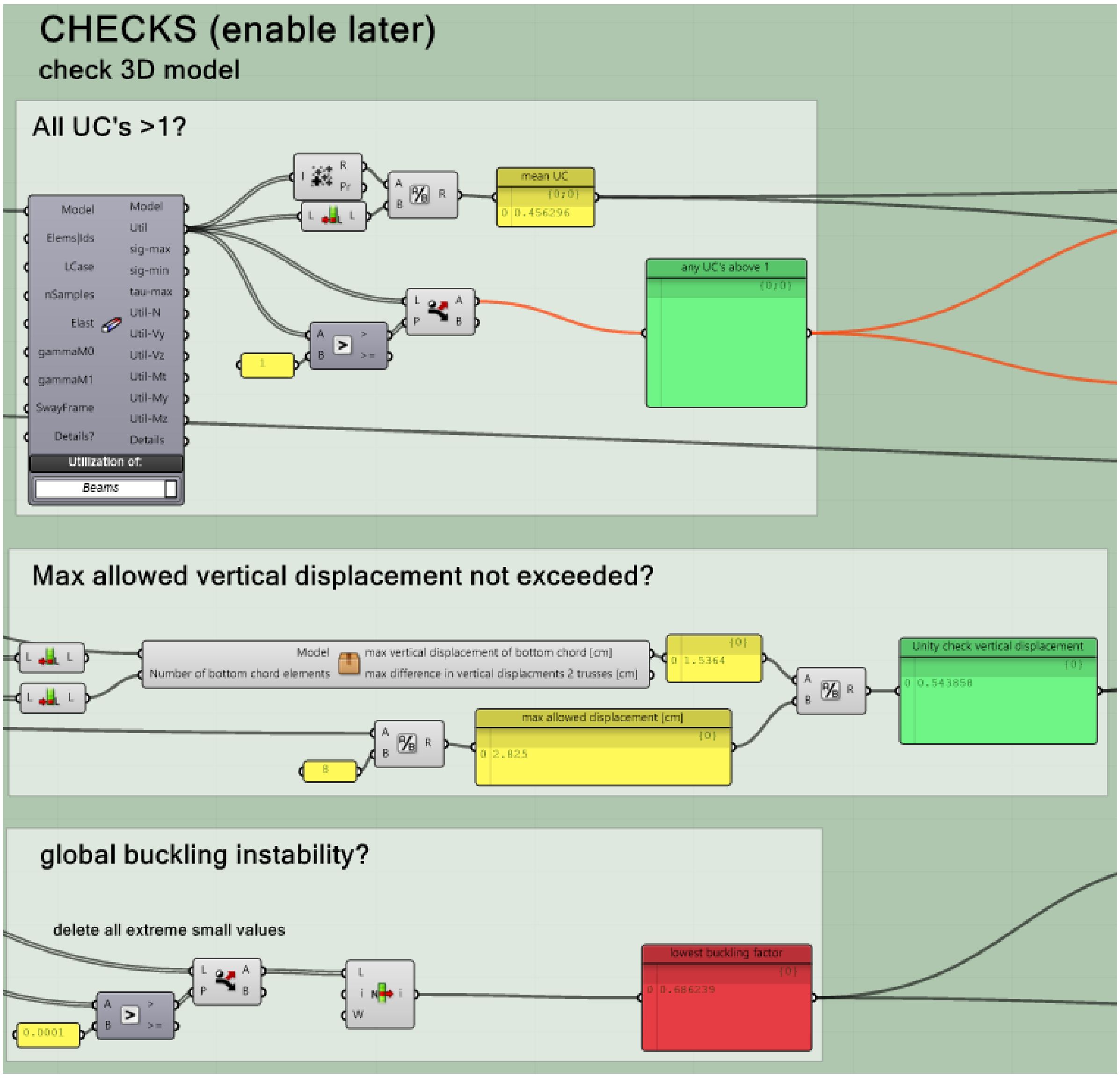


Figure 7.33: Grasshopper components

A full run through of the design process up to this step is shown in appendix B.

7.10. Choosing the best truss bridge design out of a solution cloud

7.10.1. Truss solution cloud

When all of the above checks are satisfied, the truss design can be seen as a valid option. By rerunning the whole process multiple times, several valid options may be obtained. These will all have different configurations due to the random aspects in the design process. They will therefore also have other characteristics. From these valid options, the truss may be selected with:

- The lowest environmental impact, the main goal of this thesis
- The lowest mass
- The highest capacity utilisation
- The least number of elements
- No angles between welded hollow sections lower than 30 °

The steps performed in Grasshopper are shown in figure 7.34, with a picture of the required actions in Grasshopper shown in 7.35.

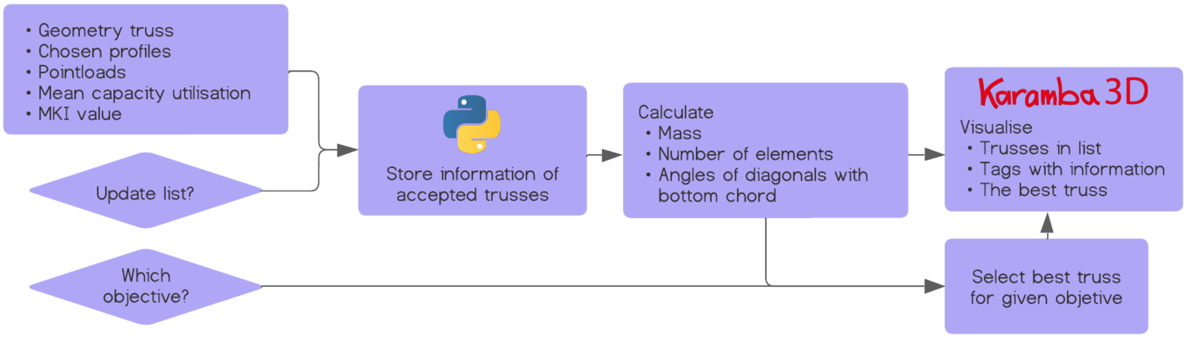


Figure 7.34: Grasshopper steps

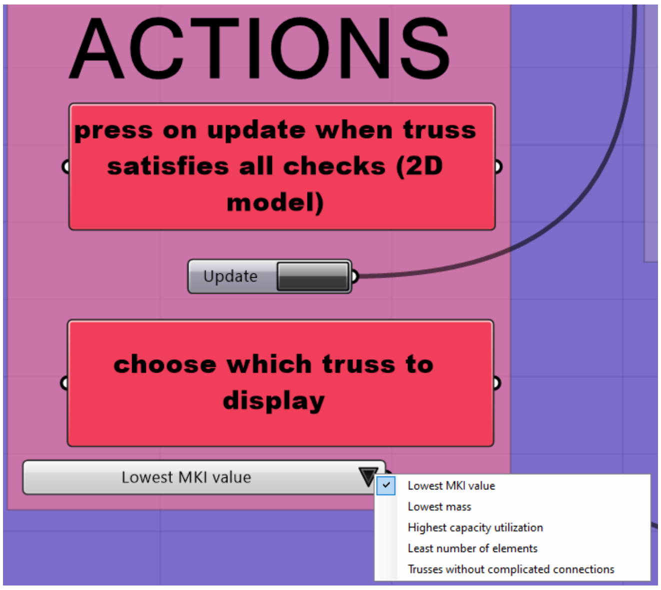

ACTIONS
press on update when truss satisfies all checks (2D model)
Update
choose which truss to display
Lowest MKI value
Lowest MKI value
Lowest mass
Highest capacity utilization
Least number of elements
Trusses without complicated connections

Figure 7.35: Update button

When a truss satisfies all required checks, the design may be stored. This is done in a Python component whereby a toggle input button ('update') defines whether the input variables are stored as persistent variables, shown in the output. This way, the data is kept when the algorithm reruns. This component will be called the 'storing component'. Every time a valid truss option comes up, the update button may be clicked and the storing component will add its information to the list of already stored data.

The output of the storing component includes all necessary data from the valid truss options so that the trusses may be visualized and compared among each other. The data is stored in data trees, among which their geometry, profiles, mean UC value and MKI value (see 9.2). From this data also the total mass of the trusses, the number of elements and the angles between welded hollow sections are calculated. Now all information is listed so that the best truss option can be defined according to each of the comparison goals listed above.

The trusses are first visualized alongside each other in the Rhino viewport. They are visualized next to the truss that is currently being analysed. When this truss satisfies all checks, it is added to the valid truss options column. Their characteristics are displayed next to them. Depending on which comparison goal is chosen, which can be picked in the dropdown menu in the 'ACTIONS' group, the best truss will be shown in black, while all other trusses are coloured blue. An example is shown in figure 7.36.

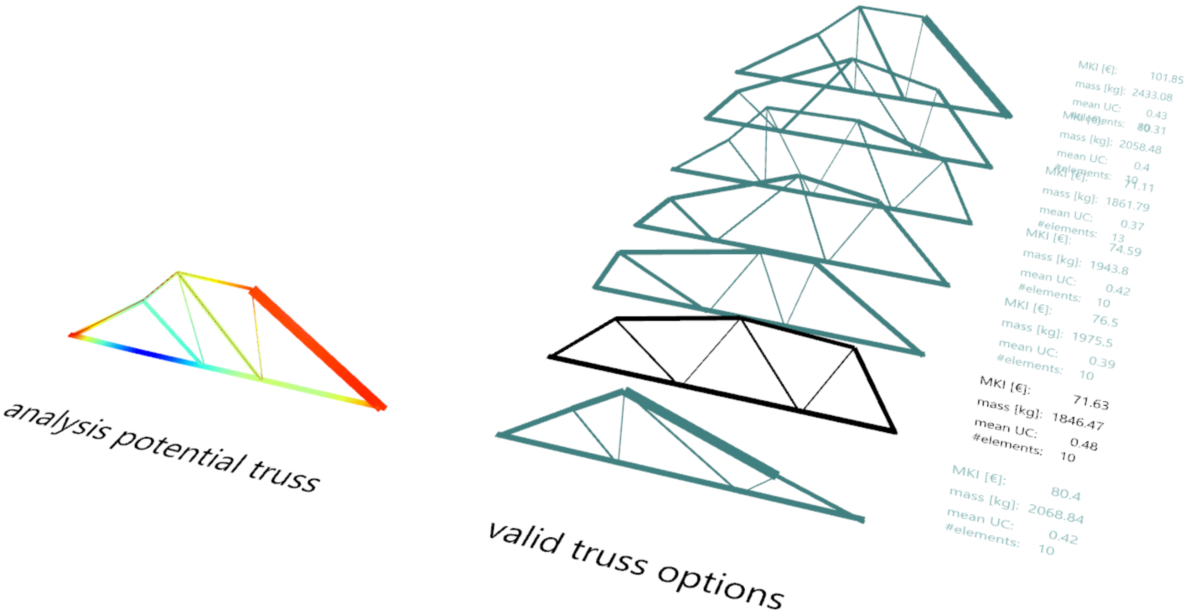
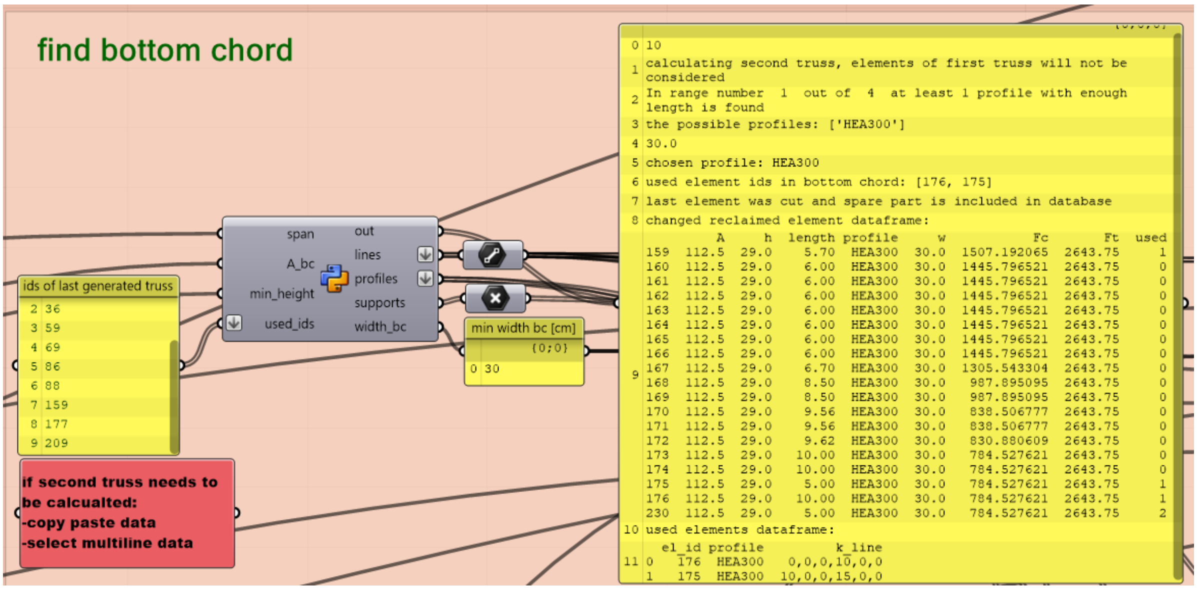


Figure 7.36: Truss solution cloud

7.10.2. Truss bridge solution cloud

To generate a full truss bridge design, two trusses need to be generated whereby no reclaimed steel element may be present in both designs. This is done by first generating the first truss, storing it if satisfying all checks and with it, storing the element id's of the used elements. This list of element id's can subsequently be used as input for the very first python component in the algorithm: generating the bottom chord. At the start of the Python script, all used elements will get a '1'in the 'used' column, so that they can not be picked in the rest of the algorithm (see figure 7.37). A second truss can then be generated, while making sure that it will not include any of the already picked elements of the last valid truss option.



ids of last generated truss

```
2 36
3 59
4 69
5 86
6 88
7 159
8 177
9 209
```

if second truss needs to be calcualted:
-copy paste data
-select multiline data

```
 0 10
 1 calculating second truss, elements of first truss will not be
   considered
 2 In range number  1  out of  4  at least 1 profile with enough
   length is found
 3 the possible profiles: ['HEA300']
 4 30.0
 5 chosen profile: HEA300
 6 used element ids in bottom chord: [176, 175]
 7 last element was cut and spare part is included in database
 8 changed reclaimed element dataframe:
            A     h  length profile     w           Fc       Ft  used
   159  112.5  29.0    5.70  HEA300  30.0  1507.192065  2643.75     1
   160  112.5  29.0    6.00  HEA300  30.0  1445.796521  2643.75     0
   161  112.5  29.0    6.00  HEA300  30.0  1445.796521  2643.75     0
   162  112.5  29.0    6.00  HEA300  30.0  1445.796521  2643.75     0
   163  112.5  29.0    6.00  HEA300  30.0  1445.796521  2643.75     0
   164  112.5  29.0    6.00  HEA300  30.0  1445.796521  2643.75     0
   165  112.5  29.0    6.00  HEA300  30.0  1445.796521  2643.75     0
   166  112.5  29.0    6.00  HEA300  30.0  1445.796521  2643.75     0
 9 167  112.5  29.0    6.70  HEA300  30.0  1305.543304  2643.75     0
   168  112.5  29.0    8.50  HEA300  30.0   987.895095  2643.75     0
   169  112.5  29.0    8.50  HEA300  30.0   987.895095  2643.75     0
   170  112.5  29.0    9.56  HEA300  30.0   838.506777  2643.75     0
   171  112.5  29.0    9.56  HEA300  30.0   838.506777  2643.75     0
   172  112.5  29.0    9.62  HEA300  30.0   830.880609  2643.75     0
   173  112.5  29.0   10.00  HEA300  30.0   784.527621  2643.75     0
   174  112.5  29.0   10.00  HEA300  30.0   784.527621  2643.75     0
   175  112.5  29.0    5.00  HEA300  30.0   784.527621  2643.75     1
   176  112.5  29.0   10.00  HEA300  30.0   784.527621  2643.75     1
   230  112.5  29.0    5.00  HEA300  30.0   784.527621  2643.75     2
10 used elements dataframe:
     el_id profile          k_line
11 0   176  HEA300   0,0,0,10,0,0
   1   175  HEA300  10,0,0,15,0,0
```

Figure 7.37: Used ids

In the figure above, the list of used id's can be seen as input, which will be empty when generating the first truss and including id numbers when generating the second truss. In the beginning of the yellow output file it is stated which type of truss is being calculated.

The reason for this step to be manual is the fact that despite producing persistent variables in the storing component, grasshopper will still recognize the id list as a recursive data stream and will show an error. On the other hand, a controlled, manual step is also desirable in this part of the process as there will be more control over firstly, which truss to choose to form a truss bridge with and secondly, whether you want the algorithm to generate a new truss or a 'second' truss.

After the two trusses are generated, the truss bridge model may be created. The main steps performed in this blue group of Grasshopper components are visualised in the flowchart in figure 7.38.

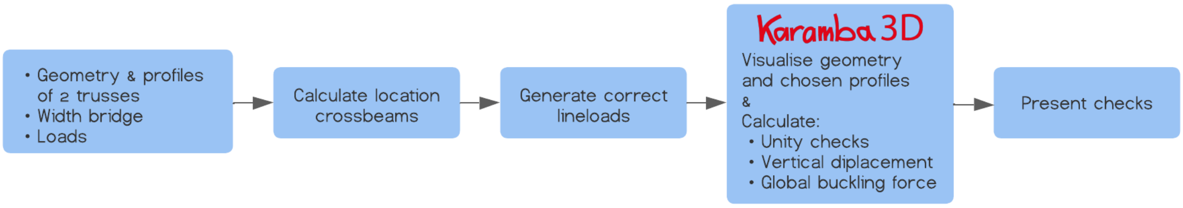

• Geometry & profiles of 2 trusses
• Width bridge
• Loads
Calculate location crossbeams
Generate correct lineloads
Karamba3D
Visualise geometry and chosen profiles
&
Calculate:
• Unity checks
• Vertical diplacement
• Global buckling force
Present checks

Figure 7.38: Grasshopper steps

As a default, every time two trusses are added to the list of valid truss options (assuming the second truss will not contain any elements included in the first truss), grasshopper will join these together to a 3D model of a truss bridge design. Crossbeams will be added between the trusses. The location of the crossbeams are chosen by listing all x-coordinates of diagonal connections to the bottom chord of the two trusses. These x-coordinates are sorted and subsequently from start to end, a python component will eliminate x-coordinates when they are located within 1 meter of the previous crossbeam and add x-coordinates halfway when an x-coordinate is distanced more than 3.5 meters from the previous crossbeam. This way, crossbeams will be located as much as possible at diagonal connections but without lying to close or to far from each other. Examples of resulting truss bridge designs can be seen in figure 7.40.

Subsequently, the same structural mechanical checks may be performed as before. An additional check that is included is the maximum difference in vertical displacements in the bridge deck. As two trusses with different geometries will support the bridge deck, the vertical displacements of these trusses under the same load will be different due to varying stiffnesses. To prevent uncomfortable displacements for the pedestrians and cyclists, this difference in displacement must be minimal. The difference in displacement will be expressed in the resulting angle of rotation, to also consider the bridge deck width. As no Eurocode rules exist on the maximum allowed rotation, the angle will be presented without a corresponding unity check. This step is thus more of an informative one, so that the user may consider this characteristic before adding the truss bridge design to the solution cloud.

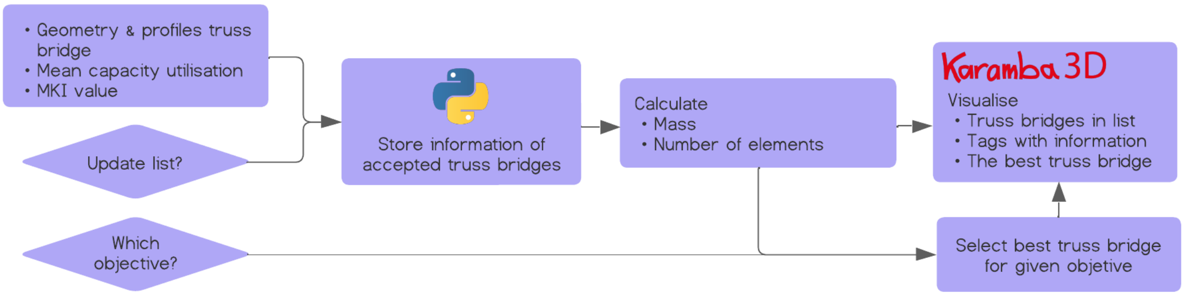


Figure 7.39: Grasshopper steps

Furthermore, the same steps may be repeated as for the analysis and comparison of valid truss options. These are visualised in the flowchart in figure 7.39. Once the truss bridge model is checked, one may manually click the update button of a second storing component, which will now start storing a list of valid truss bridge design options. These options will again be visualized alongside each other in the Rhino viewport with the same characteristics calculated as for the truss designs. A same dropdown menu is provided so that the desired comparison goal is selected and the corresponding best truss bridge design is visualized in black. The result in Rhino is shown in figure 7.40.

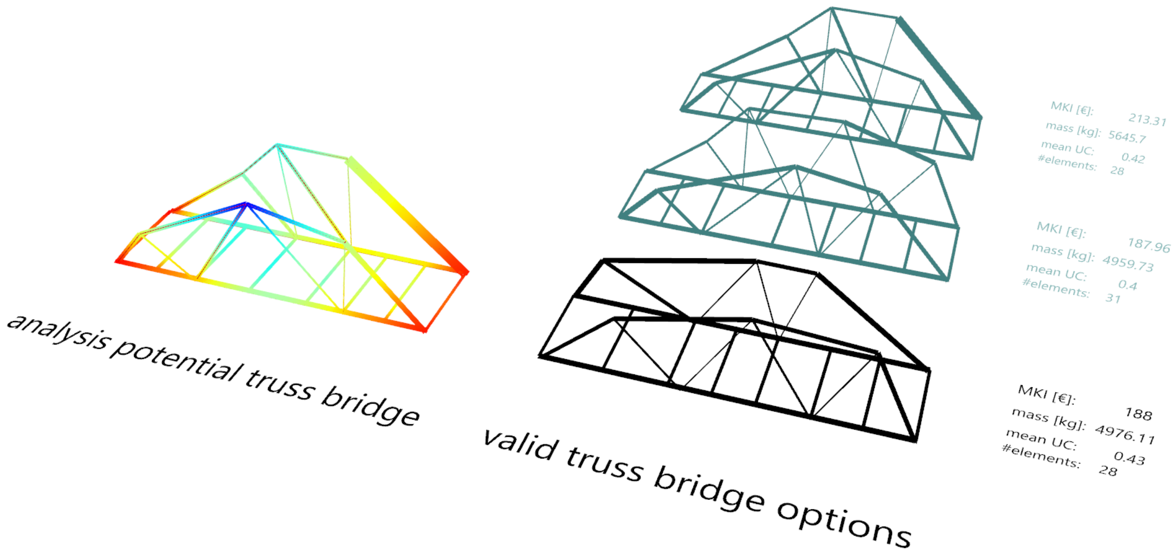


Figure 7.40: Truss bridge solution cloud

7.11. Automatic and manual part algorithm

The end of the design process has now been reached. To recap, the design process starts off by defining the desired width and length of a truss bridge design and uploading a reclaimed stock database. The growth method algorithm will automatically start calculating and growing a locally optimal truss design. The truss generation is a random process so that every time the algorithm is restarted, a different truss design will be generated.

After the generation of the truss, global structural mechanical aspects will be automatically checked using Karamba. When the truss is deemed sufficient, it may be manually added to a list of valid truss options. The second manual step is defining which goal is desired. Depending on the goal, the best truss out of the list of valid truss options is pointed out.

The same steps are repeated to generate full truss bridge designs including two unique trusses and to compare these in the same way.

To conclude, the design process has an automatic part, which will generate a solutioncloud of different but local optimal designs, and a manual part, where preferences of the user are considered. This way, the tool provides enough freedom to comply with the different ways a 'best' design can be chosen. In addition, the two solutionclouds provide the possibility of using the design tool for designing either trusses or full truss bridges.

An example of the full design process carried out is shown in figure 7.41.

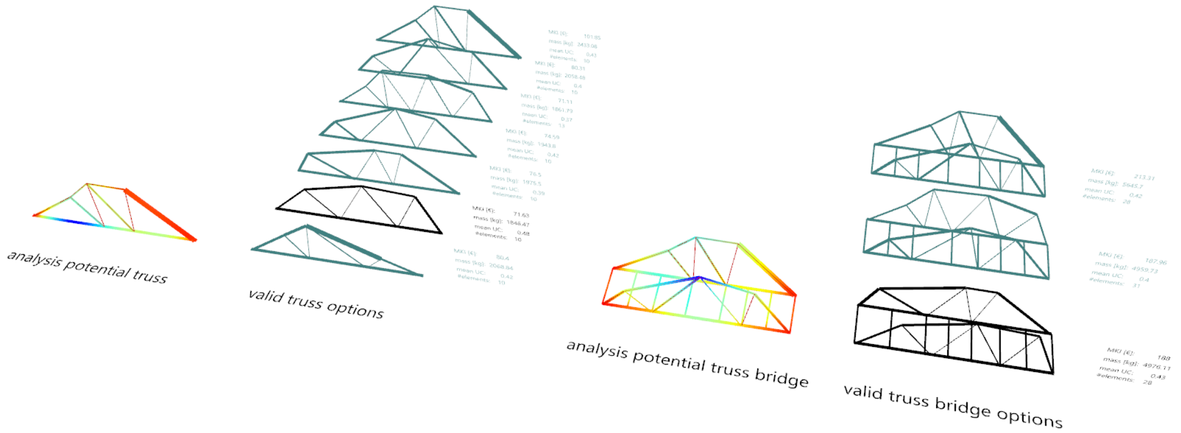


Figure 7.41: Full design process

8
Input, design example and performance

8.1. Input

Next to the width and length of the bridge, a stock of elements needs to be specified. The stock used for the examples in this thesis is explained in 8.1.1. Furthermore, a choice is made for the considered bridgedeck. This choice is explained in 8.1.2.

8.1.1. Reclaimed element database

To make a representative database of reclaimed elements, research was done on the availability of steel elements from steel traders in the Netherlands. Some of these steel traders present their available steel profiles online. The names of these steel traders are:

- ’Omgekeerd bouwen’, by Kamphuis
- ’gebruiktebouwmaterialen.com’, as a part of A van Liempd Sloopbedrijven
- ’tweedehandsmaterialen.nl’, by Snellen

A database of elements has been made including all the steel profiles available online on a certain day. This resulted in a stock of almost 50 tonnes of steel. This is a conservatively assumed stock compared to the actual available amount of reusable steel at this time. Interesting to acknowledge is that most reclaimed steel elements are open sections. The steel traders do have some hollow sections but definitely a minority. The elements have been summarised in an excel sheet whereby their information is sorted into three columns including: the name of the profile, the length of the element and the number of available elements. This excel sheet is imported into the Grasshopper file.

8.1.2. Bridge deck

As inputs, the required span and width of the bridge need to be defined. In addition, a bridge deck needs to be assumed. It is chosen to consider a steel deckplate with stiffeners, supported by crossbeams. This is the most common type of bridge deck. When choosing another type of deckplate, the new weight can easily be adapted in the input stage. When a bridgedeck with a different type of load transfer is chosen, the design process will need to be adapted to consider these changes.

For the design process, representative values of the weight of the bridge deck and the stiffness of the crossbeams are needed to correctly calculate the required trusses. More specifically, the weight of the deck plate and the crossbeams is needed to calculate the design load that the trusses will need to carry. On the other hand, the profile of the crossbeams needs to be known to correctly model the stiffness of the bridge deck, which will counteract global lateral buckling.

To calculate the weight of a steel deckplate, a design example is used: the bicycle path part of the renovated Van Brienenoord bridge. It has a stiff steel bridge deck which may also serve as a representative bridge deck for an open truss bridge. The steel deck plate has a thickness of 10 mm. Stiffeners are placed every 200 mm. The height of the stiffeners is 145 mm and they have a thickness of 12 mm.

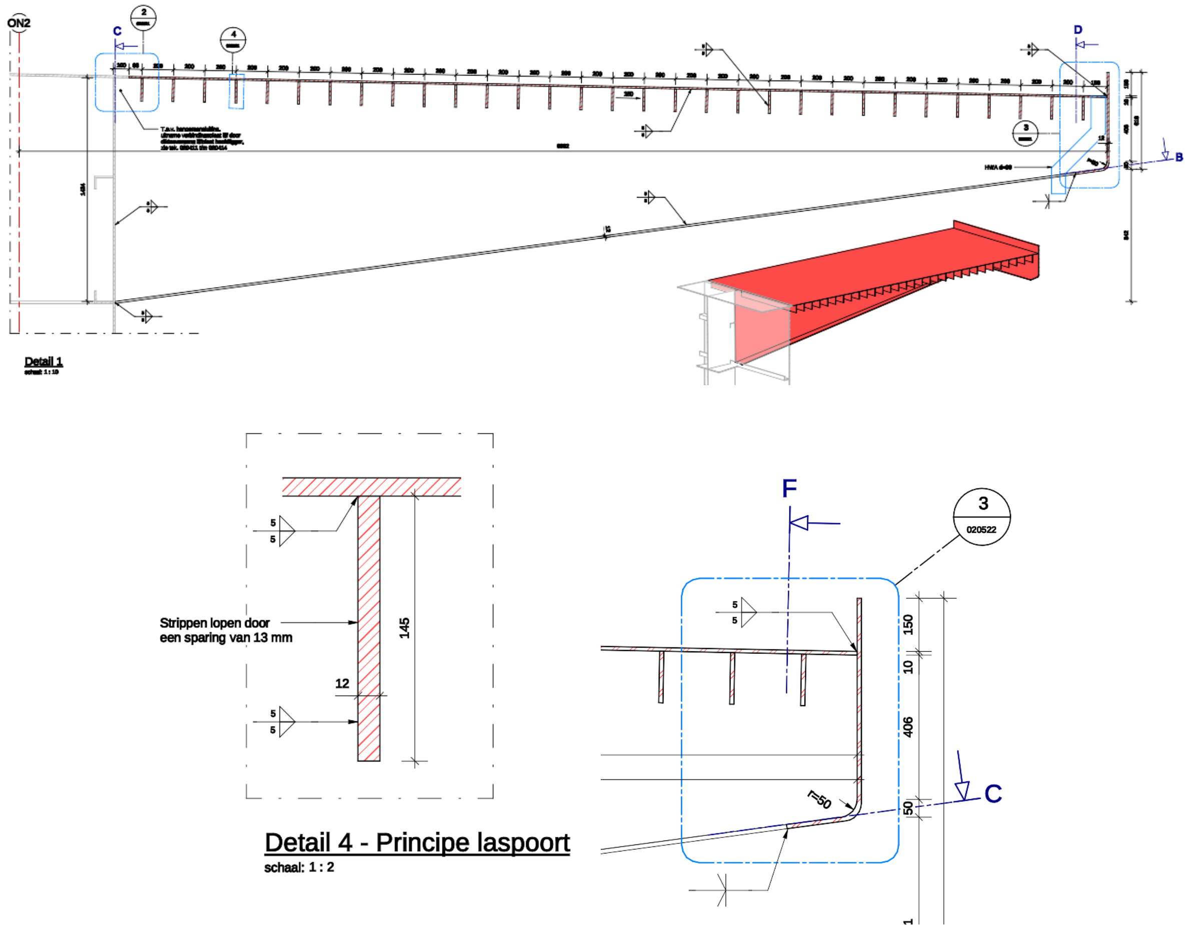


Figure 8.1: Renovated Van Brienenoord bridge, bicycle path deck

These dimensions result in the following weight per m^2:

Cross sectional area deck plate every 200 mm:

$$(12 * 145) + (200 * 10) = 3740mm^2/200mm$$

Cross sectional area deck plate per meter:

$$3740 * 5 = 18700mm^2/m$$

Volume of deck plate per square meter:

$$18700 * 1000 = 0.0187m^3/m^2$$

Weight of deck plate per square meter:

$$0.0187 * 7850 = 146.80kg/m^2$$

$$146.80 * 9.81 = 1440N/m^2 = 1.4kN/m^2$$

The cross beams supporting the deck plate are placed every 3.66 meters. This means that the strength of the deck plate is sufficient for spans up to 3.66 meters. In general, larger spans are not recommended. In the algorithm, cross beams will be located at every intersection point of the diagonals of the truss with the bottom chord. In addition, the decision has been made to include an additional cross beam halfway when the distance between the intersection points becomes larger than 3.5 meters. For the generation of the truss, an initially guessed line load is needed whereby the number of crossbeams, and subsequently their weight, needs to be assumed. This is done by considering a spacing of 3.5 meters. The weight of the deck plate is kept conservatively on $1.4kN/m^2$ as the distance between the crossbeams will vary in the final design, with a large chance that a distance of 3.5 meters will be present somewhere. The final deck plate will be based on the largest distance it has to span.

With the assumed deck plate weight and the boundary condition of a maximum distance of 3.5 meters, the required profile of the cross beam may be calculated. Next to these two parameters, the required profile will depend on the width of the bridge, which is a variable parameter that is defined in the input of the design process. By modelling a simply supported beam with a length equal to the width of the bridge, carrying the weight of the steel deck plate and the traffic load over a normative supporting span of 3.5 meters, Karamba can calculate the required profile of the crossbeam.

For the hand calculations in the growth process, a design load will be used consisting of the weight of the steel deck plate, the cross beams (3.5 meters apart) and the traffic load, with corresponding load factors (of consequence class CC3), see figure 8.2.

For the final Karamba calculation, the weight of the crossbeams is left out as the self weight of the modelled elements will be included in the calculation, see figure 8.2. In the 3D model the crossbeams are modelled with the calculated profile.

The steps are summarized in figure 8.2.

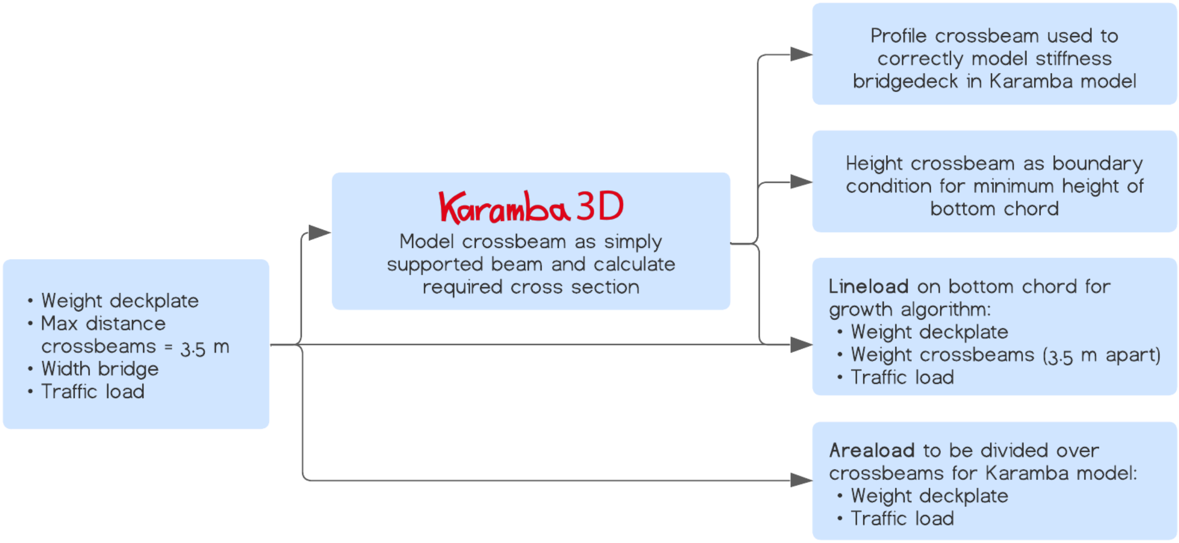

• Weight deckplate
• Max distance crossbeams = 3.5 m
• Width bridge
• Traffic load
Karamba 3D
Model crossbeam as simply supported beam and calculate required cross section
Profile crossbeam used to correctly model stiffness bridgedeck in Karamba model
Height crossbeam as boundary condition for minimum height of bottom chord
Lineload on bottom chord for growth algorithm:
• Weight deckplate
• Weight crossbeams (3.5 m apart)
• Traffic load
Areaload to be divided over crossbeams for Karamba model:
• Weight deckplate
• Traffic load

Figure 8.2: Grasshopper steps

8.2. Design example

An example of a truss bridge model that could be produced by the developed design process is shown in figure 8.3. The input parameters discussed above have been used. The selected elements from the stock of reclaimed elements is visualised in figure 8.4. Table 8.1 presents the characteristics of the presented design.

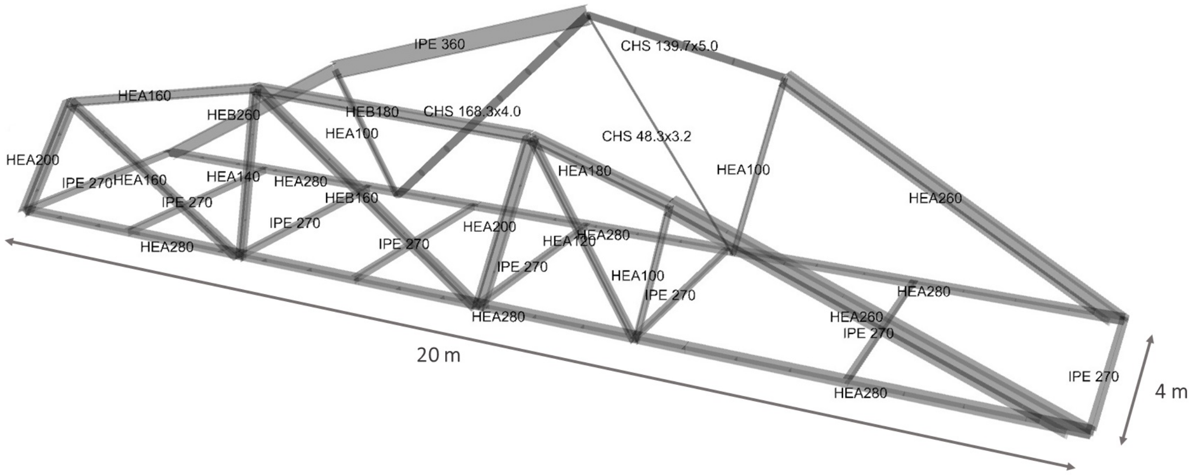


Figure 8.3: Truss bridge design example

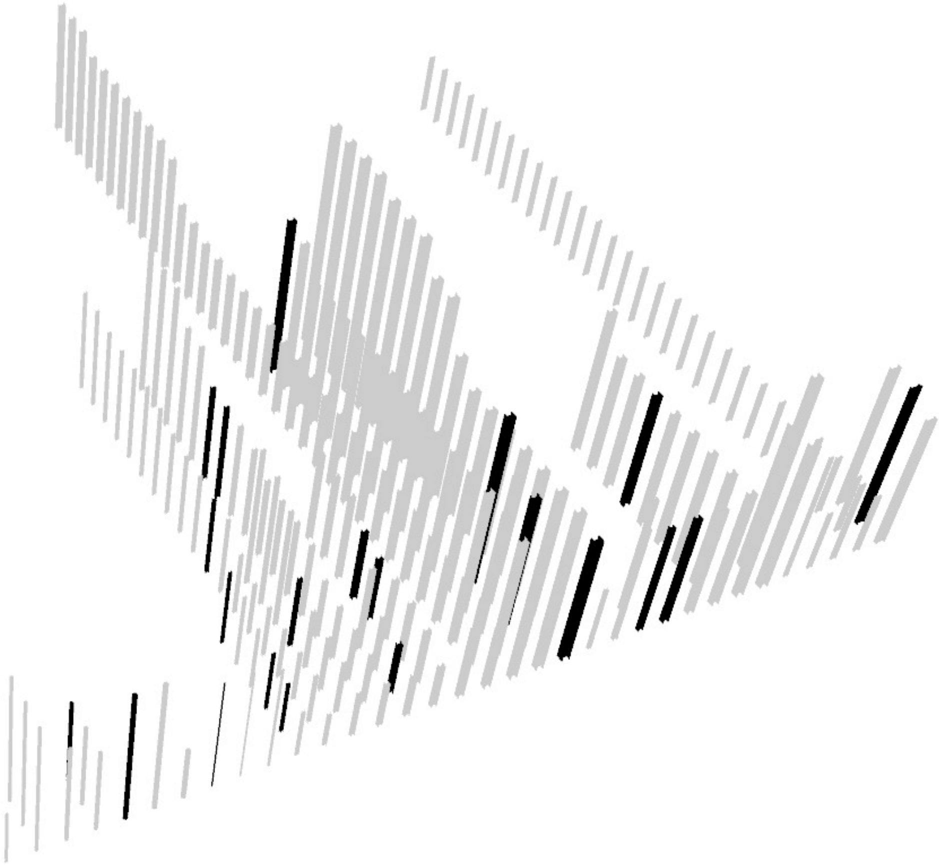


Figure 8.4: Used reclaimed elements

Length	20 m
Width	4 m
Approximate height	4 m
Mass	2142 kg
MKI	€ 285.73
mean UC	0.43
Number of elements	33

Table 8.1: Characteristics truss bridge design example

8.3. Analyses

8.3.1. Sensitivity study performance parameters

A sensitivity study has been performed to understand the influence of two performance parameters. Firstly, the lower unity check limit, which may be set higher to only allow more efficient options to be included in the list of valid options. More iterations will be needed to find these options, but it will increase the final mean unity check of the truss members. Secondly, the number of valid options that first have to be found before comparing them to choose the best option. The larger the list of valid options, the higher the chance that options with higher unity checks are included.

The sensitivity study is carried out by letting the design process run for two different UC-limits: a lower limit of 0,05 and 0,1. Important to note is that these limits needed to be relatively low due to the used stock. The used stock has high amount of large profiles. For certain diagonals these will always be quite overdimensioned. Setting a higher UC limit would not work for this database as no options would be found for these specific diagonals. The performance parameters are therefore presented as input values, so that these may be adapted according to the available stock of elements.

For each of the two UC limits, five runs have been performed with respectively 1, 3, 6, 9 and 12 valid options to be found before letting the script compare them to each other. The analysis was performed on a span of 15 meters and a width of 4 meters. The result of the analysis can be seen in figure 8.5.

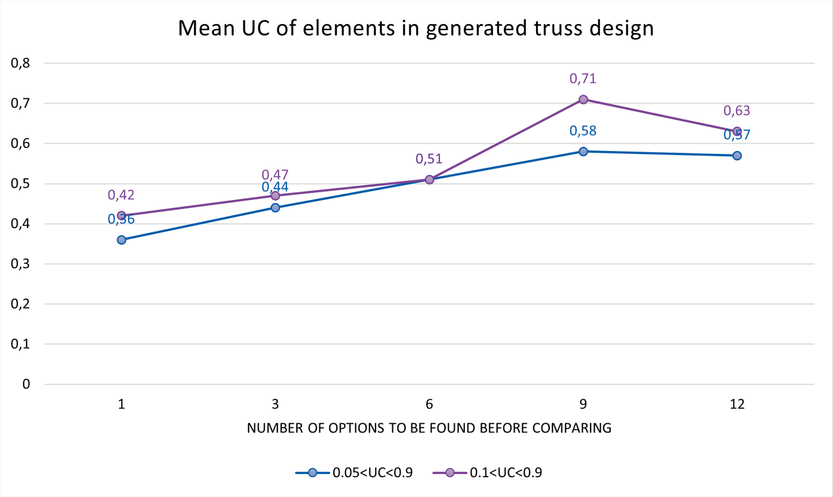


Figure 8.5: Mean unity check values considering different constraints in the algorithm

In the graph it can be seen that in general, the mean UC increases when the number of options increases. This differs for a number of options equal to 12. When the lower UC limit is increased, the mean UC increases as well. This only differs for a number of options equal to 6. The general conclusion from this graph is that both performance parameters work as expected, but do not guarantee a more efficient solution. Also the randomness of the process influences the result.

During the runs, it was not always possible to find the required amount of valid options inside the maximum allowed number of iterations. The process needed to be repeated to find the above values. The maximum number of iterations is also a performance input parameter and can be varied between 20-300. Sometimes, it is desired to set the maximum number of iterations to a lower value and repeat the algorithm several times, instead of letting grasshopper run for a long time. In this case, there is more control over the process and intermediate output values can be used to adapt performance parameters (for example the capacity range constraint) to suit the current situation. The run time of the process also differs quite a lot and is not always linearly related to the strictness of performance parameters. Sometimes the process will run more slowly. This could have to do with laptop properties and again with the randomness of the process. Restarting the algorithm or Grasshopper sometimes helped in that case. The run times of the values in the graph varied from 1 minute to 13 minutes. However, run times in between that did not result in the desired number of valid options sometimes rose to more than 60 minutes. The runs were performed on different days and different networks, so the times may not be compared correctly.

8.3.2. Comparison calculated normalforces

An analysis is performed to check whether the normal forces that were calculated during the growth process using the method of section, correspond to the normal forces calculated in Karamba. In figure 8.6, the result is presented. Tags were placed on top of the elements with the value of the axial force calculated with the method of section ('own'), the axial force calculated by Karamba ('K') and the difference between them in percentages ('diff'). The percentage being calculated with: $\frac{N_{own}-N_K}{N_{own}}$. The tags of the axial forces are coloured purple when the force is a compressive force and dark blue if it is a tensile force. The differences are coloured green when smaller than 20% and red when larger than 20%

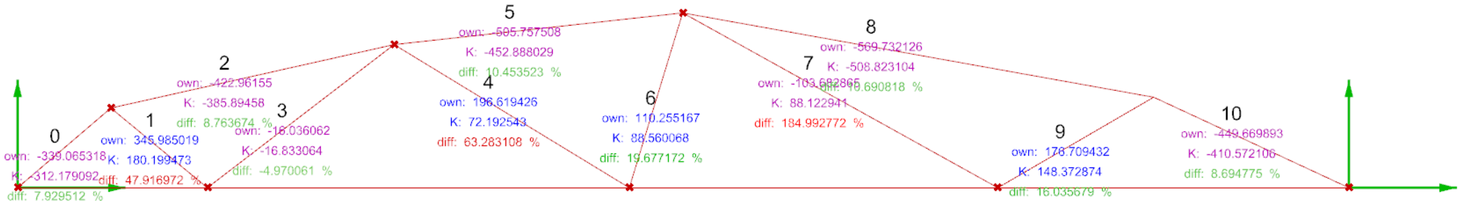


Figure 8.6: Comparison of the calculated axial forces

The result is that the normal forces calculated with Karamba are similar to those calculated with the method of section, but do have minor differences (less than 20%). The only big differences were seen in the elements that needed to be guessed during the growth process, elements 1, 4 and 7 in figure 8.6. These elements needed an assumed horizontal top chord so that their forces could be calculated using the method of section. It is therefore to be expected that the final forces in these elements will differ. An additional step could be scripted to change these elements to more efficient ones now that the final forces are known. However, this would require the elements to be cut to let them fit into the geometry, resulting in a nonlogical truss design including cut-off waste. The decision is therefore made to keep the elements and withhold the truss design from the solutioncloud when the elements have unity checks higher than 1 in the final calculation.

The other differences in axial forces may be due to Karamba making more exact calculations. More specifically, the following differences between the models influence the final forces in the elements:

- The calculation in the growth process using the method of section includes a simplified truss model with all **nodes modelled as hinges**. The model in Karamba includes a continuous bottom chord, without hinges at the nodes. This is a more exact representation of how the truss bridge will transfer loads in reality.
- The **selfweight of the crossbeams** was approximated in the algorithm calculations, as the location and the number of the crossbeams were not yet known and a simplified approach of including the bridge deck properties was preferred. In the Karamba model, the placement of the cross beams is known and so their weight is included more correctly.
- The Karamba model also includes the **extra crossbeams** wherever the width of the truss triangle exceeds 3.5 meters. This causes an increased selfweight load to be carried by the truss and an adapted transfer of forces.

8.3.3. Analysis on different loadcases

As explained in 3.5, the traffic load needs to be applied at the most unfavourable position. Additional analysis is required to check the governing load positions for each element. Due to the irregular geometry and unconventional profile choices, it is not possible to predict the governing load positions for each element. Therefore, non-uniform load cases are not taken into account in the algorithm but have to be checked afterwards. The aim of this analysis is to determine if, when and to what extent non-uniform loadcases are governing in the determination of the required element capacities. This is done in the following way.

In the grasshopper file, two parameters are added which define the position and width of the distributed traffic load. All possible positions and widths are considered. For the purpose of this analysis, only one distributed load was applied. It is recommended that the full analysis of the potential truss design should also consider multiple locally distributed traffic loads spread along the length of the bridge.

In the Karamba model, the pointloads at the locations of the crossbeams are calculated based on a uniformly distributed selfweight of the crossbeams and the steel deck plate, and the defined position and width of the traffic load. Karamba then analyses the model and presents the unity checks of each element, considering strength and member stability. From this list, the largest unity check is retrieved. The evolutionary optimisation tool Galapagos is then used the vary the two traffic load parameters to find the the values that cause the largest overall maximum unity check in the truss design. It may be that the uniformly distributed traffic load is governing or that a more concentrated load is. The amount of elements that fail under a different and more normative loadcase also varies. There are two possibilities when elements fail: either the failing elements can be replaced by larger reclaimed elements which will have to be cut to fit into the design, or the algorithm can be re-run, creating a new truss design.

A couple of runs were performed in which the governing distribution of traffic load was analysed. The results are shown in table 8.2.

Run	**Normative type of traffic load for governing element**	**Percentage max UC / max UC(uniform load)**	**Element with max UC**
1	Uniformly distributed		top chord
2	Locally distributed on left side	192 %	diagonal
3	Uniformly distributed		diagonal
4	Locally distributed on left side	124%	diagonal
5	Uniformly distributed		top chord
6	Uniformly distributed		bottom chord
7	Uniformly distributed		bottom chord
8	Locally distributed on right side	103%	diagonal
9	Uniformly distributed		diagonal
10	Uniformly distributed		top chord

Table 8.2: Results normative traffic load case analysis on a truss designs with span=20m an width=4m

The results show that only in 33% of the truss designs, non-uniform load cases are governing and result in an element to fail. This shows that assuming a uniformly distributed load in the growth algorithm will, in most cases, yield truss design that can withstand all types of loadcases. In case a non-uniform loadcase becomes governing, the increase in load in the governing element, compared to the load calculated with a uniformly distributed load, differs. In run number 8 the load increases only by 8% when changing the load to a more concentrated distributed load on the right side of the truss. However, in run number 2, the load in the governing element increases drastically when changing the loadcase. Furthermore, it can be seen that in the three trusses with a different governing loadcase, the governing element is always a diagonal. This is most likely due to the large varying forces that can be present in diagonals, depending on the position of the load.

9
Comparison with other methods

9.1. Methods and compared characteristics

To make conclusions on the designs obtained using the growth method developed in this thesis, it is important to compare them to designs generated with existing traditional and reuse methods. The comparison focuses on 2D trusses. The following characteristics will be compared:

- The environmental impact of the final design
- The capacity utilisation of the final design
- The material mass of the final design

First of all, a final truss design of the developed growth method will be compared with a truss design following the standard design procedure, without considering the reuse of steel elements. It is to be expected that when not performing a stock constrained design, the final design can be more optimised on material usage and volume. However, new steel has a much larger environmental impact so the reduction in material volume may not compensate in the eventual environmental impact of the design. In the comparison a standard warren truss will be included, which will be optimised on height and number of triangles to find a design with the lowest material volume. In addition, the cross section optimiser component of Karamba will be used to optimise this standard truss even further, now by calculating the smallest required profile for each element. This will result in a truss design with a large number of different profiles, which will be more material efficient, but less easy to manufacture.

Furthermore, the growth method may be compared to other existing stock-constrained design methods. As explained in 5.3.2, two methods are available to be tried out. These are the Phoenix3D tool for Grasshopper developed by the Structural Xploration Lab of EFPL 2021 and the method by van Gelderen (2021), available as a Python script. However, the two methods were not applicable in this comparison. The Python script from van Gelderen could not be run due to problems that arose when changing old Python packages. On the other hand, the Phoenix3D tool incorporated different structural mechanical checks than Karamba, which is used in the other methods. This resulted in a deviating selection of elements. The tool lacks transparency and therefore the method could not be correctly calibrated with the other methods to ensure a correct comparison.

The choice was therefore made to manually fit the reclaimed elements into the initial standard warren truss design, while allowing cut-off waste. This method can then be compared to the developed stock-constrained growth method to determine whether the decision to turn the design process around and let the availability of elements determine the truss geometry does indeed result in more environmental friendly truss designs.

Overall, the methods that will be used in the comparison can be summed up in the following way:

- Normal design process, only considering new steel elements
  - Warren truss with optimised geometry
  - Warren truss with optimised geometry and cross sections
- Stock-constrained design process

  - Fitting a stock of elements into an initial design
  - Generating a truss design with what is available: the stock-constrained growth method developed in this thesis

To compare these methods accurately the following characteristics are equalized:

- Span of the truss bridge: 15 meters
- Width of the bridge: 4 meters
- Weight of the bridge deck: $1.4kN/m^2$
- Crossbeam is added halfway a triangle when distance is larger than 3.5 meters
- Loads modelled as pointloads at the positions of the crossbeams
- Maximum vertical displacement may not be larger than: $\frac{span}{8}$
- Width of the diagonals may not be larger than width of the bottom chord
- Same stock of elements in case of performing a stock-constrained design process

The comparison is performed on 2D trusses and therefore characteristics of the bridgedeck and the check on lateral torsional instability are left out. Also the constraint requiring the bottom chord to have a larger height than the height of the calculated crossbeam is left out.

9.2. Calculation of the environmental impact

In February 2023 an NTA (Nederlandse Technische Afspraak) will be published providing a standard for the reuse of steel elements in the Netherlands. The NTA presents the environmental impact value of new steel (produced from recycled steel scrap) versus reclaimed steel. These values will be used to make a correct comparison between the considered methods.

The values that the NTA provides are the MKI-value (Milieukostenindicator) in €/kg and the GWP (Global Warming Potential) value in kg CO2-eq/kg. They use the methodology prescribed by NEN-EN 15804 (including annex A2), whereby the environmental impact values are calculated per lifecycle module. See figure 9.1 for an overview of these modules.

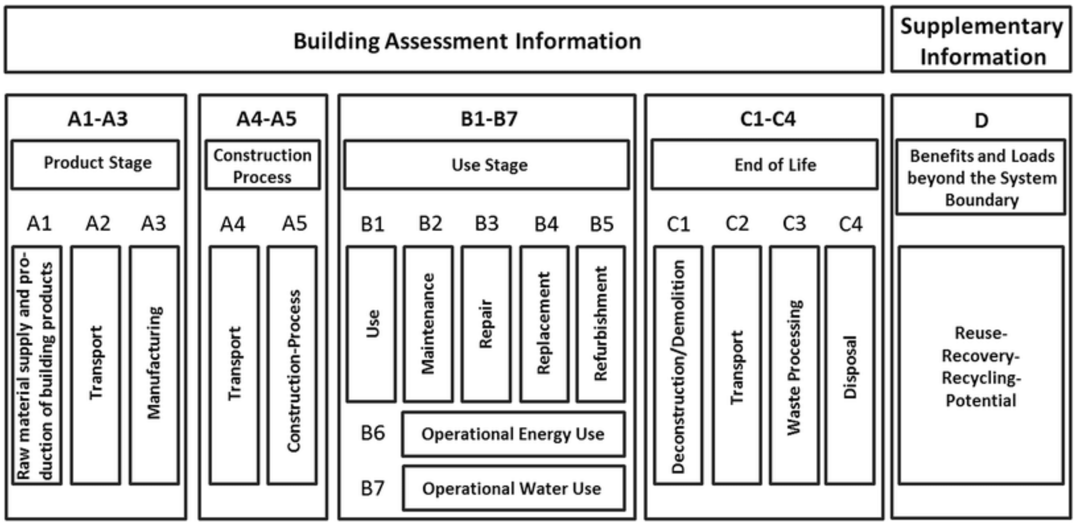


Figure 9.1: System boundaries according to EN 15804/EN 15978

The NTA assumes a 16% reuse in the End of Life stage, for both types of steel. Additionally, the NTA assume that 100% reuse will never be possible as components like endplates will still need to be manufactured.

Therefore it calculates the environmental impact values for reclaimed steel considering 90% reuse and 10% new steel.

		A1-A3	A4	A5	C1	C2	C3	D	SUM	
New steel	**MKI**	0.12	0.0024	0.0053	0.0053	0.00079	0.0052	-0.023	**0.1160**	€/kg
	GWP	1.12	0.02	0.048	0.048	0.0066	0.026	-0.22	**1.0486**	kg CO2-eq/kg
Reclaimed steel	**MKI**	0.021	0.0024	0.0053	0.0053	0.00079	0.0052	-0.0022	**0.0378**	€/kg
	GWP	0.192	0.02	0.048	0.048	0.0066	0.026	-0.017	**0.3236**	kg CO2-eq/kg

Table 9.1: GWP and MKI values per lifecycle module for new and reclaimed structural steel elements (normcommissie 342086 - NTA Hergebruik Staal, 2023, table 6.2, p16)

As can be seen in table 9.1, values are given per kg steel. In the grasshopper files where the methods are being tested, outputs are necessary providing information on the total mass of included new steel elements and the total mass of reclaimed steel elements in the truss design. These masses can subsequently be multiplied by the values in the table to provide a conclusion on the total GWP or MKI value of the truss structure.

A fast calculation can already provide information on the maximum difference in mass that a structure completely made out of reclaimed steel may have so that it will still result in a structure with a lower environmental impact than a minimum-weight structure made out of new steel.

$$MKI_{reclaimed} < MKI_{new}$$
$$\Leftrightarrow 0.0378 * m_{reclaimed} < 0.1160 * m_{new}$$
$$\Leftrightarrow m_{reclaimed} < 3.07 * m_{new}$$

$$GWP_{reclaimed} < GWP_{new}$$
$$\Leftrightarrow 0.3236 * m_{reclaimed} < 1.0486 * m_{new}$$
$$\Leftrightarrow m_{reclaimed} < 3.24 * m_{new}$$

This means that a structure using only reclaimed elements can have up to 3.07 or 3.24 times higher mass than a design variant made of new steel to still have a lower MKI or GWP value, respectively.

As the fitting-in method produces cut-off waste, this waste should be included in the total mass of used reclaimed steel as this waste increases the environmental impact of the structure.

Also in the growth method, the last element of the bottom chord and the last diagonal needs to be cut to fit inside the span. The last element of the bottom chord is chosen so that the length of the spare part is maximised. The reason for this being that the spare part will then be as useful as possible when returned to the stock. Including this spare part as waste material in the environmental impact calculation is thus not entirely correct. The spare part resulting from the cut last diagonal will be considered as cut-off waste, as in this case no effort was made to search for an element that would result in producing a large (enough) spare part. The spare part is included in the database, but to make a fair comparison with the fitting-in method, it has to be considered as waste.

9.3. Execution of the different design methods

9.3.1. Standard warren truss

Besides the input parameters that are equal for all methods in the comparison, the warren truss design has two input parameters that are still variable. These are the height of the truss and the number of triangles. Using the Grasshopper optimiser component Galapagos, these two parameters can be optimised such that the total mass of the structure is minimised. The result is a smart, but standard Warren truss design. The mean UC value of the truss is 0.68. CHS sections are used in the design as this gives the smallest possible mass. The considered stock-constrained design methods have the restriction that the bottom chord may not consist of CHS sections, but since the aim is to test these methods with the most optimal new steel design, it remains a safe comparison. All parameters and constraints that do compete with stock-constrained design methods are considered. The resulting truss design and the calculation of the environmental impact values can be seen in figures 9.2 and 9.3.

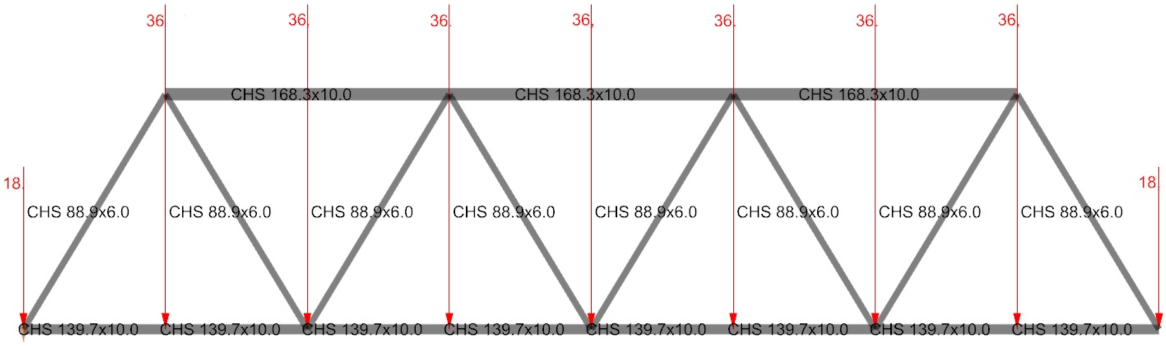


Figure 9.2: Truss design

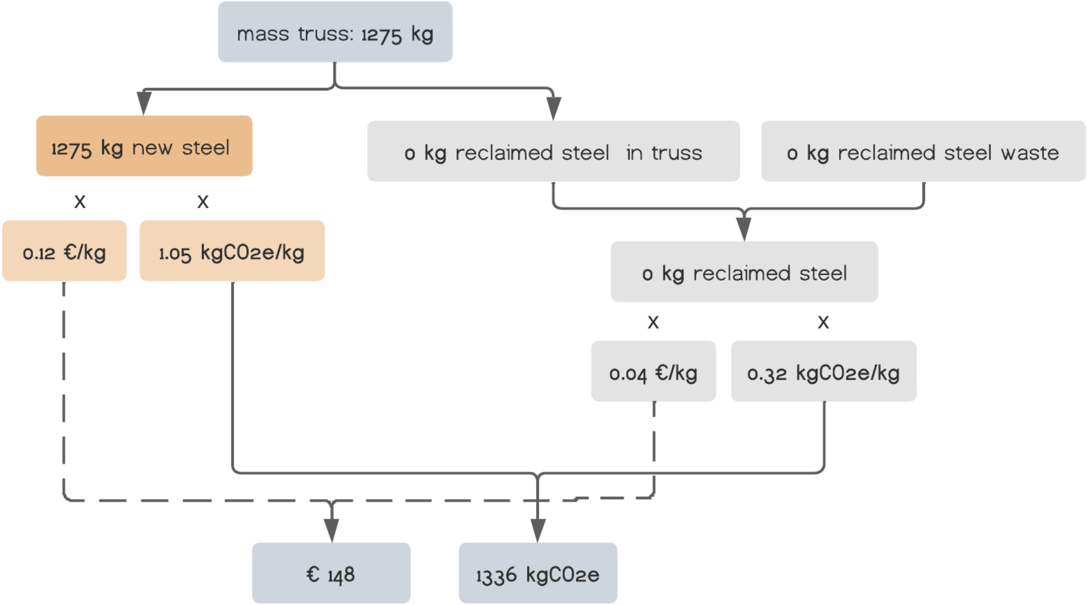


Figure 9.3: Environmental impact values

9.3.2. Standard warren truss with optimised cross sections

Karamba provides a component which can also optimise the cross sections of the truss design. This way, the material mass is further minimised as well as the material efficiency being more maximised. In the optimisation, Karamba calculates the required strength capacity but also the capacity needed so that no element buckling can occur. The result is a fully optimised warren truss, whereby all elements have different profiles. The mean UC value of the truss is 0.91.

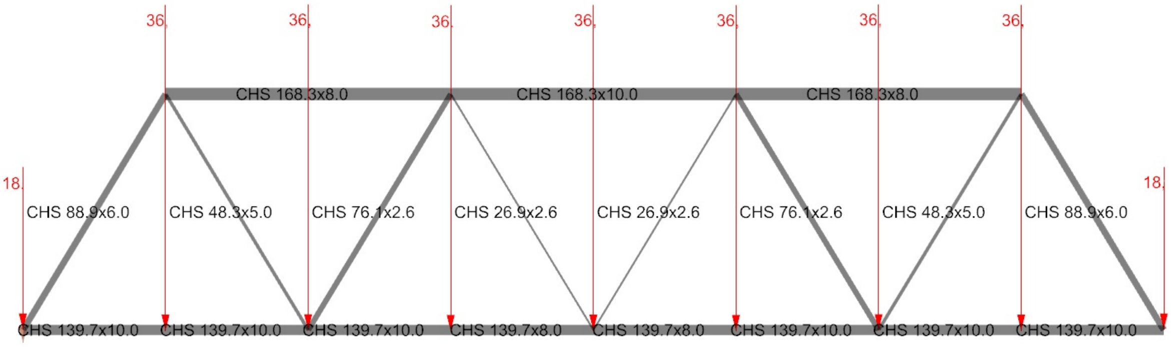


Figure 9.4: Truss design

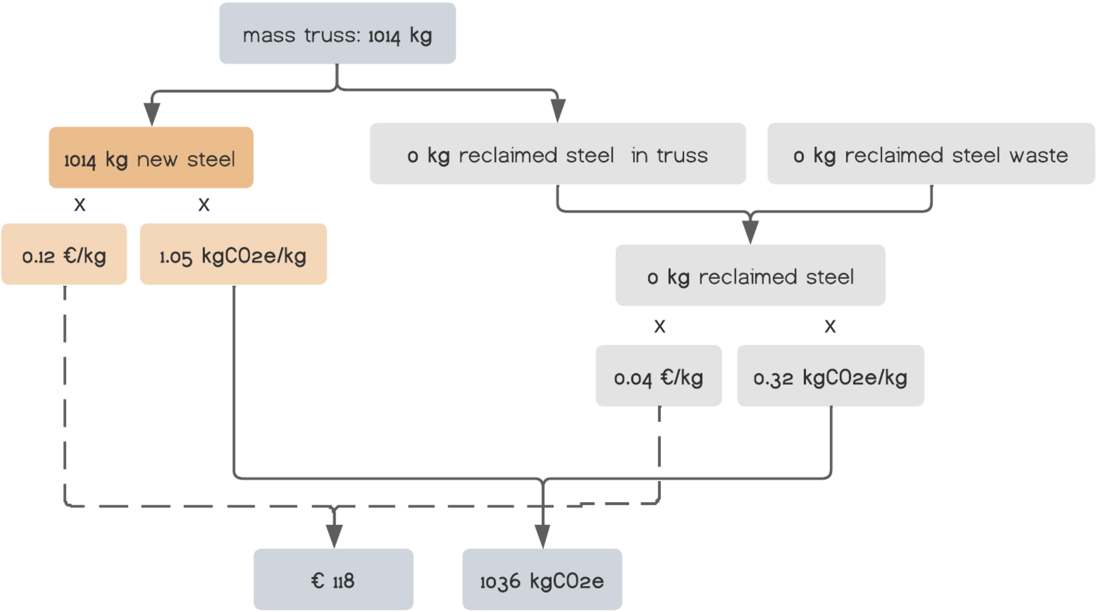


Figure 9.5: Environmental impact values

9.3.3. Fitting in method

In this method, the optimized warren truss is used as starting point. Manually, elements from the stock are matched as good as possible with the elements calculated in the optimal truss design. With the considered dimensions and stock of elements, all elements will be slightly overdimensioned. Also, every element needs to be cut in order to fit into the geometry. The mean UC value of the truss is 0.28.

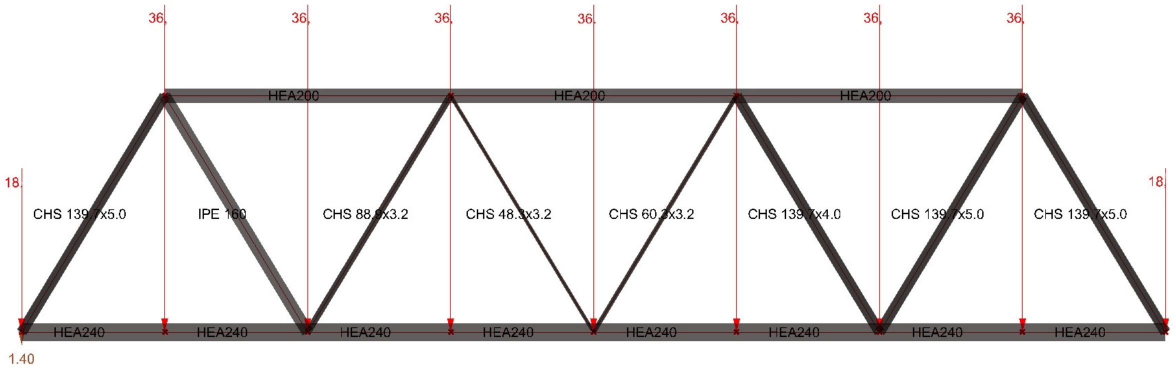


Figure 9.6: Truss design

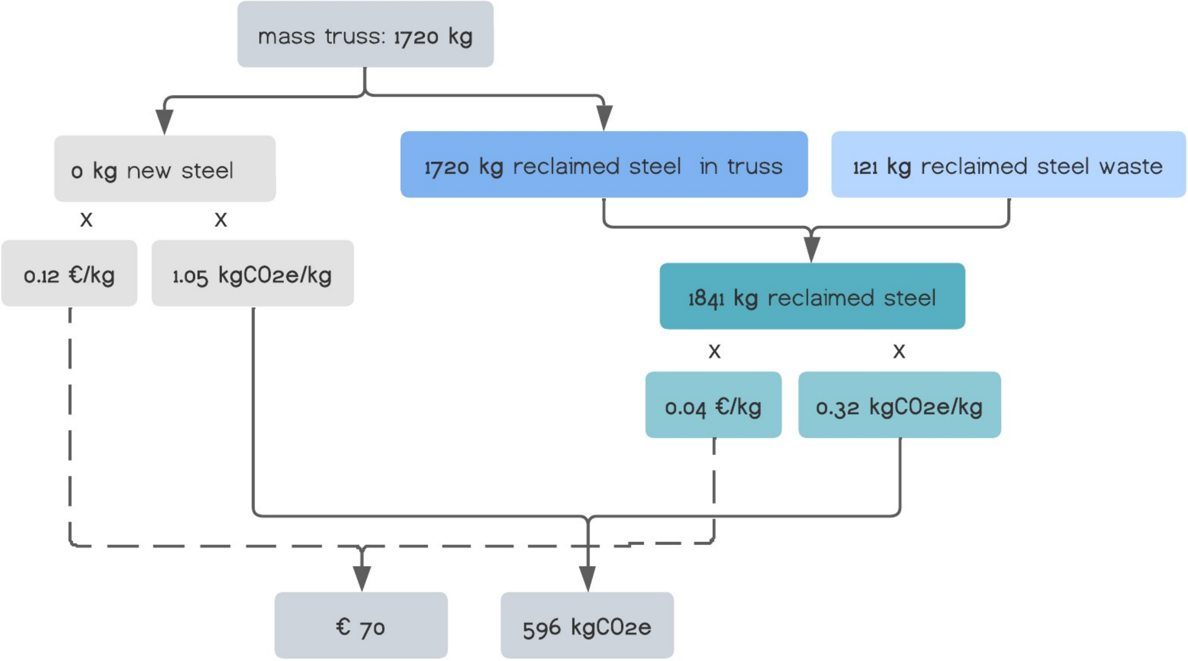


Figure 9.7: Environmental impact values

9.3.4. Growth method

The growth method has been performed using the same input values and other characteristics as summarised in paragraph 9.1. The used reclaimed elements are visualized in black in the total stock, see figure 9.9. The mean UC value of the truss is 0.43.

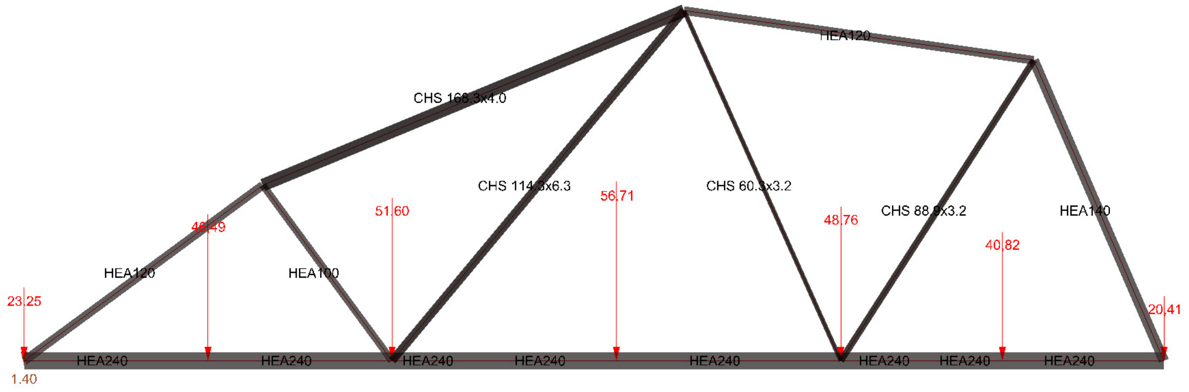


Figure 9.8: Truss design

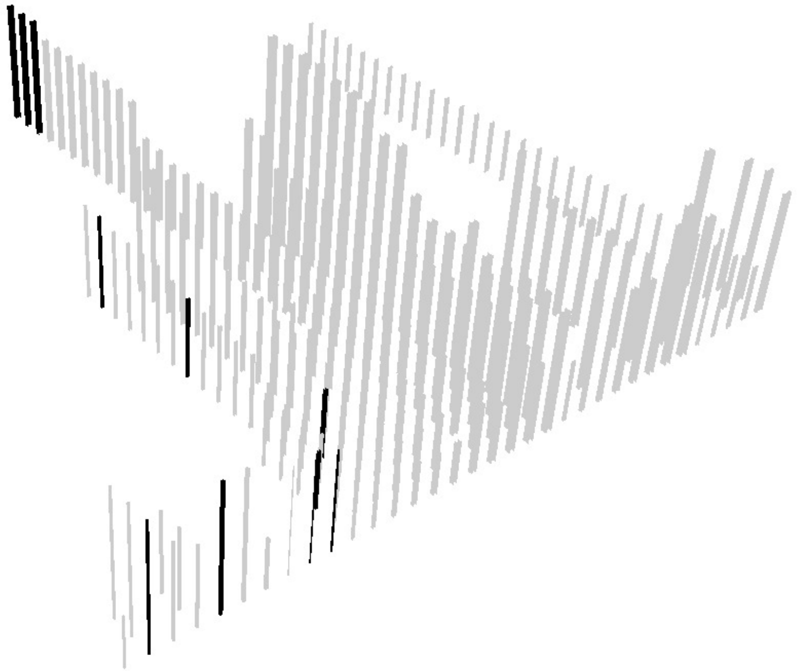


Figure 9.9: Used stock

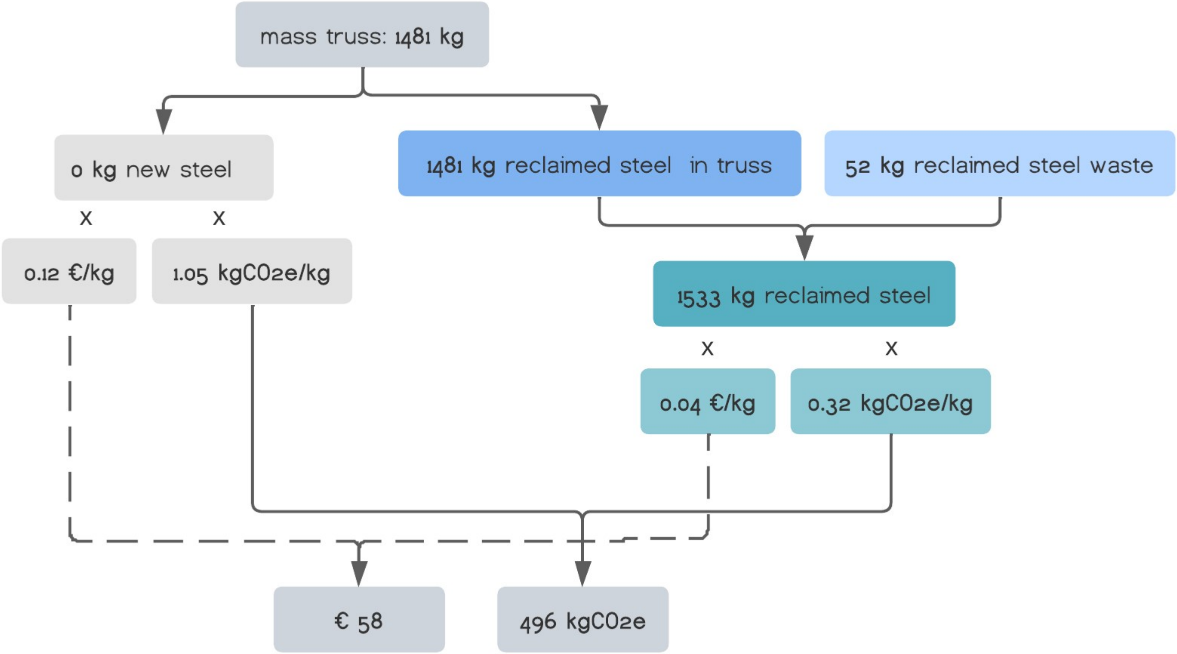


Figure 9.10: Environmental impact values

9.4. Comparison

An overview of the results is given in the graph below.

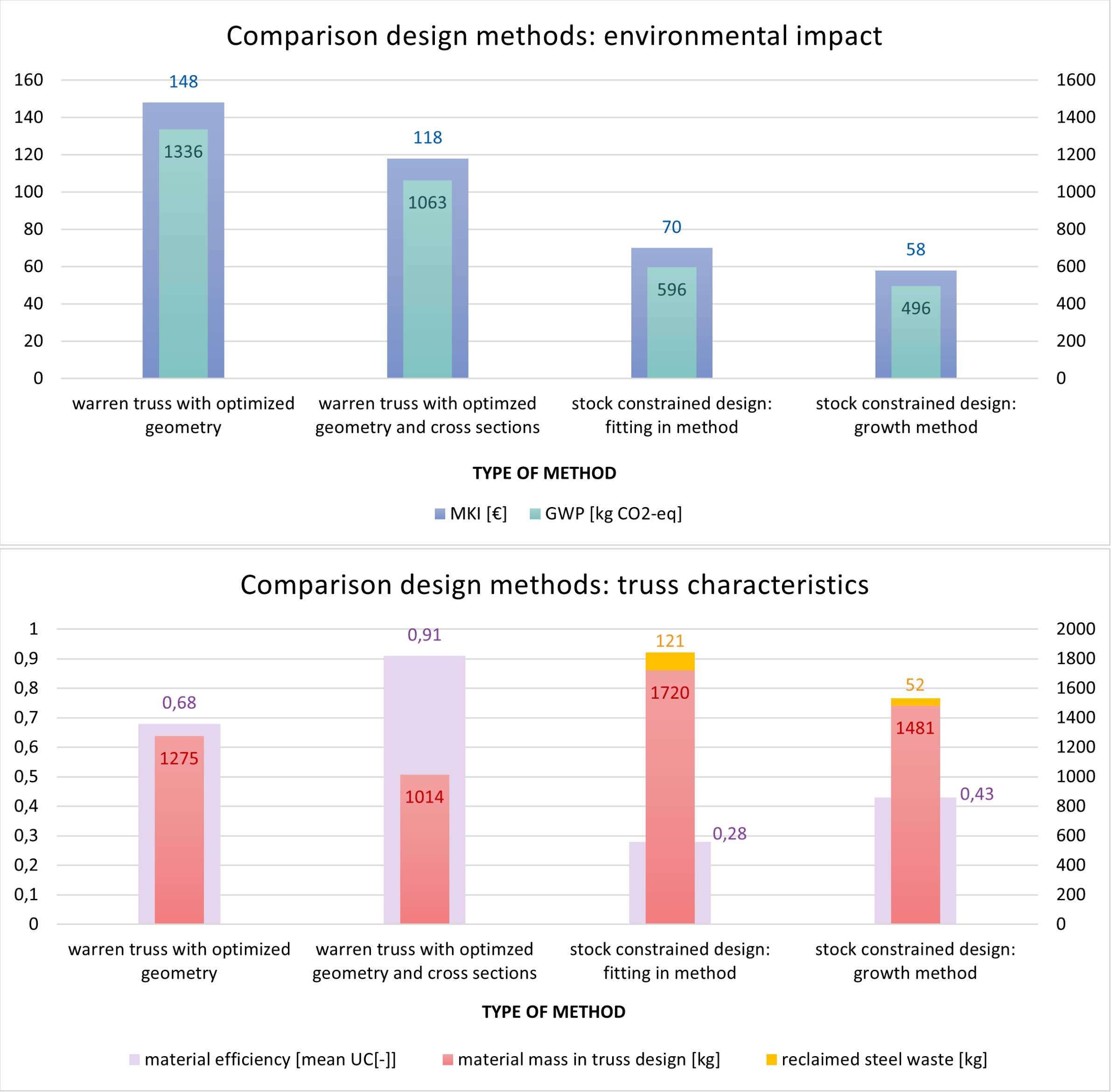


Figure 9.11: Comparison design methods

As can be seen in the graphs, the GWP and MKI values are the highest for a standard truss with optimised height and number of triangles, manufactured from only new steel. Next comes the standard truss out of new steel but this time with optimised cross sections. Which is to be expected as this truss needs less material. Furthermore, the method of fitting in elements and cutting them to appropriate lengths comes next in line. The growth method shows the lowest environmental impact factors. The environmental impact is reduced by 63% compared to the warren truss with optimised geometry, considered as the most commonly used truss design method nowadays.

In first instance, it can be concluded that however the normal design methods provide design options with lower steel mass and higher capacity utilisation, their environmental impacts are still higher than the stock-constrained design methods due to the use of new steel. In second instance, it can be concluded the growth method provides more environmental friendly truss designs than the fitting in method due to cut-off waste being avoided and in this case also an overall lower material mass in the truss design. The fitting-in method could have lower mass if the stock of elements better suits the members of the initially designed truss. However, once the supply of reclaimed elements starts deviating, the growth method holds the potential to adapt the design so that it better matches with the available elements. This is also reflected in the mean unity checks of the two methods. The growth method includes the reclaimed elements more efficiently in terms of capacity utilisation, resulting in a higher mean unity check than the fitting in method.

To conclude, the growth method does not provide the truss design with the lowest steel mass or the highest material efficiency, but it can provide a superstructure design with the lowest environmental impact compared to the other three considered methods. As explained at the beginning of the thesis, this was also the goal of this method.

10
Conclusions and recommendations

10.1. Conclusions

The aim of this thesis was to investigate whether and how the traditional design process can be changed to make more efficient use of the available limited stock of reclaimed steel elements. The starting point was to turn the design process around so that form follows availability. Additionally, the question was whether such an inverted design process would lead to designs without the need for new steel elements and without excessive overdimensioning. These two aspects may be summarized in one overarching goal of this thesis: to create truss bridge designs with lower environmental impact.

First of all, it may be concluded that the design process may and should be turned around and that is by using the idea behind the growth method. The growth method is a truss topology optimisation method where truss designs are grown in density or dimensions. In case of stock-constrained design, available elements can be attached to each other to generate a triangular mesh. This removes the need to fit elements into an original design or a grid of possible positions, as in the commonly used ground-structure method. By moving away from fitting-in methods, truss designs can be generated that better match element availability and do not require new elements or result in a large amount of cut-off waste. In this thesis, a stock constrained growth method is developed where, after introducing the required width and length of the bridge and the available stock of reusable steel elements, a solution cloud of possible truss bridge designs is generated. The steps in the algorithm include required calculations, design constraints and optimisation steps so that the resulting designs are valid and locally optimal in terms of capacity utilisation. Thus, all truss designs comply with Eurocode provisions (except the check of certain loadcases mentioned in the recommendations) and take into account the manufacturability of the required connections. Furthermore, the following design choices were made:

- The design process is applicable for simply supported models.
- The tool is designed for truss bridges with steel deckplates, supported on crossbeams. If the type of deckplate changes, the deckplate weight can be easily adapted in the input parameters. If a different bridgedeck system is used which does not include crossbeams, the transfer of loads and the stiffness of the bridge deck should be adapted in the tool.
- The algorithm includes dimensional and geometrical constraints so that diagonals may easily be welded onto the bottom chord, the sequenced elements in the bottom chord may be welded or bolted on to each other and consider the attachability of the bridgedeck and the diagonals and top chord elements may be bolted onto a connection plate (see 3.2.2). When different connections are considered, these constraints should be reviewed in the algorithm.

Furthermore, an answer can be given to whether the changed design process leads to efficient designs without the need for new steel elements. The thesis resulted in an algorithm whereby no new steel is needed in the resulting truss bridge design. Moreover, the algorithm ensures that almost no cut-off waste is produced. Furthermore, the algorithm makes design choices with the aim of maximising the capacity utilisation of the final truss design. The design process results in a solution cloud with locally optimal truss designs in terms of capacity utilisation. By changing the performance parameters of the algorithm, the strictness of the required minimum capacity utilisation of the final design can be changed, as well as the thoroughness of the algorithm to find more optimal solutions. This solution cloud of valid designs allows the user to set his own priority in objectives. The design can be chosen with the highest capacity utilisation, the lowest mass, the least number of elements, or the lowest environmental shadowcost.

Finally, a comparison was made between the developed stock-constrained growth method and existing traditional and reuse methods. Compared to the traditional truss made of new steel, the truss of the growth method has 16% more mass. However, the case study shows that by avoiding new steel in the design, presenting a method that does not produce any cut-off waste and adapting the geometry to element availability, the developed algorithm still results in a steel truss with 17% less embodied carbon than the truss design whereby reclaimed elements are fit in afterwards, and 63% less embodied carbon than the traditionally designed new steel truss.

10.2. Recommendations

First of all, the following points could be added to the growth process to improve and expand it further:

- The check on minimum natural frequency so that no dynamic issues will arise in the final truss bridge designs
- The inclusion of different loadcases, considering scattered unfavourable loading positions and horizontal wind loads

Furthermore, a larger analysis on the performance parameters may be performed so that their influence on the run time and efficiency of the design results can be made clearer. These performance parameters include:

- The limits on allowed unity checks in the algorithm
- The number of valid triangle options that need to be found before they are compared to select the most efficient option
- The number of iterations that Grasshopper can handle before crashing
- The capacity range included as a constraint in the random selection of elements in the algorithm

Also the effect of the type of stock on the performance of the algorithm may be analysed to clarify the applicability of the design method.

Fourth, a broader comparison can be made of the developed design process with other methods. An interesting comparison would be to compare the method with a fully optimised truss design, where the geometry is not limited to standard warren truss topology. Grasshopper tools like Peregine or voxel-based topology optimisation methods could be used. These methods focus on minimisation of mass and thereby aim to lower the environmental impact in a different manner.

Finally, the following steps are needed to start using the developed algorithm:

- The encouragement of dismantling rather than demolition of structures
- The development of an overarching database
- The application of the protocol for reuse of steel in the Netherlands, which clarifies the required tests so that reclaimed steel can safely be used
- The development and application of a general material passport for reclaimed structural elements so that all required information is recorded
- The implementation of a system to reserve elements from a stock

Bibliography

AbdelRahman, M. (2017). GH_CPython: CPython plugin for grasshopper. https://doi.org/10.5281/ZENODO.888148

Achtziger, W. (2007). On simultaneous optimization of truss geometry and topology. *Structural and Multidisciplinary Optimization*, *33*(4), 285–304.

AmRoR. (2021, January). *Handreiking hergebruik bruggen*. https://www.nationalebruggenbank.nl/wp-content/uploads/2021/03/20607-RWS-AMROR-handleiding-bruggen-16PS.pdf

Bendsøe, M. P., & Kikuchi, N. (1988). Generating optimal topologies in structural design using a homogenization method. *Computer methods in applied mechanics and engineering*, *71*(2), 197–224.

Bila, K. (2019). Fatigue analysis of a lively footbridge.

Bruggenbank. (2021, November 10). *Missie visie bruggenbank*. Retrieved March 28, 2022, from https://www.bruggenbank.nl/over-ons/

Brütting, J., Desruelle, J., Senatore, G., & Fivet, C. (2018). Design of truss structures through reuse. *Structures*, *18*, 128–137.

Brütting, J., Ohlbrock, P. O., Hofer, J., & D'Acunto, P. (2021). Stock-constrained truss design exploration through combinatorial equilibrium modeling. *International Journal of Space Structures*, *36*(4), 253–269.

Brütting, J., Senatore, G., & Fivet, C. (2018). Optimization formulations for the design of low embodied energy structures made from reused elements. *Workshop of the European Group for Intelligent Computing in Engineering*, 139–163.

Brütting, J., Senatore, G., & Fivet, C. (2019). Form follows availability–designing structures through reuse. *Journal of the International Association for Shell and Spatial Structures*, *60*(4), 257–265.

Bukauskas, A. (2020). *Inventory-constrained structural design* (Doctoral dissertation). University of Bath.

Carroll. (2019, January). *Warren truss*. https://www.degreetutors.com/what-is-a-truss/

Cirkelstad. (2021). *Paspoorten voor de bouw*. https://platformcb23.nl/images/leidraden/GreenPaper_Materiaalpaspoorten.pdf

Coniglio, S., Morlier, J., Gogu, C., & Amargier, R. (2020). Generalized geometry projection: A unified approach for geometric feature based topology optimization. *Archives of Computational Methods in Engineering*, *27*(5), 1573–1610.

Cui, C.-Y., & Jiang, B.-S. (2014). A morphogenesis method for shape optimization of framed structures subject to spatial constraints. *Engineering Structures*, *77*, 109–118.

Cuvilliers, P., Mueller, C., & of Technology, M. I. (2017). GHPython Remote. https://github.com/pilcru/ghpythonremote

cype. (n.d.). *Types of implemented connections for flat trusses with hollow structural sections*. https://info.cype.com/en/product/joints-v-flat-trusses-with-hollow-structural-sections/

de Boer, M., Teuffel, I. P., Habraken, A., & van Hove, B. (2021). Design and optimization of structures with re-used primary elements and new robotic manufactured connections.

Dorn, W. S., Gomory, R. E., & Greenberg, H. J. (1964). Automatic design of optimal structures.

Dunant, C. F., Drewniok, M. P., Sansom, M., Corbey, S., Cullen, J. M., & Allwood, J. M. (2018). Options to make steel reuse profitable: An analysis of cost and risk distribution across the uk construction value chain. *Journal of Cleaner Production*, *183*, 102–111.

ESDEP. (2012). *Welded column splice for sections of differing serial size*. http://fgg-web.fgg.uni-lj.si/~/pmoze/esdep/master/wg11/l0800.htm

Fairclough, H., & Gilbert, M. (2020). Layout optimization of simplified trusses using mixed integer linear programming with runtime generation of constraints. *Structural and Multidisciplinary Optimization*, *61*(5), 1977–1999.

Gemeente Amsterdam. (2017, April). *Meerjarenplan fiets 2017 – 2022*. Verkeer en openbare ruimte van de gemeente Amsterdam. https://www.verkeerskunde.nl/Uploads/2017/4/concept-mjp-fiets-2017---2022-vrijgave-voor-inspraak.pdf

Gemeente Rotterdam & Goudappel Coffeng. (2017, January). *Slimme bereikbaarheid voor een gezond, economisch sterk en aantrekkelijk rotterdam*. https://archief12.archiefweb.eu/archives/archiefweb/20200809091431/

http://www.rotterdam.nl/wonen-leven/mobiliteitsaanpak/Stedelijk-Verkeersplan-Rotterdam-20170123.pdf

Gilbert, M., & Tyas, A. (2003). Layout optimization of large-scale pin-jointed frames. *Engineering computations*.

Gorgolewski, M. (2008). Designing with reused building components: Some challenges. *Building Research & Information*, *36*(2), 175–188.

Guo, X., Zhang, W., & Zhong, W. (2014). Doing topology optimization explicitly and geometrically—a new moving morphable components based framework. *Journal of Applied Mechanics*, *81*(8).

He, L., Gilbert, M., & Song, X. (2019). A python script for adaptive layout optimization of trusses. *Structural and Multidisciplinary Optimization*, *60*(2), 835–847.

Heinemeyer, C., Martin, P.-O., Trometor, S., Keil, A., Cunha, A., Lukic, M., Lemaire, A., Butz, C., Chabrolin, B., Caetano, E., et al. (2009). *Design of lightweight footbridges for human induced vibrations: Background document in support to the implementation, harmonization and further development of the eurocodes; joint report, prepared under the jrc-eccs cooperation agreement for the evolution of eurocode 3 (programme of cen/tc 250)* (tech. rep.). Lehrstuhl für Stahl-und Leichtmetallbau und Institut für Stahlbau.

Hillege. (2022, August 9). *Impact categories (lca) – overview*. Retrieved October 14, 2022, from https://ecochain.com/knowledge/impact-categories-lca/

Hirt, M., & Lebet, J.-P. (2013). *Steel bridges: Conceptual and structural design of steel and steel-concrete composite bridges*. CRC Press.

Historic Bridge Foundation. (2021). *Bridge types – historic bridge foundation*. Retrieved March 28, 2022, from https://historicbridgefoundation.com/bridge-types/

Iacovidou, E., & Purnell, P. (2016). Mining the physical infrastructure: Opportunities, barriers and interventions in promoting structural components reuse. *Science of the Total Environment*, *557*, 791–807.

IEA. (2021). 2021 global status report for buildings and construction. *United Nations Environment Programme*.

International Organization for Standardization. (2006). *Iso 14040:2006(en) environmental management — life cycle assessment — principles and framework*. Retrieved March 25, 2022, from https://www.iso.org/obp/ui/#iso:std:iso:14040:ed-2:v1:en

ipv Delft. (2015, June). *Brief dutch design manual for bicycle and pedestrian bridges, english summary of the crow design guide*. CROW.

kiwa. (n.d.). *En 1090-1 certificaat: Ce-markering staal aluminiumconstructies*. Retrieved June 14, 2022, from https://www.kiwa.com/nl/nl/service/en-1090-1-ce-marking-steel-and-aluminum-constructions/

Klemmt, C. (2021). *Structural growth* (Doctoral dissertation).

LBPSight. (2022, February). *Analyse co2 impact staal in de bouw*. https://duurzamemetaalbouw.nl/app/uploads/2021/11/analyse-CO2-impact-staal-in-de-bouw1.pdf

LimitState. (2019). *Peregrine*. Retrieved June 8, 2022, from https://www.limitstate.com/peregrine

Madaster. (2022, January 17). *Madaster*. Retrieved March 28, 2022, from https://madaster.com/

Mangus & Sun. (2000). *Bridge engineering handbook, orthotropic deck bridges*. CRC Press. http://freeit.free.fr/Bridge%20Engineering%20HandBook/ch14.pdf

Martinez, P., Marti, P., & Querin, O. (2007). Growth method for size, topology, and geometry optimization of truss structures. *Structural and Multidisciplinary Optimization*, *33*(1), 13–26.

McCormac, J. . C. . (1984). *Structural analysis*. Harper Row.

MetaalNieuws. (2022, February 11). *Meer hergebruik staal kan co2-emissie bouw verlagen*. Retrieved March 17, 2022, from https://www.metaalnieuws.nl/meer-hergebruik-staal-kan-co2-emissie-bouw-verlagen/

Ministerie van Infrastructuur en Waterstaat. (2022, July 25). *Ienw heeft een voorbeeldrol*. https://magazines.rijksoverheid.nl/ienw/duurzaamheidsverslag/2022/01/energiegebruik-en-emissies-eigen-organisatie

Nakamura, S.-i., & Kawasaki, T. (2006). Lateral vibration of footbridges by synchronous walking. *Journal of Constructional steel research*, *62*(11), 1148–1160.

Nationale Bruggenbank. (2022, February 15). *Nationale bruggenbank*. Retrieved March 28, 2022, from https://www.nationalebruggenbank.nl/

Nibe. (n.d.). *Veel gestelde vragen milieuclassificaties*. Retrieved November 17, 2022, from https://www.nibe.info/nl/faq

normcommissie 342086 - NTA Hergebruik Staal. (2023, February). *Norm voor hergebruik constructieve stalen elementen*.

Normcommissie 351001 'Technische Grondslagen voor Bouwconstructies'. (2011a, December). *Nen-en 1991-1-4+a1+c2: Actions on structures - part 1-4: General actions - wind actions*. Retrieved May 4, 2022, from https://connect.nen.nl/Family/Detail/29446?compId=10037&collectionId=0

Normcommissie 351001 'Technische Grondslagen voor Bouwconstructies'. (2011b, December). *Nen-en 1993: Ontwerp en berekening van staalconstructies - deel 1-8: Ontwerp en berekening van verbindingen*. Retrieved September 20, 2022, from https://connect.nen.nl/Standard/Detail/164764?compId=10037&collectionId=0

Normcommissie 351001 'Technische Grondslagen voor Bouwconstructies'. (2015, October). *Nen-en 1991-2+c1: Actions on structures - part 2: Traffic loads on bridges*. Retrieved May 4, 2022, from https://connect.nen.nl/Family/Detail/31954?compId=10037&collectionId=0

Normcommissie 351001 'Technische Grondslagen voor Bouwconstructies'. (2017, July). *Nen-en 1993-2: Design of steel structures - part 2: Steel bridges*. Retrieved May 3, 2022, from https://connect.nen.nl/Family/Detail/31906?compId=10037&collectionId=0

Normcommissie 351001 'Technische Grondslagen voor Bouwconstructies'. (2018, June). *Nen-en 1090-2: Execution of steel structures and aluminum structures - part 2: Technical requirements for steel structures*. Retrieved May 3, 2022, from https://connect.nen.nl/Family/Detail/18228?compId=10037&collectionId=0

Normcommissie 351001 'Technische Grondslagen voor Bouwconstructies'. (2019a, November). *National annex to nen-en 1990+a1:2006+a1:2006/c2:2019 eurocode: Basis of structural design*. Retrieved May 4, 2022, from https://connect.nen.nl/Family/Detail/31942?compId=10037&collectionId=0

Normcommissie 351001 'Technische Grondslagen voor Bouwconstructies'. (2019b, November). *Nationale bijlage bij nen-en 1991-2+c1: Eurocode 1: Belastingen op constructies - deel 2: Verkeersbelasting op bruggen*. Retrieved May 3, 2022, from https://connect.nen.nl/Standard/Detail/3622776?compId=10037&collectionId=0

NSC. (2019, October 9). *Bottom chord joint – webs vertical*. https://www.newsteelconstruction.com/wp/connection-design-in-trusses/

Oakland Museum of California. (2016). *Bay bridge steel*. Retrieved March 17, 2022, from https://museumca.org/bay-bridge-steel

Packer. (2018, December). *Alternate bolted lap splice connections: With the plate welded into a slot in the hss (left), and with the plate welded inside the hss (right)*. https://steeltubeinstitute.org/resources/hidden-bolted-hss-splices/

PBL Netherlands Environmental Assessment Agency, The Hague. (2017). *Opportunities for a circular economy*. Retrieved March 29, 2022, from https://themasites.pbl.nl/o/circular-economy/

Platform CB'23. (2018). *Over platform cb'23*. Retrieved April 15, 2022, from https://platformcb23.nl/over-platform-cb-23

Platform CB'23. (2021, July). *Leidraad circulair ontwerpen*. https://platformcb23.nl/images/leidraden/PlatformCB23_Leidraad_Circulair-Ontwerpen_versie1.pdf

Platform CB'23. (2022a). *Leidraad toekomstig hergebruik*. https://platformcb23.nl/images/consultatie/2022/conceptleidraad-toekomstig_hergebruik-1-2022-03-16.pdf

Platform CB'23. (2022b, March). *Leidraad paspoorten voor de bouw*. https://platformcb23.nl/images/consultatie/2022/conceptleidraad-paspoorten_voor_de_bouw-3-2022-03-16.pdf

Preisinger, C., & Heimrath, M. (2014). Karamba—a toolkit for parametric structural design. *Structural Engineering International*, *24*(2), 217–221.

Querin, O. M., Steven, G. P., & Xie, Y. M. (1998). Evolutionary structural optimisation (eso) using a bidirectional algorithm. *Engineering computations*.

Ramirez. (2019). *Diagrams of four types of truss engineering*. https://www.teachengineering.org/lessons/view/ind-2472-analysis-forces-truss-bridge-lesson

Shallan, O., Eraky, A., Sakr, T., & Hamdy, O. (2014). Optimization of plane and space trusses using genetic algorithms. *International Journal of Engineering and Innovative Technology (IJEIT)*, *3*, 66–73.

Snyder, D. . (2020, November 19). *Sustainability of steel in infrastructure*. Retrieved March 17, 2022, from https://www.shortspansteelbridges.org/steel-sustainability/

Snyder, D. . (2022, January 21). *Ohio county finds creative, sustainable steel solutions for bridge replacement*. Retrieved March 17, 2022, from https://www.shortspansteelbridges.org/ohio-county-finds-creative-sustainable-steel-solutions-for-bridge-replacement/

Sotiropoulos, S., & Lagaros, N. D. (2020). Topology optimization of framed structures using sap2000. *Procedia Manufacturing*, *44*, 68–75.

Steel Construction Institute, Brown, D. . G. ., Pimentel, & Sansom. (2019). *Structural steel reuse*. SCI. https://steel-sci.com/assets/downloads/steel-reuse-event-8th-october-2019/SCI_P427.pdf steelconstruction.info. (2022). *Design of steel footbridges*. Retrieved March 21, 2022, from https://www.steelconstruction.info/Design_of_steel_footbridges#Trusses_and_Vierendeel_girder_bridges

Svanberg, K. (1987). The method of moving asymptotes—a new method for structural optimization. *International journal for numerical methods in engineering*, *24*(2), 359–373.

Svanberg, K. (2002). A class of globally convergent optimization methods based on conservative convex separable approximations. *SIAM Journal on Optimization*, *12*(2), 555–573. https://doi.org/10.1137/S1052623499362822

Svanberg, K. (2007). Mma and gcmma, versions september 2007. *Optimization and systems theory*, *104*.

Tekla. (2021). *Beam to beam bolted end plate splice*. https://support.tekla.com/doc/tekla-structures/2021/det_appb_connection_map_splice_connections

Terwel, Moons RC, & Korthagen RC. (2021). Voorbij de pioniersfase. *Bouwen met Staal*, *54*(284).

United Nations. (2020). *Net zero coalition*. Retrieved April 6, 2022, from https://www.un.org/en/climatechange/net-zero-coalition

van Gelderen, T. (2021). Truss topology optimization with reused steel elements: An optimization tool for designing steel trusses with a set of reclaimed elements.

van Maastrigt, J. (2019). Quantifying life cycle environmental benefits of circular steel building designs: Development of an environmental assessment tool for reuse of steel members in building designs for the netherlands.

Vierlinger, R., & Bollinger, K. (2014). Acommodating change in parametric design.

Wakefield, K. . (2017, December 15). *Cmg and treasure island community development awarded bay bridge steel*. Retrieved March 17, 2022, from https://www.cmgsite.com/cmg-treasure-island-community-development-awarded-bay-bridge-steel/

Wardenier, J., Packer, J., Zhao, X.-L., & Van der Vegte, G. (2002). *Hollow sections in structural applications*. Bouwen met staal Rotterdam„ The Netherlands.

Warmuth, J., Brütting, J., & Fivet, C. (2021). Computational tool for stock-constrained design of structures. *Proceedings of the IASS Annual Symposium 2020/21 and the 7th International Conference on Spatial Structures*, (CONF), 1–9.

World Steel Association. (2022). *Steel facts*. Retrieved November 17, 2022, from https://worldsteel.org/wp-content/uploads/steelFacts-2022-2.pdf?x25904

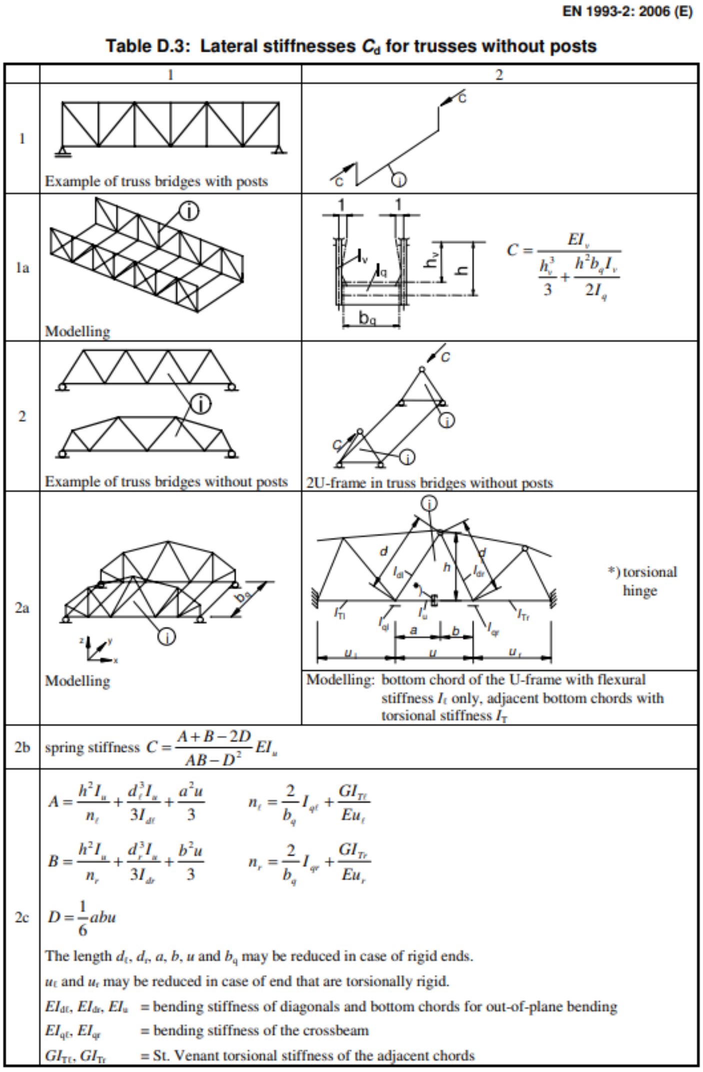
Zhang, W., Yuan, J., Zhang, J., & Guo, X. (2016). A new topology optimization approach based on moving morphable components (mmc) and the ersatz material model. *Structural and Multidisciplinary Optimization*, *53*(6), 1243–1260.

Zwierzycki, M. (2015). Anemone plugin for Grasshopper. https://www.food4rhino.com/en/app/anemone

A

Eurocode tables

A.1. Lateral stiffnesses for trusses without posts



EN 1993-2: 2006 (E)

Table D.3: Lateral stiffnesses C_d for trusses without posts

	1	2
1	Example of truss bridges with posts	
1a	Modelling	$C = \dfrac{EI_v}{\dfrac{h_v^3}{3} + \dfrac{h^2 b_q I_v}{2I_q}}$
2	Example of truss bridges without posts	2U-frame in truss bridges without posts
2a	Modelling	*) torsional hinge Modelling: bottom chord of the U-frame with flexural stiffness I_ℓ only, adjacent bottom chords with torsional stiffness I_T
2b	spring stiffness $C = \dfrac{A+B-2D}{AB-D^2} EI_u$	
2c	$A = \dfrac{h^2 I_u}{n_\ell} + \dfrac{d_\ell^3 I_u}{3I_{d\ell}} + \dfrac{a^2 u}{3}$ $\quad n_\ell = \dfrac{2}{b_q} I_{q\ell} + \dfrac{GI_{T\ell}}{Eu_\ell}$ $B = \dfrac{h^2 I_u}{n_r} + \dfrac{d_r^3 I_u}{3I_{dr}} + \dfrac{b^2 u}{3}$ $\quad n_r = \dfrac{2}{b_q} I_{qr} + \dfrac{GI_{Tr}}{Eu_r}$ $D = \dfrac{1}{6} abu$ The length d_ℓ, d_r, a, b, u and b_q may be reduced in case of rigid ends. u_ℓ and u_r may be reduced in case of end that are torsionally rigid. $EI_{d\ell}$, EI_{dr}, EI_u = bending stiffness of diagonals and bottom chords for out-of-plane bending $EI_{q\ell}$, EI_{qr} = bending stiffness of the crossbeam $GI_{T\ell}$, GI_{Tr} = St. Venant torsional stiffness of the adjacent chords	

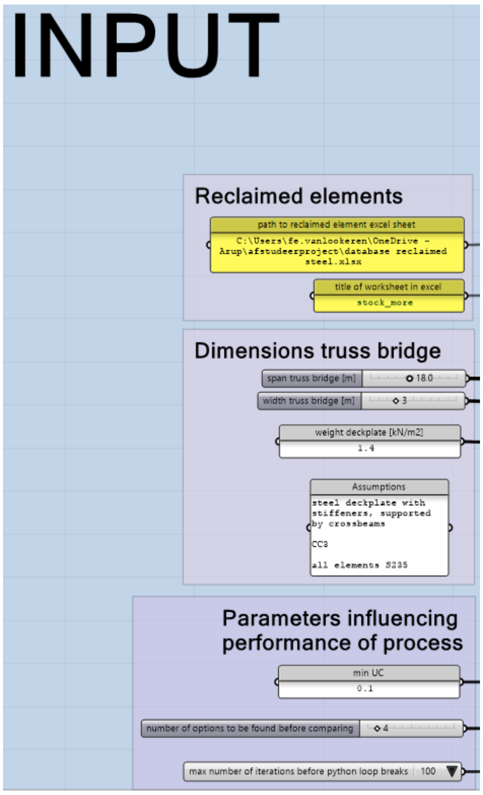
Figure A.1: Table D3 NEN-EN 1993-2 (Normcommissie 351001 ’Technische Grondslagen voor Bouwconstructies’, 2017)

B
Run through of the growth algorithm

A run through of the growth algorithm will be shown in the next figures. It visualizes the growing process of the truss up to the end of the algorithm and subsequently the global checks performed by Karamba. Next comes the manual step where one can choose whether to add the truss design (if it passes all checks) to the solution cloud. The generation of the solution cloud and the subsequent modelling of 3D truss bridge designs is visualized in chapter 7.10.

The first part of the design process, visualized in this appendix, aims to generate a valid 2D truss design. As explained in 6.3, the global lateral stability is in this step provisionally checked by copying the generated truss so that a temporary truss bridge design can be modelled. When a solution cloud of valid trusses is formed (see chapter 7.10.1), two trusses may be selected to model a representative truss bridge design. This step is shown in 7.10.2.

The input parameters of this design are presented in B.1.



path to reclaimed element excel sheet
C:\Users\fe.vanlookeren\OneDrive - Arup\afstudeerproject\database reclaimed steel.xlsx

title of worksheet in excel
stock_more

span truss bridge [m] 18.0

width truss bridge [m] 3

weight deckplate [kN/m2]
1.4

Assumptions
steel deckplate with stiffeners, supported by crossbeams

CC3

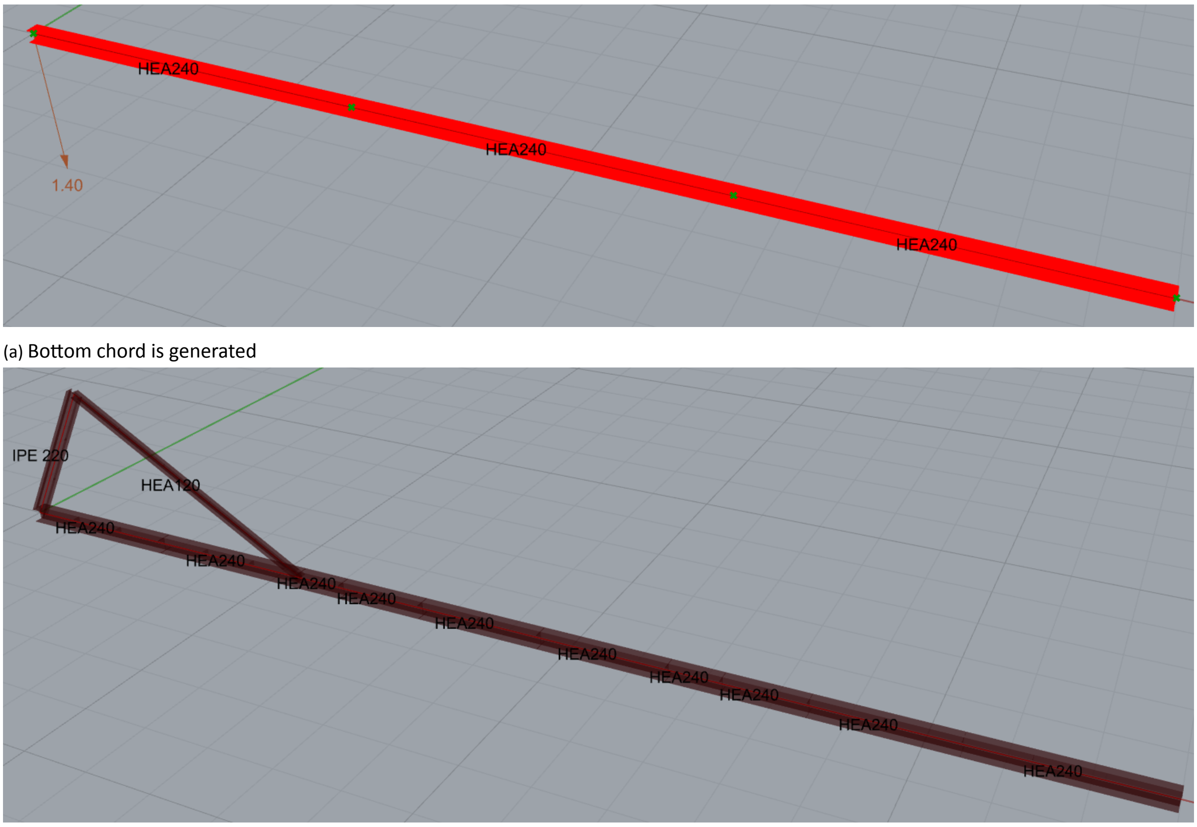
all elements S235

min UC
0.1

number of options to be found before comparing 4

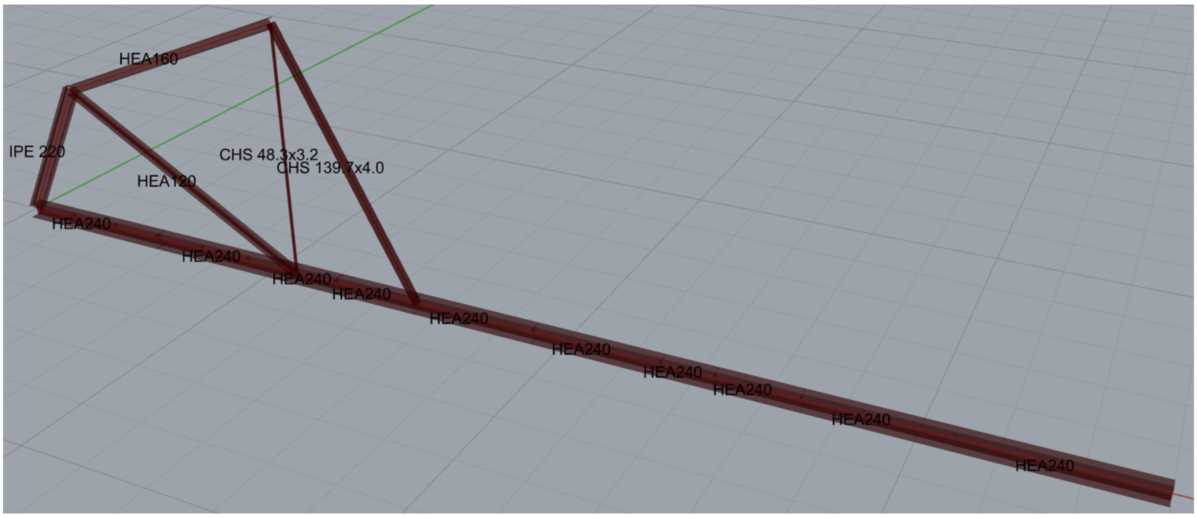
max number of iterations before python loop breaks 100

Figure B.1: Input parameters

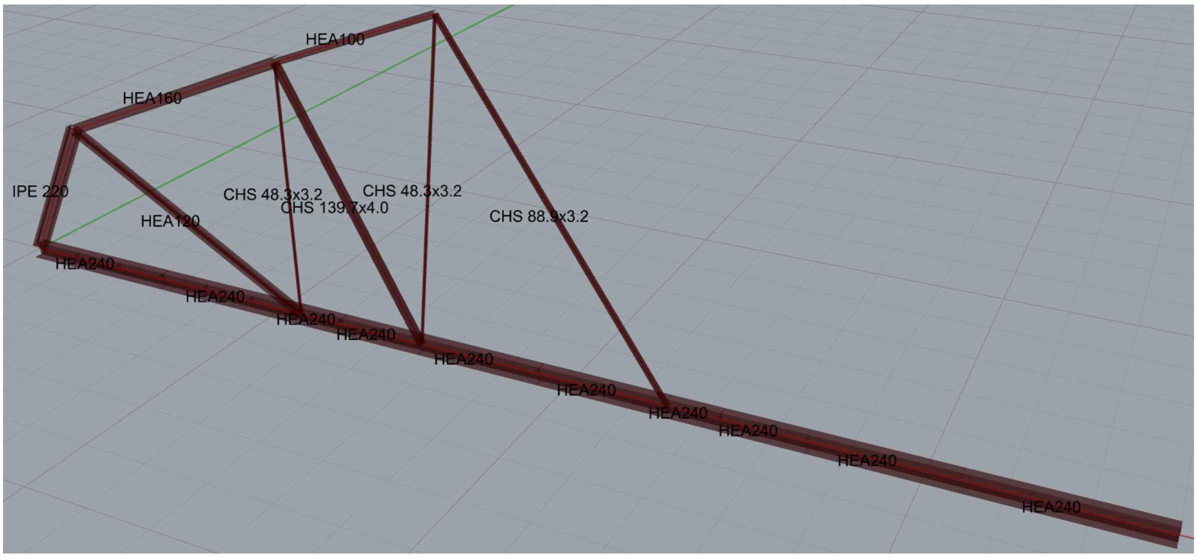


(a) Bottom chord is generated

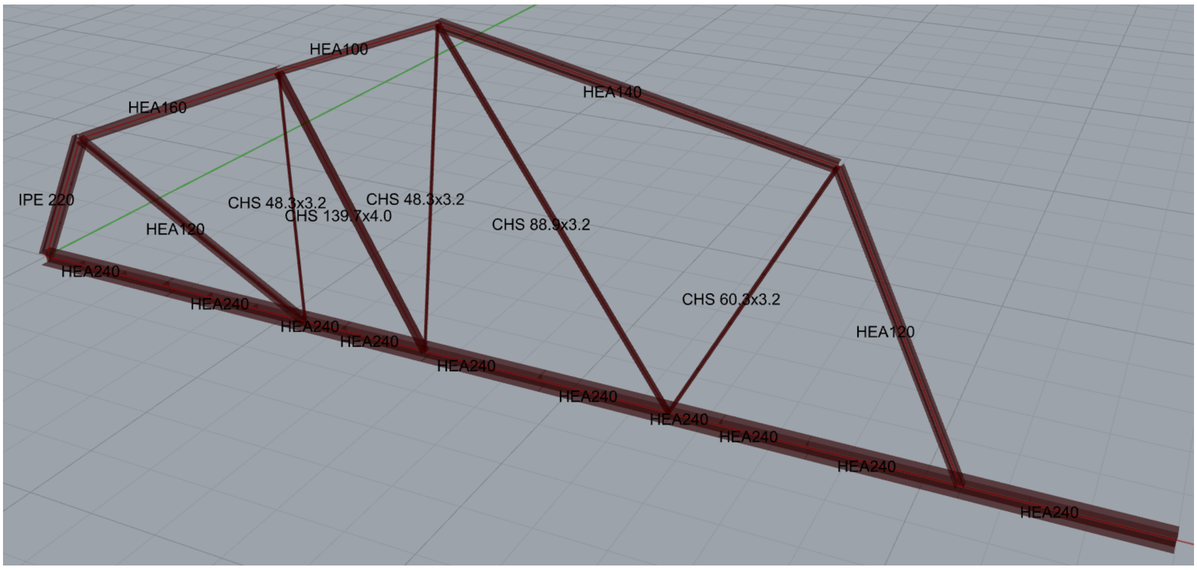
(b) First triangle is generated



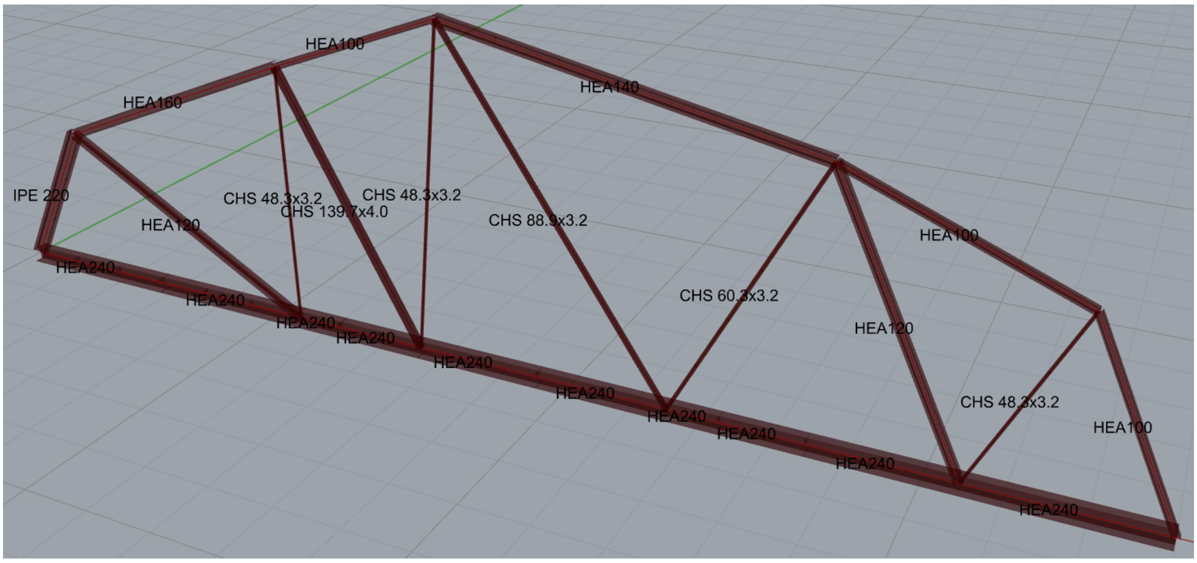
(c) Generating the middle elements



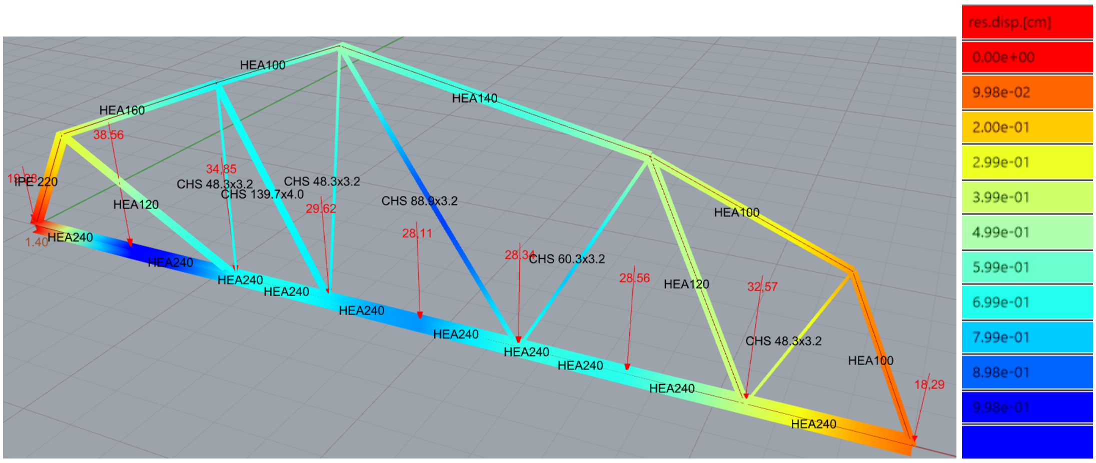
(d) Generating the middle elements



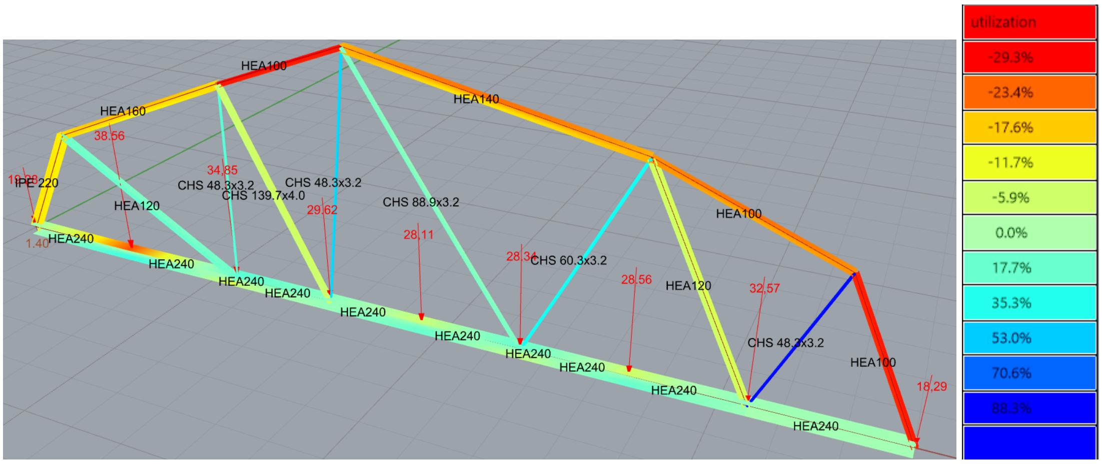
(e) Generating the middle elements



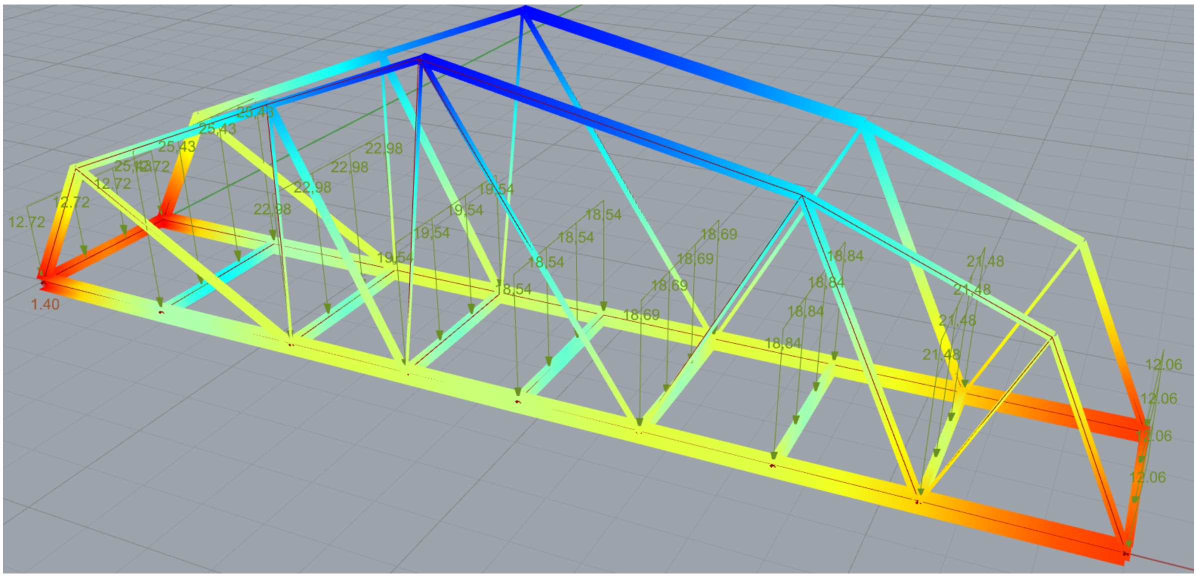
(f) Last triangle is generated



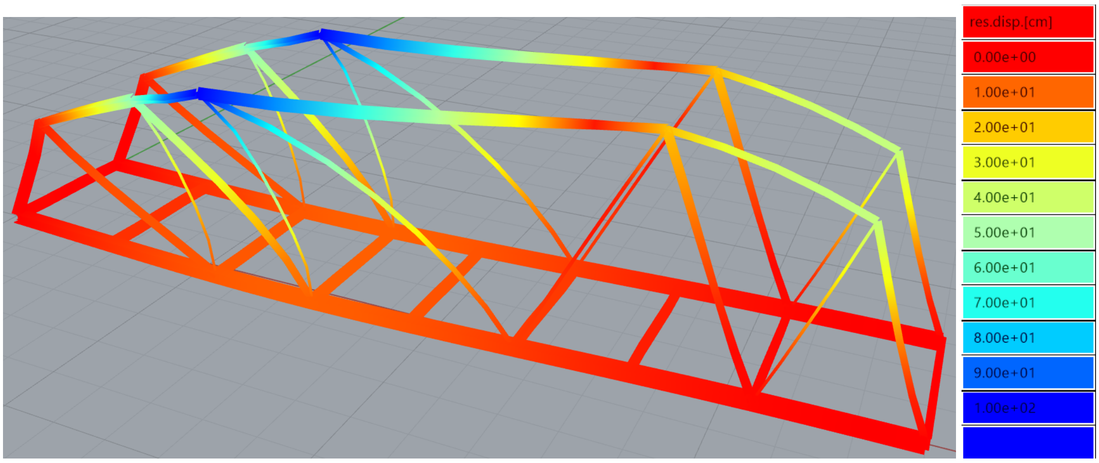
(g) Truss model with pointloads; calculation of displacements



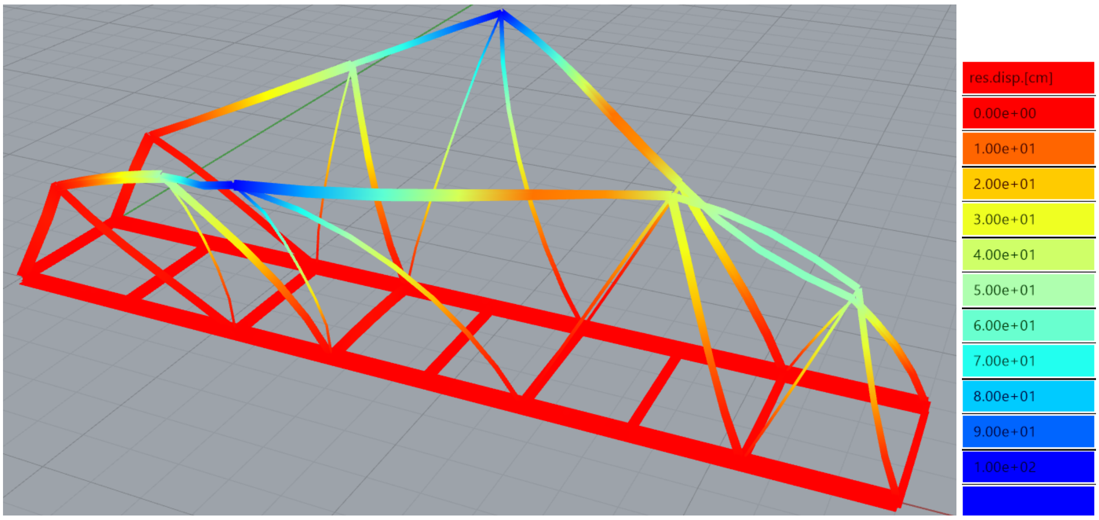
(h) Calculation of unity checks (without element instability)



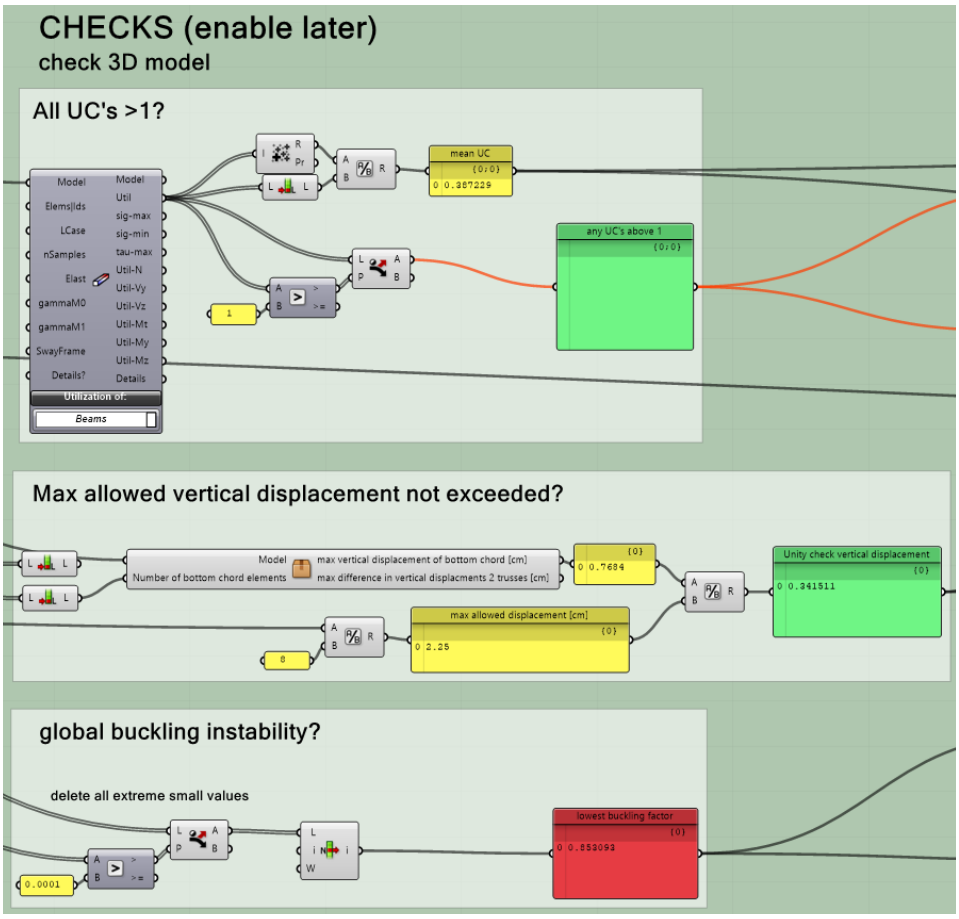
(i) Generating the 3D model with lineloads



(j) First buckling mode with buckling factor=0.85



(k) Second buckling mode with buckling factor=0.88



(l) Performed checks

Figure B.2: Intermediate results growth process

In the shown run it can be seen that the generated truss does not satisfy all checks. The lateral buckling check is coloured red as the lowest buckling factor is below 1. The current truss is thus unstable under the applied loads. This truss will therefore not be inserted into the list of valid truss options. The algorithm may be run again.

C

Python scripts

C.1. Generating the dataframe with reclaimed elements

```python
import rhinoscriptsyntax as rs
import Rhino.Geometry as rg
import scriptcontext as sc
import ghpythonremote as pr
from ghpythonlib import treehelpers as th
import ghpythonlib.components as ghcomp

np = sc.sticky[’numpy’]
sp = sc.sticky[’scipy’]
pd = sc.sticky[’pandas’]
rpy = sc.sticky[’rpy’]


#transform excel input to dataframe
re_list=th.tree_to_list(reclaimed_elements, None)
re=pd.DataFrame({re_list[0][0]: re_list[0][1:],re_list[1][0]: re_list[1][1:], re_list[2][0]: re_list[2][1:], ”A”:A, ”w”:w, ”h”:h})
re[’length’]=pd.to_numeric(re[’length’], errors=’coerce’)
re[’number’]=pd.to_numeric(re[’number’], errors=’coerce’)
print re

#calculate capacity of elements
Aa=np.array(A)
alphaZa=np.array(alphaZ)
Izza=np.array(Izz)
Lcr=np.array(pr.obtain(re.length.tolist()))*100 #cm


fy=23.5 #kN/cm2
E=21000.0 #kN/cm2
gM1=1.15 #because we are considering reclaimed steel

Ft=(fy*Aa).tolist() #with A[cm2], so Ft[kN]

i=(Izza/Aa)**0.5
lambda1=np.pi*(E/fy)**0.5
lambda_dash=Lcr/(i*lambda1)
phi=0.5*(1+alphaZa*(lambda_dash-0.2)+lambda_dash**2)
xi=1/(phi+((phi**2)-(lambda_dash**2))**0.5)
xi=np.array([min(x,1) for x in xi])
Fc=(xi*Aa*fy/gM1).tolist() #kN

#add two columns with capacities
re[’Fc’]=Fc
re[’Ft’]=Ft


#extract list with all profile names, to import into Karamba component
```

```
#Karamba will find the characteristics of these profiles
#A and Izz are then looped back into this component, the component is run a second time and the correct values are now put into the dataframe
profiles_list=pr.obtain(re.profile.tolist())

#dataframe is sorted by 1) profile name and 2)length
re=re.sort_values(by=[”profile”,”length”])
re=re.reset_index(drop=True)

#include all elements as separate elements when number of elements>1
re_all_elements=re.loc[re.index.repeat(re.number)]
re_all_elements_new_indices=re_all_elements.reset_index(drop=True)
re=re_all_elements_new_indices.drop(”number”, axis=1) #delete column with number info

#store dataframe
re.to_pickle(”re1.pkl”)

print re.head(10)
```

C.2. Selecting the bottom chord elements

```
import rhinoscriptsyntax as rs
import Rhino.Geometry as rg
import scriptcontext as sc
import ghpythonremote as pr
from ghpythonlib import treehelpers as th
import ghpythonlib.components as ghcomp

np = sc.sticky[’numpy’]
sp = sc.sticky[’scipy’]
pd = sc.sticky[’pandas’]
rpy = sc.sticky[’rpy’]

#import dataframe
re=pd.read_pickle(”re1.pkl”)    #sorted by profile, then sorted by length
re[’used’] = 0 #add column wherein 0 means not used in truss design and 1 means it is used

if len(used_ids)>1:
    print ’calculating second truss, elements of first truss will not be considered’
    for i in used_ids:
        re.at[int(i), ’used’]=1
else:
    print ’calculating first truss’
#---------------CHOOSE BOTTOM CHORD ELEMENTS ------------------------------------------------------------------------------------------------------------

#first analyse the available cross sections and lengths by
# 1) looking at profiles with cross section equal or bit larger than required
# 2) look if in this range there are profiles with large enough lengths or amount of elements to reach the entire span (case 1)
# 3) if this is not the case, look at larger elements, etc.
# 4) if there exist no single profile which can reach the entire span, weld different profiles onto each other, starting from smallest sufficient profile (case 2)
# 5) if there are no sufficient elements with large enough cross sections, print that no solution is possible (case 3)
# Then add the chosen elements to the element dataframe

#analyse possible profiles, find out which of the three cases we are working with
ranges=np.linspace(1,re.A.max()/A_bc, 4) #creates 4 ranges of A to test out
found = 0 #found=0: not all possibilities yet analysed, found=1: at least 1 profile found which can reach span(case1), found=2: combination of different profiles
#needed to reach span(case2), found=3: no solution possible(case3)
number=0
while found==0:
    options=re[(re.A>ranges[number]*A_bc)&(re.A<ranges[number+1]*A_bc)&(~re.profile.str.contains(”CHS”))&(re.h>=min_height)] #profiles with A in tested range and not considering circular sections
    sum_lengths = options.groupby([”profile”]).length.sum()
    preferred_profiles=pr.obtain(sum_lengths[sum_lengths>span].index.tolist()) #profiles which have sufficient elements to create span
    if len(preferred_profiles) != 0:
        print ’In range number ’, number+1, ’ out of ’,len(ranges),’ at least 1 profile with enough length is found’
        print ’the possible profiles:’,preferred_profiles
        found=1
    else:
        if number<2: #if no profile is found in current range, switch to next range
            number+=1
        else:
            print ’all ranges have been analysed, elements with different profiles will have to be used’
            options=re[(re.A>A_bc) & (~re.profile.str.contains(”CHS”))]
            sum_lengths = options.groupby([”profile”]).length.sum()
            if sum(sum_lengths.tolist()) < span:
                print ’there are not sufficient elements to reach span, lower min A’
                found=3
            else:
                possible_profiles=pr.obtain(options.profile.tolist())
                found=2

#now pick and add elements to bottom chord
n_el=100 #dummy value

if found==1: #when at least 1 profile with sufficient length is found
    chosen_profile=’none yet’
    bc_final=[] # will be the list with element indices of bottom chord elements
    spare_part_final={} #dummy value
    for i in preferred_profiles: #analyse all the selected profiles
        bc=[]
        length=0.0
        counter=1
        while length<span: #add longest elements until span is reached
            rest_length=span-length
            longest_element=re[re.profile==i].index[-counter] #get index of longest element of the specific ’preferred profile’
            if re.at[longest_element,’length’]< rest_length: #if span has not yet been reached, add full element to bottom chord
                bc.append(longest_element)
                length+=re.at[longest_element,’length’]
            else: #if the largest element results in bottom chord larger then required span:
                if len(re[(re.profile==i)&(re.length==rest_length)])!=0: # search for fitting element
                    last_element=re[(re.profile==i)&(re.length==rest_length)].index[0]
                    bc.append(last_element)
                    length+=re.at[last_element,’length’]
                    spare_part=’none’
                else: #if not found keep largest element and cut off spare part
                    spare_part_length=re.at[longest_element,’length’]-rest_length
                    #add the spare part as a new element to the dataframe, with the leftover length
                    spare_part = {’A’: re.at[longest_element,’A’], ’length’: spare_part_length, ’profile’: i, ’Fc’: re.at[longest_element,’Fc’], ’Ft’: re.at[longest_element,’Ft’], ’used’:2, ’w’: re.at[longest_element,’w’], ’h’:re.at[longest_element,’h’] }
                    bc.append(longest_element)
                    length+=rest_length
            counter+=1
        if len(bc)<n_el: #if the considered profile needs less elements to reach the bridge span, this profile will be chosen
            chosen_profile=i
            n_el=len(bc)
            bc_final=bc
            spare_part_final=spare_part
    width_bc=re.at[bc_final[0], ’w’].tolist()
    print ’chosen profile:’, chosen_profile
    print ’used element ids in bottom chord:’, bc_final
    #update dataframes
    if spare_part_final != ’none’:
        print ’last element was cut and spare part is included in database’
        re=re.append(spare_part_final, ignore_index = True)
        re.at[bc_final[-1],’length’]=re.at[bc_final[-1],’length’]-spare_part_final[’length’]
    for i in bc_final:
        re.at[i, ’used’]=1
    print ’changed reclaimed element dataframe:’
    print re[re.profile==chosen_profile]


elif found==2: #different profiles need to be used
    bc_final=[]
    length=0.0
    counter=0
    while length<span: #add smallest (in cross sectional area) elements until span is reached
        rest_length=span-length
        element=options.index[counter] #get index of first (so smallest) element of options
        if re.at[element,’length’]< rest_length: #if span has not yet been reached, add full element to bottom chord
            bc_final.append(element)
            length+=re.at[element,’length’]
        else: #if the element results in bottom chord larger then required span:
            if len(options[options.length==rest_length])!=0: # search for fitting element
                last_element=options[options.length==rest_length].index[0]
                bc_final.append(last_element)
                length+=re.at[last_element,’length’]
                spare_part=’none’
            else: #if not found keep element and cut off spare part
                spare_part_length=re.at[element,’length’]-rest_length
                #add the spare part as a new element to the dataframe, with the leftover length
                spare_part = {’A’: re.at[element,’A’], ’length’: spare_part_length, ’profile’: re.at[element,’profile’], ’Fc’: re.at[element,’Fc’], ’Ft’: re.at[element,’Ft’],  ’used’:2, ’w’: re.at[longest_element,’w’], ’h’:re.at[longest_element,’h’]}
                bc_final.append(element)
                length+=rest_length
        counter+=1
    print ’used element ids in bottom chord:’, bc_final
    width_bc=re.at[bc_final[0], ’w’].tolist()

    #update dataframes
    if spare_part != ’none’:
        print ’last element was cut and spare part is included in database’
        re=re.append(spare_part, ignore_index = True)
        re.at[bc_final[-1],’length’]=re.at[bc_final[-1],’length’]-spare_part[’length’]
    for i in bc_final:
        re.at[i, ’used’]=1
    print re[re.A>A_bc]


else:
    print ’no bottom chord’


#resort reclaimed element dataframe after elements have been picked, cut and spare elements added
re=re.sort_values(by=[”profile”,”length”])

#---------------------------CREATE GEOMETRY------------------------------------------------------------------------------------------

#create element dataframe and include bottom chord elements
ue=pd.DataFrame(columns = [’el_id’,’profile’, ’k_line’])

p1=rs.CreatePoint(0,0,0) #starting point
for i in bc_final:
    p2=rs.CreatePoint(p1.X+re.length[i], 0, 0)
    karamba_line=rg.Line(p1,p2)
    ue = ue.append({’el_id’:i,’profile’:re.at[i,’profile’], ’k_line’:karamba_line}, ignore_index = True)
    p1=p2

print ’used elements dataframe:’
print ue

```

```

#-------------------------------------OUTPUT -----------------------------------------------------------------------------------------
#position supports
supports=[rs.CreatePoint(0, 0, 0), rs.CreatePoint(span, 0, 0)]

#output lists (from dataframes) for visualization
lines=pr.obtain(ue.k_line.tolist())
profiles=pr.obtain(ue.profile.tolist())

#updated dataframes
re.to_pickle(”re2.pkl”)
ue=ue.drop(columns=[’k_line’]) #the geometry data isn’t stored correctly in pickle file so these need to be transferred separately to next python component
ue.to_pickle(”ue2.pkl”)
```

C.3. Selecting the elements of the first triangle

```
import rhinoscriptsyntax as rs
import Rhino.Geometry as rg
import scriptcontext as sc
import ghpythonremote as pr
from ghpythonlib import treehelpers as th
import ghpythonlib.components as ghcomp
import rpyc

np = sc.sticky[’numpy’]
sp = sc.sticky[’scipy’]
pd = sc.sticky[’pandas’]
rpy = sc.sticky[’rpy’]

#import dataframes
re=pd.read_pickle(”re2.pkl”) #sorted by profile, then sorted by length
ue=pd.read_pickle(”ue2.pkl”)
ue[”k_line”]=lines
print ue

print ’percentage of available elements with width lower than width bottom chord: ’, 100* len(re[(re.used==0) & (re.w <= width_bc)]) / len(re[re.used==0]), ’%’

#-------------------------------------ALGORITHM ----------------------------------------------------------------
if any(”RHS” in p for p in ue.profile):#check if bottom chord consists out of hollow sections, if so, extra width constraint on diagonals
    rhs_bc_i = [i for i in ue.index.values if ”RHS” in ue.profile[i]] #ue dataframe index values of RHS profiles in bottom chord
    rhs_bc_id = [ue.el_id[i] for i in rhs_bc_i] #element id’s of RHS profiles in bottom chord
    widths=[re.w[i] for i in rhs_bc_id]
    min_width=min(widths) / 2
else:
    min_width=0

options=[] #new valid options of elements are appended to the list until a certain amount of valid options is found
options_lines=[] #store their geometry
options_UC=[] #store their unity checks
options_N=[] #store their axial forces
counter=0
while len(options)<n_options: #define here the number of valid options that need to be found before comparing them to find the best option
    #pick 2 random elements
    first_elements=re[(re.used!=1) & (re.w <= width_bc) & (re.w >= min_width)].sample(n=2) # width needs to be smaller than width of bottom chord element so that connection is possible
    re_id1, re_id2=first_elements.index.tolist()
    l1,l2=first_elements.length.tolist()

    #position them so that the mean angle is equal to 45 degrees/so sum of the angles is equal to 90 degrees, for explanation of the equations, see report
    alpha_sum=np.radians(90)
    alpha2=np.arctan(np.sin(alpha_sum)/(np.cos(alpha_sum)+l2/l1))
    alpha1=np.arcsin(np.sin(alpha2)*l2/l1)
    alpha1d=np.degrees(alpha1)
    alpha2d=np.degrees(alpha2)

    p1=rs.CreatePoint(0,0,0)
    p2=rs.CreatePoint(np.cos(alpha1)*l1,0,np.sin(alpha1)*l1)
    p3=rs.CreatePoint(np.cos(alpha1)*l1+np.cos(alpha2)*l2,0,0)
    line1=rg.Line(p1,p2)
    line2=rg.Line(p2,p3)
    lines_triangle=[line1,line2]

    #calculate force in first diagonal with node equilibrium
    Rf= -q*span/2 #left reaction force = half the total ULS load
    F1= q*(p3.X)/2#point load on first node
    N1=(-Rf-F1)/np.sin(alpha1)

    #to calculate force in second element, a top chord element is assumed so that cut can be made after point 2
    #third element is assumed to be horizontal, with decision that added elements may only be horizontal or directing upwards until half of the bridge span
    #and the other way around after the middle (horizontal bar then most conservative option that will create largest force in element 2)
    #cut is made halfway between point 2 and point 3
    x_cut=(p2.X+p3.X)/2
    #first moment equilibrium around point 2 to calculate force in bottom chord
    Nbc=(Rf+F1)*p2.X/p2.Z
    #then moment equilibrium around intersection cut and assumed horizontal third element
    N2=((Rf+F1)*x_cut-Nbc*p2.Z)/(np.sin(alpha2)*(x_cut-p2.X))

    #check capacities of chosen elements
    UC1=N1/-re.Fc[re_id1]
    UC2=N2/re.Ft[re_id2]
    if np.all([UC1<0.9,UC1>min_UC,UC2<0.9,UC2>min_UC]): #if they fall into the allowed UC range, include the option in the list of valid options
        options.append([re_id1,re_id2])
        options_lines.append(lines_triangle)
        options_UC.append([UC1,UC2])
        options_N.append([N1,N2])

    counter+=1
    if counter==500:
        print ’loop forced to stop after’, counter, ’iterations’
        break


print ’After’, counter, ’iterations’, len(options), ’options were found for the first triangle’

#analyze the options
sum_UC=[]
for i in options_UC:
    sum_UC.append(i[0]+i[1])

best_option=sum_UC.index(max(sum_UC)) #best option is the option with the highest unity checks, being the most efficient option
best_option_ids=options[best_option]
best_option_lines=options_lines[best_option]
print ’element ids of the best option:’, best_option_ids

#-------------------------------UPDATE DATAFRAMES---------------------------------------------------------
for i in best_option_ids:
    re.at[i, ’used’]=1

for i in range(len(best_option_ids)):
    ue = ue.append({’el_id’:best_option_ids[i],’profile’:re.at[best_option_ids[i],’profile’],’k_line’:best_option_lines[i]},ignore_index = True)

print ’updated used element dataframe:’
print ue
```

```


#-------------------------------------OUTPUT--------------------------------------------------------------------
#output lists to visualize
lines=pr.obtain(ue.k_line.tolist())
profiles=pr.obtain(ue.profile.tolist())

#output for next step
end_node=ue.loc[ue.index[-1], ’k_line’].To
top_node=ue.loc[ue.index[-1], ’k_line’].From
F=[q*(end_node.X)/2]
Fx=[0]
N_best=options_N[best_option]
N=[float(i) for i in N_best]
UC_best=options_UC[best_option]
UC=[float(i) for i in UC_best]

#updated dataframes
re.to_pickle(”re3.pkl”)
ue=ue.drop(columns=[’k_line’])
ue.to_pickle(”ue3.pkl”)
```

C.4. Selecting the elements of the middle triangles

```
import rhinoscriptsyntax as rs
import Rhino.Geometry as rg
import scriptcontext as sc
import ghpythonremote as pr
from ghpythonlib import treehelpers as th
import ghpythonlib.components as ghcomp


np = sc.sticky[’numpy’]
sp = sc.sticky[’scipy’]
pd = sc.sticky[’pandas’]
rpy = sc.sticky[’rpy’]

#import dataframes
re=pd.read_pickle(”re3.pkl”) #sorted by profile, then sorted by length
ue=pd.read_pickle(”ue3.pkl”)
ue[”k_line”]=lines

UC1_list=[]
UC2_list=[]
UC3_list=[]

print ’percentage of available elements with width lower than width bottom chord: ’, 100* len(re[(re.used==0) & (re.w <= width_bc)]) / len(re[re.used==0]), ’%’

#---------------------------------ALGORITHM------------------------------------------------------------------------------------
options=[] #new valid options of elements are appended to the list until a certain amount of valid options is found
options_lines=[] #store their geometry
options_UC=[] #store their unity checks
options_F=[] #store the last added pointload, which depends on position of point 4
options_N=[] #store the calculated normalforces
counter=0
if len(N)==2: #for the very first iteration, the N input originates from the first triangle, which has ony 2 elements.
    N1=N[0]
    N2=N[0]
    N3=N[1]
else:
    N1=N[0]
    N2=N[1]
    N3=N[2]

while len(options)<n_options: #define here the number of valid options that need to be found
```

```
before comparing them to find the best option
print 'iteration ', counter
counter+=1
if counter==max_iterations:
    print 'loop forced to stop after',counter,' iterations'
    break
print 'N1,N2,N3', N1, N2, N3

#PICK ELEMENTS AND FIND GEOMETRY
#starting nodes
p1=top_node
p2=start_node
dist=p2.X-p1.X #horizontal distance between nodes, needed for constrain on lengths to be picked elements

#constraint on length 1rst element
l1_min=1.1*dist

#pick 1rst random element, acknowledging constraints
if N1<0:
    try:
        el1=re[(re.used!=1) & (re.length>l1_min) & (re.Fc<-F_range[0]*N1) & (re.Fc>-F_range[1]*N1)].sample(n=1)
        print len(re[(re.used!=1) & (re.length>l1_min) & (re.Fc<-F_range[0]*N1) & (re.Fc>-F_range[1]*N1)]), ' options to choose from, when applying all constraints'
    except:
        print 'too small capacity range, capacity constraint is left out for first element'
        el1=re[(re.used!=1) & (re.length>l1_min)].sample(n=1)
else:
    try:
        el1=re[(re.used!=1) & (re.length>l1_min) & (re.Ft<F_range[0]*N1) & (re.Ft>F_range[1]*N1)].sample(n=1)
        print len(re[(re.used!=1) & (re.length>l1_min) & (re.Ft<F_range[0]*N1) & (re.Ft>F_range[1]*N1)]), ' options to choose from, when applying all constraints'
    except:
        print 'too small capacity range, capacity constraint is left out for first element'
        try:
            el1=re[(re.used!=1) & (re.length>l1_min)].sample(n=1)
        except:
            print 'no first element can be found'
            continue
id1=el1.index.tolist()[0]
l1=el1.length.tolist()[0]

#constraint on length 2nd element
if p2.X < span/2: # first half of the truss, different constraints
    l2_min=np.sqrt((p1.Z)**2 + (l1-dist)**2)
    l2_max=p1.Z+np.sqrt(l1**2 - dist**2)
else: #second half of the truss
    print 'second half of truss'
    l2_min=max(0.2*p1.Z, p1.Z-np.sqrt(l1**2 - dist**2))
    l2_max=np.sqrt((p1.Z)**2 + (l1-dist)**2)
#pick 2nd random element, acknowledging constraints
if N2<0:
    try:
        el2=re.drop([id1], axis=0)[(re.used!=1) & (re.length>l2_min) & (re.length<l2_max) & (re.Fc<-F_range[0]*N2) & (re.Fc>-F_range[1]*N2) & (re.w <= width_bc) & (re.w >= min_width)].sample(n=1)
        print len(re.drop([id1], axis=0)[(re.used!=1) & (re.length>l2_min) & (re.length<l2_max) & (re.Fc<-F_range[0]*N2) & (re.Fc>-F_range[1]*N2) & (re.w <= width_bc) & (re.w >= min_width)]), ' options to choose from, when applying all constraints'
    except:
        print 'too small capacity range, capacity constraint is left out for second element'
        try:
            el2=re.drop([id1], axis=0)[(re.used!=1) & (re.length>l2_min) & (re.length<l2_max) & (re.w <= width_bc) & (re.w >= min_width)].sample(n=1)
        except:
            print 'no second element can be found. length range: ', l2_min, l2_max
else:
    try:
        el2=re.drop([id1], axis=0)[(re.used!=1) & (re.length>l2_min) & (re.length<l2_max) & (re.Ft<F_range[0]*N2) & (re.Ft>F_range[1]*N2) & (re.w <= width_bc) & (re.w >= min_width)].sample(n=1)
        print len (el2=re.drop([id1], axis=0)[(re.used!=1) & (re.length>l2_min) & (re.length<l2_max) & (re.Ft<F_range[0]*N2) & (re.Ft>F_range[1]*N2) & (re.w <= width_bc) & (re.w >= min_width)]), ’ options to choose from, when applying all constraints’
    except:
        print ’too small capacity range, capacity constraint is left out for second element’
        try:
            el2=re.drop([id1], axis=0)[(re.used!=1) & (re.length>l2_min) & (re.length<l2_max) & (re.w <= width_bc) & (re.w >= min_width)].sample(n=1)
        except:
            print ’no second element can be found. length range: ’, l2_min, l2_max
            continue
id2=el2.index.tolist()[0]
l2=el2.length.tolist()[0]

#find position of nodes
#create circles to find intersetion point of 2 elements
c1=rg.Circle(rg.Plane(p1, rg.Vector3d(0,1,0)), p1, l1) #using length element 1 as radius
c2=rg.Circle(rg.Plane(p2, rg.Vector3d(0,1,0)), p2, l2) #using length element 2 as radius
its=rg.Intersect.Intersection.CircleCircle(c1,c2) #find intersections of two circles
if its[1].X > its[2].X: #2 intersecting points: choose point with highest x-coordinate
    p3=rs.CreatePoint(its[1])
else:
    p3=rs.CreatePoint(its[2])
line1=rg.Line(p1,p3)
line2=rg.Line(p2,p3)

#pick 3rd random element
if N3<0:
    try:
        el3=re.drop([id1, id2], axis=0)[(re.used!=1) & (re.length>p3.Z) & (re.Fc<-F_range[0]*N3) & (re.Fc>-F_range[1]*N3) & (re.w <= width_bc) & (re.w >= min_width)].sample(n=1)
        print len(re.drop([id1, id2], axis=0)[(re.used!=1) & (re.length>p3.Z) & (re.Fc<-F_range[0]*N3) & (re.Fc>-F_range[1]*N3) & (re.w <= width_bc) & (re.w >= min_width)]), ’ options to choose from, when applying all constraints’
    except:
        print ’too small capacity range, capacity constraint is left out for third element’
        try:
            el3=re.drop([id1, id2], axis=0)[(re.used!=1) & (re.length>p3.Z) & (re.w <= width_bc) & (re.w >= min_width)].sample(n=1)
        except:
            print ’no third element can be found’
            continue
else:
    try:
        el3=re.drop([id1, id2], axis=0)[(re.used!=1) & (re.length>p3.Z) & (re.Ft<F_range[0]*N3) & (re.Ft>F_range[1]*N3) & (re.w <= width_bc) & (re.w >= min_width)].sample(n=1)
        print len(re.drop([id1, id2], axis=0)[(re.used!=1) & (re.length>p3.Z) & (re.Ft<F_range[0]*N3) & (re.Ft>F_range[1]*N3) & (re.w <= width_bc) & (re.w >= min_width)]), ’ options to choose from, when applying all constraints’
    except:
        print ’too small capacity range, capacity constraint is left out for third element’
        try:
            el3=re.drop([id1, id2], axis=0)[(re.used!=1) & (re.length>p3.Z) & (re.w <= width_bc) & (re.w >= min_width)].sample(n=1)
        except:
            print ’no third element can be found’
            continue
#and which has sufficient length to reach the bottom chord
id3=el3.index.tolist()[0]
l3=el3.length.tolist()[0]

#find intersection with bottom chord
bottom_chord=rg.Line(rs.CreatePoint(0,0,0), rs.CreatePoint(span,0,0))
c3=rg.Circle(rg.Plane(p3, rg.Vector3d(0,1,0)), p3, l3)
its=rg.Intersect.Intersection.LineCircle(bottom_chord,c3) #find intersections of the third element and the bottom chord
if its[2].X > its[4].X: #2 intersecting points: choose point with highest x-coordinate
    p4=rs.CreatePoint(its[2])
else:
    p4=rs.CreatePoint(its[4])

if (p4.X>span-(p4.X-p2.X)*0.5):
    print ’point 4 comes to close to end bottom chord for last triangle to fit in’
    continue
else:
    line3=rg.Line(p3,p4)

#lines=[line1,line2,line3]
#print lines

#CALCULATE FORCES WITH METHOD OF SECTION
Rf=-q*span/2

#CUT1, moment equilibrium around point2
#moment due to pointloads and reactionforce
M1=Rf*p2.X
for i in range(len(F_input)):
    M1+=F_input[i]*(p2.X-Fx_input[i])

delta_x=p2.X-(p1.X+p2.X)/2 #distance between cut and point1 or point2
alpha1=np.arctan((p3.Z-p1.Z)/(p3.X-p1.X))
N1=-M1/(np.sin(alpha1)*delta_x+np.cos(alpha1)*(p1.Z+(delta_x/(p3.X-p1.X))*(p3.Z-p1.Z)))

#CUT2, moment equilibrium around intersection cut and bottom chord
x_cut=(p2.X+p3.X)/2
#extra pointload
new_F=q*(p4.X-Fx_input[-1])/2
F=F_input+[new_F]
Fx=Fx_input+[p2.X]
#moment due to pointloads and reactionforce
M2=Rf*x_cut
for i in range(len(F)):
    M2+=F[i]*(x_cut-Fx[i])

alpha2=np.arctan(p3.Z/(p3.X-p2.X))
N2=(-M2-N1*np.cos(alpha1)*(p1.Z+((x_cut-p1.X)/(p3.X-p1.X))*(p3.Z-p1.Z)))/(np.sin(alpha2)*(x_cut-p2.X))

#CUT3, moment equilibrium around point 3 to calculate Nbc, then moment equilibrium around intersection point of cut and assumed extra horizontal member
#to calculate force in third element, a fourth element acting on top node is assumed so that cut can be made after point3
#third element is assumed to be horizontal, with decision that added elements may only be horizontal or directing upwards until half of the bridge span
#and the other way around after the middle (horizontal bar then most conservative option that will create largest force in element 2)
alpha3=np.arctan(p3.Z/(p4.X-p3.X))
M3=Rf*p3.X
for i in range(len(F)):
    M3+=F[i]*(p3.X-Fx[i])
Nbc=M3/p3.Z

x_cut=(p3.X+p4.X)/2 #cut is made halfway between point 3 and point 4
M4=Rf*x_cut
for i in range(len(F)):
    M4+=F[i]*(x_cut-Fx[i])
N3=(M4-Nbc*p3.Z)/(np.sin(alpha3)*(x_cut-p3.X))

#CHECK CAPACITIES OF CHOSEN ELEMENTS
if N1>0:
    UC1=N1/re.Ft[id1]
else:
    UC1=N1/-re.Fc[id1]
if N2>0:
    UC2=N2/re.Ft[id2]
else:
    UC2=N2/-re.Fc[id2]

if N3>0:
    UC3=N3/re.Ft[id3]
else:
    UC3=N3/-re.Fc[id3]

print 'UCs: ', UC1, UC2, UC3
UC1_list.append(UC1)
UC2_list.append(UC2)
UC3_list.append(UC3)
if np.all([UC1<0.9,UC1>min_UC,UC2<0.9,UC2>min_UC,UC3<0.9,UC3>min_UC]): #if they fall into the allowed UC range, include the option in the list of valid options
    options.append([id1,id2,id3])
    options_lines.append([line1,line2,line3])
    options_UC.append([UC1,UC2,UC3])
    options_F.append(new_F)
    options_N.append([N1,N2,N3])

low1 = [i for i in UC1_list if i < 0.2]
high1 = [i for i in UC1_list if i > 1.0]
low2 = [i for i in UC2_list if i < 0.2]
high2 = [i for i in UC2_list if i > 1.0]
low3 = [i for i in UC3_list if i < 0.2]
high3 = [i for i in UC3_list if i > 1.0]
print 'percentage of UCs of element 1 below ', min_UC, ' : ', "{:.2f}".format(100*len(low1)/len(UC1_list)), '%, and above 1.0: ', "{:.2f}".format(100*len(high1)/len(UC1_list)), '%'
print 'percentage of UCs of element 2 below ', min_UC, ' : ', "{:.2f}".format(100*len(low2)/len(UC2_list)), '% , and above 1.0: ', "{:.2f}".format(100*len(high2)/len(UC2_list)), '%'
print 'percentage of UCs of element 3 below ', min_UC, ' : ', "{:.2f}".format(100*len(low3)/len(UC3_list)), '%: , and above 1.0: ', "{:.2f}".format(100*len(high3)/len(UC3_list)), '%'
print 'After', counter-1, 'iterations', len(options), 'options were found for the current triangles'

#analyze the options
sum_UC=[]
for i in options_UC:
sum_UC.append(i[0]+i[1])

best_option=sum_UC.index(max(sum_UC)) #best option is the option with the highest unity checks, being the most efficient option
best_option_ids=options[best_option]
best_option_lines=options_lines[best_option]
best_option_F=options_F[best_option]
print 'element ids of the best option:', best_option_ids

#--------------------------------UPDATE DATAFRAMES---------------------------------------------------------

for i in best_option_ids:
re.at[i, 'used']=1

for i in range(len(best_option_ids)):
ue = ue.append({'el_id':best_option_ids[i],'profile':re.at[best_option_ids[i],'profile'], 'k_line':best_option_lines[i]},ignore_index = True)

print 'updated used element dataframe:'
print ue

#-------------------------------------OUTPUT--------------------------------------------------------------------
#output lists for visualization
lines=pr.obtain(ue.k_line.tolist())
profiles=pr.obtain(ue.profile.tolist())

#output for next iteration
end_node=ue.loc[ue.index[-1], 'k_line'].To
new_top_node=ue.loc[ue.index[-1], 'k_line'].From
```

```
F_output=F_input+[best_option_F]
Fx_output=Fx_input+[p2.X]

distances=[Fx_output[i+1]-Fx_output[i] for i in range(len(Fx_output)-1)]
mean_distance=np.mean(distances).tolist() #needed to break loop when near the end

#output for analysis of results
N_best=options_N[best_option]
for i in N_best:
    N.append(float(i))
UC_best=options_UC[best_option]
for i in UC_best:
    UC.append(float(i))

#updated dataframes
re.to_pickle(”re3.pkl”)
ue=ue.drop(columns=[’k_line’])
ue.to_pickle(”ue3.pkl”)
```

C.5. Selecting the elements of the last triangle

```
import rhinoscriptsyntax as rs
import Rhino.Geometry as rg
import scriptcontext as sc
import ghpythonremote as pr
from ghpythonlib import treehelpers as th
import ghpythonlib.components as ghcomp

np = sc.sticky[’numpy’]
sp = sc.sticky[’scipy’]
pd = sc.sticky[’pandas’]
rpy = sc.sticky[’rpy’]

#import dataframes
re=pd.read_pickle(”re3.pkl”) #sorted by profile, then sorted by length
ue=pd.read_pickle(”ue3.pkl”)
ue[”k_line”]=lines

print ’percentage of available elements with width lower than width bottom chord: ’, 100* len(re[(re.used==0) & (re.w <= width_bc)]) / len(re[re.used==0]), ’%’

#---------------------------------ALGORITHM-----------------------------------------------------------------------------------
options=[] #new valid options of elements are appended to the list until a certain amount of valid options is found
options_lines=[] #store their geometry
options_UC=[] #store their unity checks
options_F=[] #store the last added pointload, which depends on position of point 4
options_N=[] #store the calculated normalforces
counter=0

N1=N[-3] #before N’s can be calculated in iterations below, take N’s of last three elements
N2=N[-2]
N3=N[-1]
#[2.7,0.01]
F_range=[2.7,0.01] #everytime new elements are picked, they require having a capcity within 0,3*N-3*N, with N being the last normalforce calculated for that element

while len(options)<n_options: #define here the number of valid options that need to be found before comparing them to find the best option
    counter+=1
    if counter==max_iterations:
        print ’loop forced to stop after ’, counter, ’ iterations’
        break

    #PICK ELEMENTS AND FIND GEOMETRY
    #starting nodes
    p1=top_node
    p2=start_node
    dist=p2.X-p1.X #horizontal distance between nodes, needed for constrain on lengths to be
    picked elements

    #constraint on length 1rst element
    l1_min=1.1*dist
    l1_max=0.9*(span-p1.X)
    #pick 1rst random element, acknowledging constraints
    if N1<0:
        try:
            el1=re[(re.used!=1) & (re.length>l1_min) & (re.length<l1_max) & (re.Fc<-F_range
    [0]*N1) & (re.Fc>-F_range[1]*N1)].sample(n=1)
        except:
            print ’too small capacity range, capacity constraint is left out’
            el1=re[(re.used!=1) & (re.length>l1_min) & (re.length<l1_max)].sample(n=1)
    else:
        try:
            el1=re[(re.used!=1) & (re.length>l1_min) & (re.Ft<F_range[0]*N1) & (re.Ft>F_range
    [1]*N1)].sample(n=1)
        except:
            print ’too small capacity range, capacity constraint is left out’
            el1=re[(re.used!=1) & (re.length>l1_min)].sample(n=1)
    id1=el1.index.tolist()[0]
    l1=el1.length.tolist()[0]

    #constraint on length 2nd element
    l2_min=max(0.2*p1.Z, p1.Z-np.sqrt(l1**2 - dist**2))
    l2_max=np.sqrt((p1.Z)**2 + (l1-dist)**2)
    #pick 2nd random element, acknowledging constraints
    if N2<0:
        try:
            el2=re.drop([id1], axis=0)[(re.used!=1) & (re.length>l2_min) & (re.length<l2_max)
     & (re.Fc<-F_range[0]*N2) & (re.Fc>-F_range[1]*N2) & (re.w <= width_bc) & (re.w >=
    min_width)].sample(n=1)
        except:
            print ’too small capacity range, capacity constraint is left out’
            el2=re.drop([id1], axis=0)[(re.used!=1) & (re.length>l2_min) & (re.length<l2_max)
     & (re.w <= width_bc) & (re.w >= min_width)].sample(n=1)
    else:
        try:
            el2=re.drop([id1], axis=0)[(re.used!=1) & (re.length>l2_min) & (re.length<l2_max)
     & (re.Ft<F_range[0]*N2) & (re.Ft>F_range[1]*N2) & (re.w <= width_bc) & (re.w >=
    min_width)].sample(n=1)
        except:
            print ’too small capacity range, capacity constraint is left out’
            el2=re.drop([id1], axis=0)[(re.used!=1) & (re.length>l2_min) & (re.length<l2_max)
     & (re.w <= width_bc) & (re.w >= min_width)].sample(n=1)
    id2=el2.index.tolist()[0]
    l2=el2.length.tolist()[0]

    #find position of nodes
    #create circles to find intersetion point of 2 elements
    c1=rg.Circle(rg.Plane(p1, rg.Vector3d(0,1,0)), p1, l1) #using length element 1 as radius
    c2=rg.Circle(rg.Plane(p2, rg.Vector3d(0,1,0)), p2, l2) #using length element 2 as radius
    its=rg.Intersect.Intersection.CircleCircle(c1,c2) #find intersections of two circles
    if its[1].X > its[2].X: #2 intersecting points: choose point with highest x-coordinate
        p3=rs.CreatePoint(its[1])
    else:
        p3=rs.CreatePoint(its[2])
    line1=rg.Line(p1,p3)
    line2=rg.Line(p2,p3)

    #create geometry third element, to be picked after calculation of its required capacity
    p4=rs.CreatePoint(span,0,0)
    l3=ghcomp.Distance(p3,p4) #left distance between last top node and end of span
    line3=rg.Line(p3,p4)

    #CALCULATE FORCES WITH SNEDEMETHODE
    Rf=-q*span/2

    #CUT1, moment equilibrium around point2
    #moment due to pointloads and reactionforce
    M1=Rf*p2.X
    for i in range(len(F_input)):
        M1+=F_input[i]*(p2.X-Fx_input[i])

    delta_x=p2.X-(p1.X+p2.X)/2 #distance between cut and point1 or point2
    alpha1=np.arctan((p3.Z-p1.Z)/(p3.X-p1.X))
    N1=-M1/(np.sin(alpha1)*delta_x+np.cos(alpha1)*(p1.Z+(delta_x/(p3.X-p1.X))*(p3.Z-p1.Z)))

    #CUT2, moment equilibrium around intersection cut and bottom chord
    x_cut=(p2.X+p3.X)/2
    #extra pointload
    new_F=q*(p4.X-Fx_input[-1])/2
    F=F_input+[new_F]
    Fx=Fx_input+[p2.X]
    #moment due to pointloads and reactionforce
    M2=Rf*x_cut
    for i in range(len(F)):
        M2+=F[i]*(x_cut-Fx[i])

    alpha2=np.arctan(p3.Z/(p3.X-p2.X))
    N2=(-M2-N1*np.cos(alpha1)*(p1.Z+((x_cut-p1.X)/(p3.X-p1.X))*(p3.Z-p1.Z)))/(np.sin(alpha2)*(x_cut-p2.X))


    #NODE EQUILIBRIUM, to find force in last element
    last_F=q*(p4.X-p2.X)/2
    F.append(last_F)
    Fx.append(p4.X)
    alpha3=np.arctan(p3.Z/(p4.X-p3.X))
    N3=(-Rf-last_F)/np.sin(alpha3)

    #CHECK CAPACITIES OF CHOSEN ELEMENTS
    if N1>0:
        UC1=N1/re.Ft[id1]
    else:
        UC1=N1/-re.Fc[id1]

    if N2>0:
        UC2=N2/re.Ft[id2]
    else:
        UC2=N2/-re.Fc[id2]

    print 'Ns: ', N1, N2
    print 'UCs: ', UC1, UC2
    #pick third element: element with long enough length, and capacity closest to N3
    if N3<0:
        try:
            options_el3=re.drop([id1, id2], axis=0)[(re.used!=1) & (re.length>l3) & (re.Fc>-1.1*N3) & (re.w <= width_bc) & (re.w >= min_width)]
            options_sorted=options_el3.sort_values(by=['Fc'])
            el3=options_sorted.iloc[0]
            id3=options_sorted.index[0]
            l3=el3.length
            UC3=N3/-re.Fc[id3] #important to note that real UC will differ as element now has different length than initial length for which Fc was calculated, real UC will be smaller
        except:
            print 'no option found for the third element'
            continue
    else:
        try:
            options_el3=re.drop([id1, id2], axis=0)[(re.used!=1) & (re.length>l3) & (re.Ft>N3) & (re.w <= width_bc) & (re.w >= min_width)]
            options_sorted=options_el3.sort_values(by=['Ft'])
            el3=options_sorted.iloc[0]
            id3=options_sorted.index[0]
            l3=el3.length
            UC3=N3/re.Ft[id3]
        except:
            print 'no option found for the third element'
            continue
    print 'UC3: ', UC3
    if np.all([UC1<0.8,UC2<0.9,UC3<0.9, UC1>min_UC,UC2>min_UC,UC3>min_UC]): #if they fall into the allowed UC range, include the option in the list of valid options
        options.append([id1,id2,id3])
        options_lines.append([line1,line2,line3])
        options_UC.append([UC1,UC2,UC3])
        options_F.append([new_F, last_F])
        options_N.append([N1,N2,N3])


print ’After’, counter-1, ’iterations’, len(options), ’options were found for the last triangle’

#analyze the options
sum_UC=[]
for i in options_UC:
    sum_UC.append(i[0]+i[1])

best_option=sum_UC.index(max(sum_UC)) #best option is the option with the highest unity checks, being the most efficient option
best_option_ids=options[best_option]
best_option_lines=options_lines[best_option]
best_option_F=options_F[best_option]
print ’element ids of the best option:’, best_option_ids

#--------------------------------UPDATE DATAFRAMES---------------------------------------------------------

for i in best_option_ids:
    re.at[i, ’used’]=1

if (re.length[best_option_ids[2]] > best_option_lines[2].Length): #if last element is longer than distance to end bottom chord, element is cut and spare part included in dataframe
    print ’last element is cut and spare element added to database’
    spare_part_length=re.length[best_option_ids[2]]-best_option_lines[2].Length
    re.at[best_option_ids[2],’length’]=best_option_lines[2].Length #change length of element to used length
    spare_part = {’A’: re.at[best_option_ids[2],’A’], ’length’: spare_part_length, ’profile’: re.at[best_option_ids[2],’profile’], ’Fc’: re.at[best_option_ids[2],’Fc’], ’Ft’: re.at[best_option_ids[2],’Ft’], ’used’:2, ’w’: re.at[best_option_ids[2],’w’], ’h’:re.at[best_option_ids[2],’h’]}
    re=re.append(spare_part, ignore_index = True)

for i in range(len(best_option_ids)):
    ue = ue.append({’el_id’:best_option_ids[i],’profile’:re.at[best_option_ids[i],’profile’], ’k_line’:best_option_lines[i]},ignore_index = True)

print ’updated used element dataframe:’
print ue

#-------------------------------------OUTPUT--------------------------------------------------------------------

#output lists for visualization
lines=pr.obtain(ue.k_line.tolist())
profiles=pr.obtain(ue.profile.tolist())

#output for next iteration
F_output=F_input+[best_option_F]
Fx_output=Fx_input+[p2.X]+[p4.X]

#output for analysis of results
N_best=options_N[best_option]
for i in N_best:
    N.append(float(i))
UC_best=options_UC[best_option]
for i in UC_best:
    UC.append(float(i))

#updated dataframes
re.to_pickle(”re4.pkl”)
ue=ue.drop(columns=[’k_line’])
ue.to_pickle(”ue4.pkl”)
```

C.6. Finding the x-coordinates of the crossbeams

```
import rhinoscriptsyntax as rs
import Rhino.Geometry as rg
import scriptcontext as sc
import ghpythonremote as pr
from ghpythonlib import treehelpers as th
import ghpythonlib.components as ghcomp

np = sc.sticky['numpy']
sp = sc.sticky['scipy']
pd = sc.sticky['pandas']
rpy = sc.sticky['rpy']


x=[Fx[0]] #make a list with x-coordinates of crossbeams

for i in range(1,len(Fx)):
    if Fx[i] - Fx[i-1] > 3.5: #if the distance between the nodes is more than 3,5 meters, add
     a crossbeam halfway
        half_x= (Fx[i] + Fx[i-1])/2
        x.append(half_x)
        x.append(Fx[i])
    else:
        x.append(Fx[i])

print x
```

C.7. Finding the x-coordinates of all intersection points along the bottom chord

```
import rhinoscriptsyntax as rs
import Rhino.Geometry as rg
import scriptcontext as sc
import ghpythonremote as pr
from ghpythonlib import treehelpers as th
import ghpythonlib.components as ghcomp

np = sc.sticky['numpy']
sp = sc.sticky['scipy']
pd = sc.sticky['pandas']
rpy = sc.sticky['rpy']


print 'all nodes: ', Fx

if len(Fx)==0:
    bc_out=bc
    profiles_out=profiles
else:
    bc_out=[]
    profiles_out=[]
    for i in range(len(bc)):
        its=[]
        its.append(bc[i].From.X)
        print 'bc element ', i+1, ': from ', bc[i].From.X, ' to ', bc[i].To.X
        for j in Fx:
            if ((j > bc[i].From.X) & (j < bc[i].To.X)):
                its.append(j)
        its.append(bc[i].To.X)
        print 'its: ', its
        for k in range(len(its)-1):
            new_line=rg.Line(rs.CreatePoint(its[k], 0, 0), rs.CreatePoint(its[k+1], 0, 0))
            bc_out.append(new_line)
            profiles_out.append(profiles[i])
```

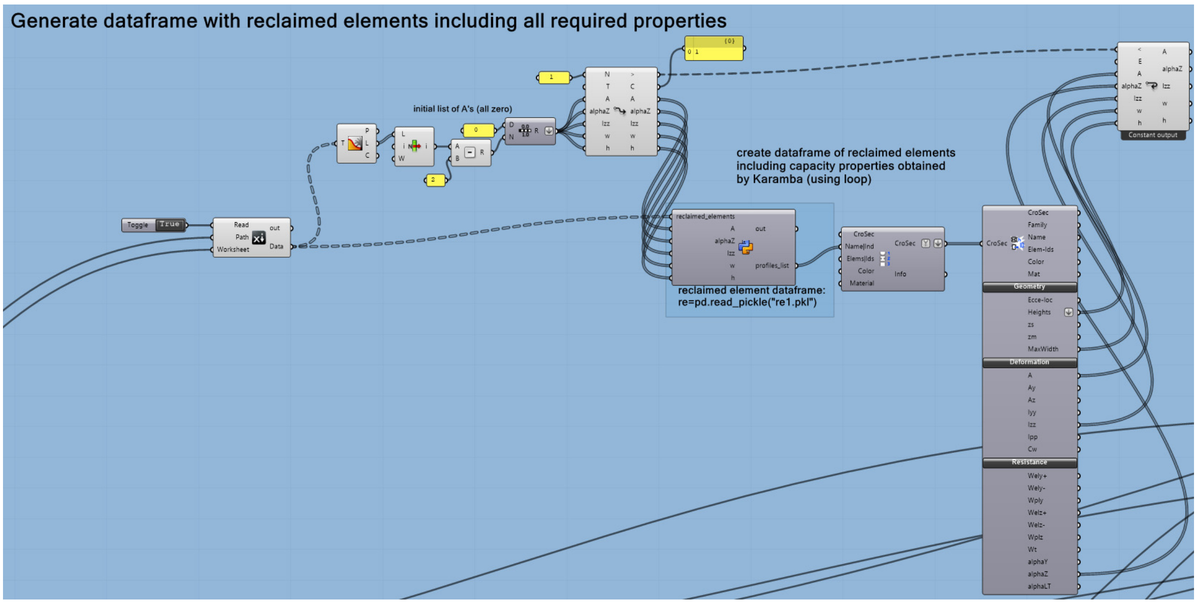
C.8. Storing the data of the valid truss options

```
import rhinoscriptsyntax as rs
import Rhino.Geometry as rg
import scriptcontext as sc
import ghpythonremote as pr
from ghpythonlib import treehelpers as th
import ghpythonlib.components as ghcomp

np = sc.sticky[’numpy’]
sp = sc.sticky[’scipy’]
pd = sc.sticky[’pandas’]
rpy = sc.sticky[’rpy’]

# Make persistent variables
if ”list_lines” not in globals():
    list_lines = []
if ”list_profiles” not in globals():
    list_profiles = []
if ”list_Fx” not in globals():
    list_Fx = []
if ”list_UC” not in globals():
    list_UC = []
if ”list_EC” not in globals():
    list_EC = []
if ”list_ids” not in globals():
    list_ids = []


# Update variable
if Update:
    list_lines.append(pr.obtain(LinesIn))
    list_profiles.append(pr.obtain(ProfilesIn))
    list_Fx.append(pr.obtain(FxIn))
    list_UC.append(UCIn)
    list_EC.append(ECIn)
    re=pd.read_pickle(”re4.pkl”)
    #spare_ids = re[re.used==2].index.values
    list_ids.append(pr.obtain(re[re.used==1].index.values.tolist()))

# Output variable
LinesOut = th.list_to_tree(list_lines)
ProfilesOut = th.list_to_tree(list_profiles)
FxOut = th.list_to_tree(list_Fx)
UCOut = list_UC
ECOut = list_EC
idsOut = th.list_to_tree(list_ids)
```

D

Grasshopper groups

D.1. Preparatory steps



Generate dataframe with reclaimed elements including all required properties

initial list of A's (all zero)

create dataframe of reclaimed elements including capacity properties obtained by Karamba (using loop)

reclaimed element dataframe:
re=pd.read_pickle("re1.pkl")

Figure D.1: Generation of database

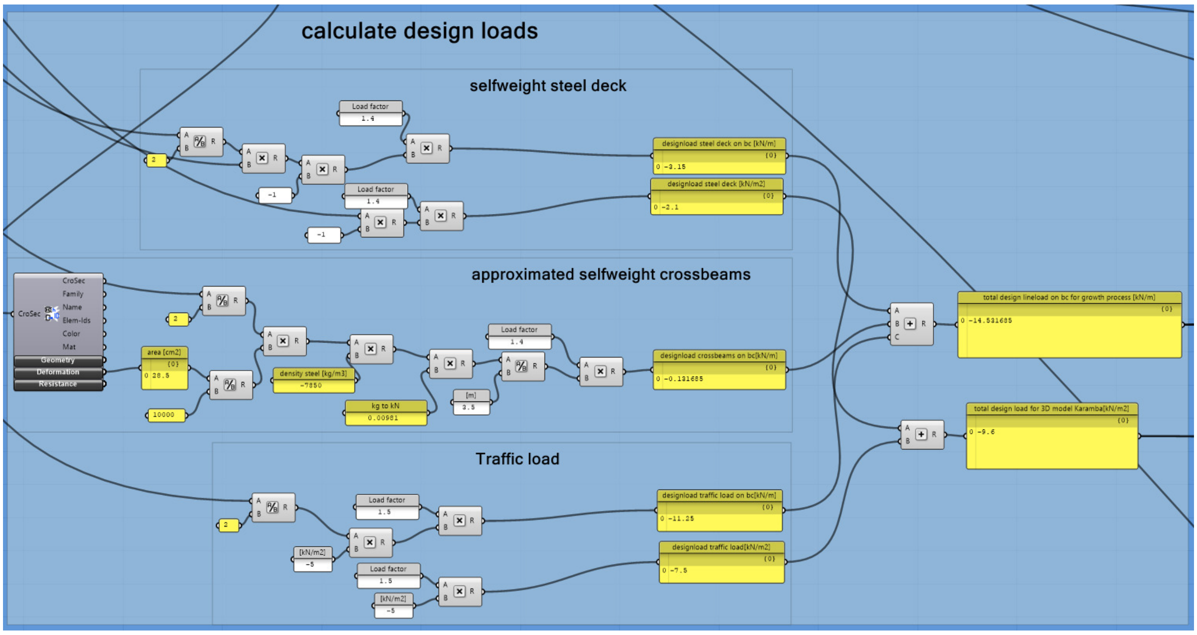


Figure D.2: Caculation of the design loads

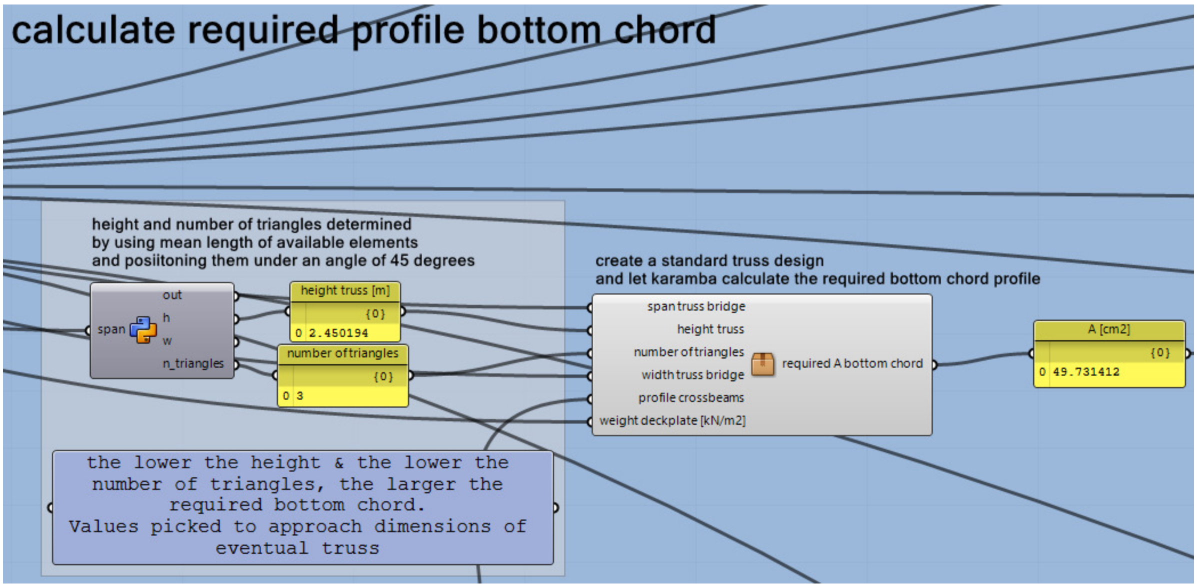
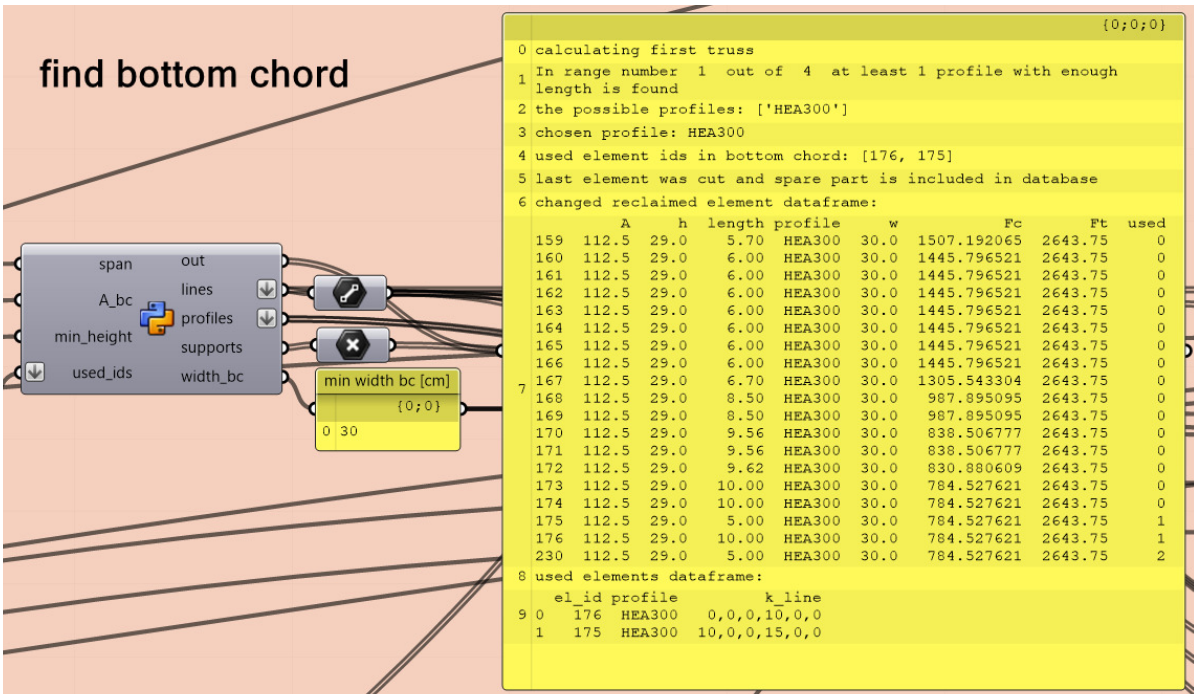

calculate required profile bottom chord
height and number of triangles determined
by using mean length of available elements
and posiitoning them under an angle of 45 degrees
span
out
h
w
n_triangles
height truss [m]
{0}
0 2.450194
number of triangles
{0}
0 3
create a standard truss design
and let karamba calculate the required bottom chord profile
span truss bridge
height truss
number of triangles
width truss bridge
profile crossbeams
weight deckplate [kN/m2]
required A bottom chord
A [cm2]
{0}
0 49.731412
the lower the height & the lower the
number of triangles, the larger the
required bottom chord.
Values picked to approach dimensions of
eventual truss

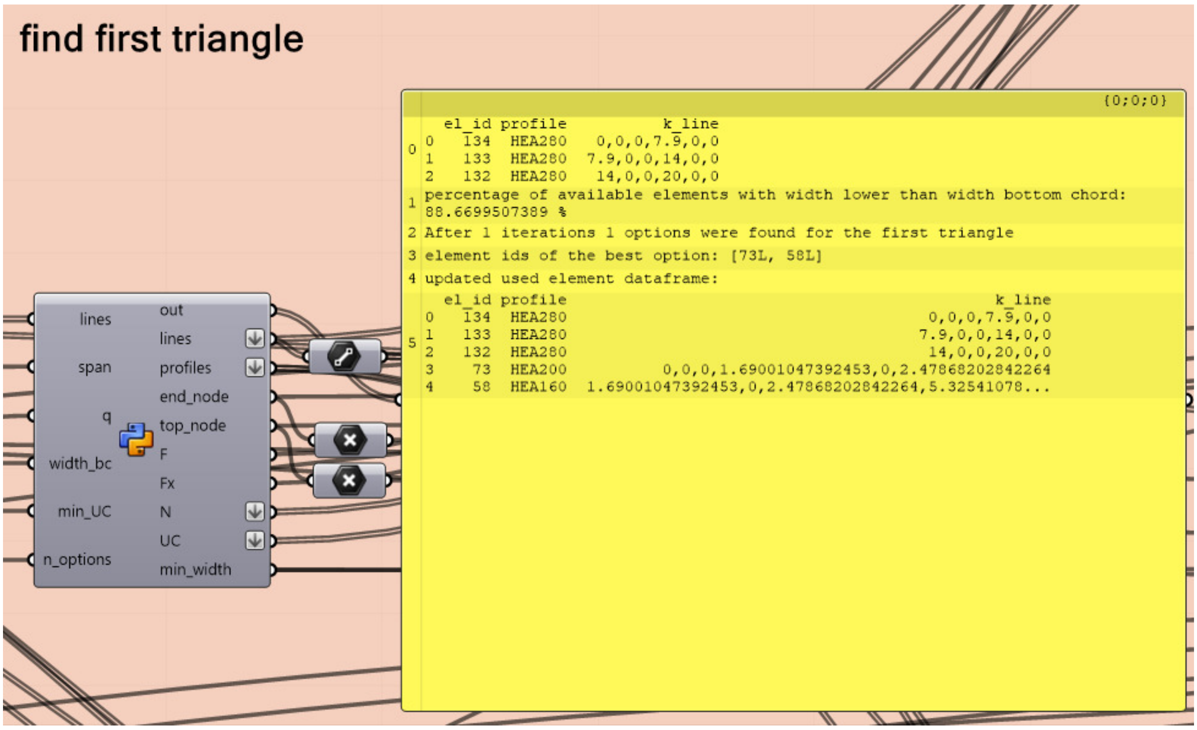
Figure D.3: Calculation of the required bottom chord

D.2. Growth algorithm



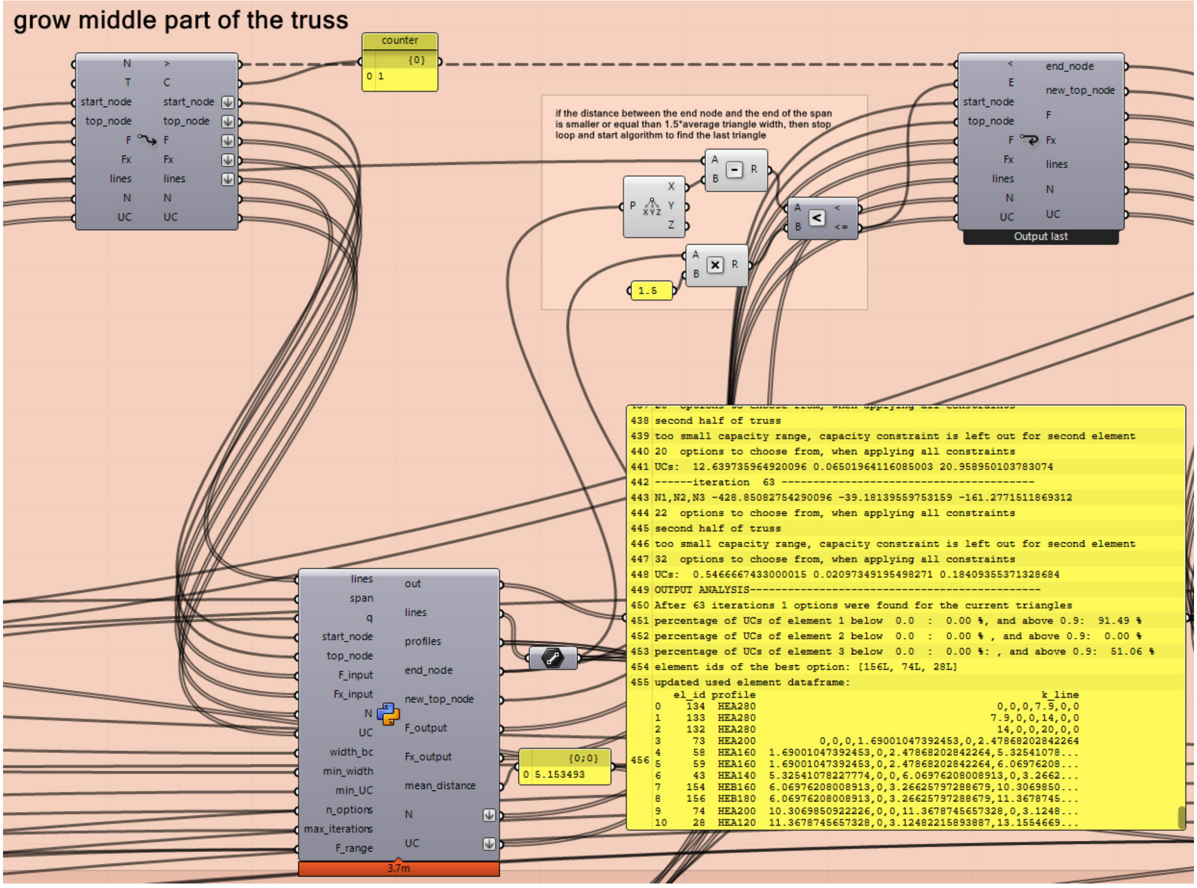
```
                                                                  {0;0;0}
0 calculating first truss
1 In range number  1  out of  4  at least 1 profile with enough
  length is found
2 the possible profiles: ['HEA300']
3 chosen profile: HEA300
4 used element ids in bottom chord: [176, 175]
5 last element was cut and spare part is included in database
6 changed reclaimed element dataframe:
             A     h  length profile     w           Fc       Ft  used
  159  112.5  29.0    5.70  HEA300  30.0  1507.192065  2643.75     0
  160  112.5  29.0    6.00  HEA300  30.0  1445.796521  2643.75     0
  161  112.5  29.0    6.00  HEA300  30.0  1445.796521  2643.75     0
  162  112.5  29.0    6.00  HEA300  30.0  1445.796521  2643.75     0
  163  112.5  29.0    6.00  HEA300  30.0  1445.796521  2643.75     0
  164  112.5  29.0    6.00  HEA300  30.0  1445.796521  2643.75     0
  165  112.5  29.0    6.00  HEA300  30.0  1445.796521  2643.75     0
  166  112.5  29.0    6.00  HEA300  30.0  1445.796521  2643.75     0
7 167  112.5  29.0    6.70  HEA300  30.0  1305.543304  2643.75     0
  168  112.5  29.0    8.50  HEA300  30.0   987.895095  2643.75     0
  169  112.5  29.0    8.50  HEA300  30.0   987.895095  2643.75     0
  170  112.5  29.0    9.56  HEA300  30.0   838.506777  2643.75     0
  171  112.5  29.0    9.56  HEA300  30.0   838.506777  2643.75     0
  172  112.5  29.0    9.62  HEA300  30.0   830.880609  2643.75     0
  173  112.5  29.0   10.00  HEA300  30.0   784.527621  2643.75     0
  174  112.5  29.0   10.00  HEA300  30.0   784.527621  2643.75     0
  175  112.5  29.0    5.00  HEA300  30.0   784.527621  2643.75     1
  176  112.5  29.0   10.00  HEA300  30.0   784.527621  2643.75     1
  230  112.5  29.0    5.00  HEA300  30.0   784.527621  2643.75     2
8 used elements dataframe:
    el_id profile        k_line
9 0   176  HEA300   0,0,0,10,0,0
  1   175  HEA300  10,0,0,15,0,0
```

Figure D.4: Selection of the bottom chord elements



```
                                                                  {0;0;0}
   el_id profile          k_line
0 0   134  HEA280   0,0,0,7.9,0,0
  1   133  HEA280  7.9,0,0,14,0,0
  2   132  HEA280   14,0,0,20,0,0
1 percentage of available elements with width lower than width bottom chord:
  88.6699507389 %
2 After 1 iterations 1 options were found for the first triangle
3 element ids of the best option: [73L, 58L]
4 updated used element dataframe:
   el_id profile                                        k_line
  0   134  HEA280                                  0,0,0,7.9,0,0
5 1   133  HEA280                                 7.9,0,0,14,0,0
  2   132  HEA280                                  14,0,0,20,0,0
  3    73  HEA200         0,0,0,1.69001047392453,0,2.47868202842264
  4    58  HEA160  1.69001047392453,0,2.47868202842264,5.32541078...
```

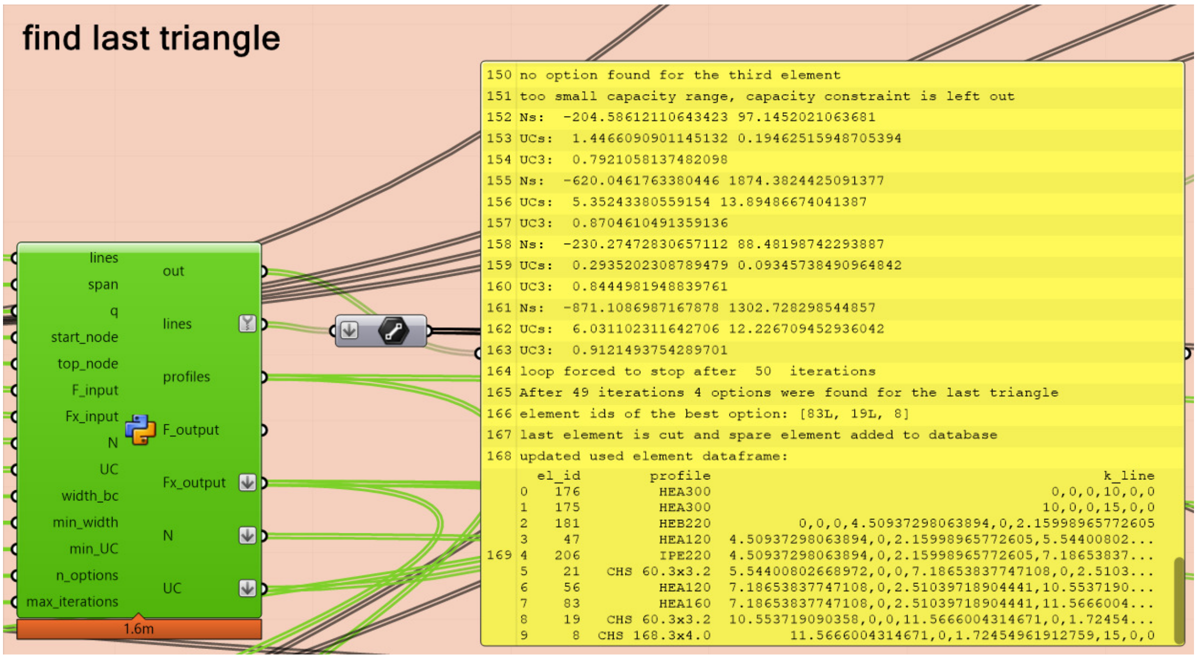
Figure D.5: Selection of the elements of the fist triangle



if the distance between the end node and the end of the span is smaller or equal than 1.5*average triangle width, then stop loop and start algorithm to find the last triangle

```
438 second half of truss
439 too small capacity range, capacity constraint is left out for second element
440 20  options to choose from, when applying all constraints
441 UCs:  12.639735964920096 0.06501964116085003 20.958950103783074
442 ------iteration  63 ----------------------------------------
443 N1,N2,N3 -428.85082754290096 -39.18139559753159 -161.2771511869312
444 22  options to choose from, when applying all constraints
445 second half of truss
446 too small capacity range, capacity constraint is left out for second element
447 32  options to choose from, when applying all constraints
448 UCs:  0.546666743300015 0.02097349195498271 0.18409355371328684
449 OUTPUT ANALYSIS-----------------------------------------
450 After 63 iterations 1 options were found for the current triangles
451 percentage of UCs of element 1 below  0.0  :  0.00 %, and above 0.9:  91.49 %
452 percentage of UCs of element 2 below  0.0  :  0.00 % , and above 0.9:  0.00 %
453 percentage of UCs of element 3 below  0.0  :  0.00 %: , and above 0.9:  51.06 %
454 element ids of the best option: [156L, 74L, 28L]
455 updated used element dataframe:
       el_id profile                                           k_line
    0    134  HEA280                                   0,0,0,7.9,0,0
    1    133  HEA280                                  7.9,0,0,14,0,0
    2    132  HEA280                                   14,0,0,20,0,0
    3     73  HEA200        0,0,0,1.69001047392453,0,2.47868202842264
456 4     58  HEA160  1.69001047392453,0,2.47868202842264,5.32541078...
    5     59  HEA160  1.69001047392453,0,2.47868202842264,6.06976208...
    6     43  HEA140  5.32541078227774,0,0,6.06976208008913,0,3.2662...
    7    154  HEB160  6.06976208008913,0,3.26625797288679,10.3069850...
    8    156  HEB180  6.06976208008913,0,3.26625797288679,11.3678745...
    9     74  HEA200  10.306985092226,0,0,11.3678745657328,0,3.1248...
    10    28  HEA120  11.3678745657328,0,3.12482215893887,13.1554669...
```

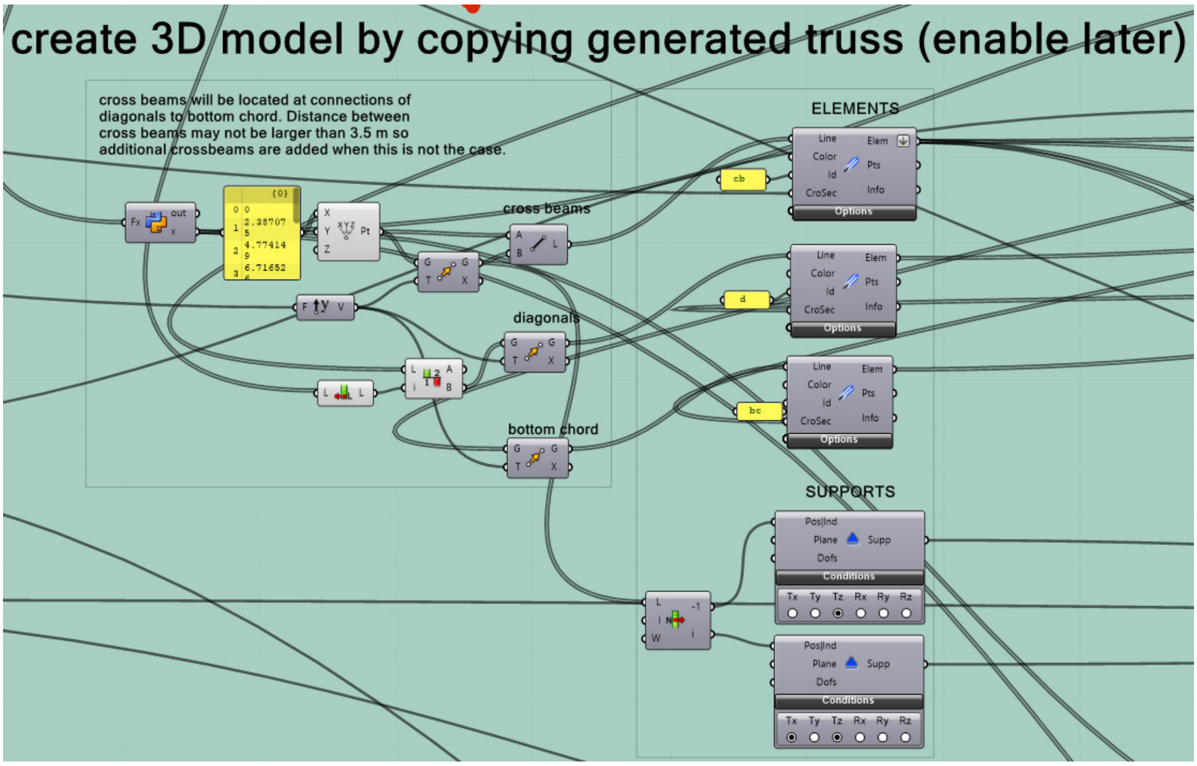
Figure D.6: Selection of the elements of the middle triangles



```
150 no option found for the third element
151 too small capacity range, capacity constraint is left out
152 Ns:  -204.58612110643423 97.1452021063681
153 UCs:  1.4466090901145132 0.19462515948705394
154 UC3:  0.7921058137482098
155 Ns:  -620.0461763380446 1874.3824425091377
156 UCs:  5.35243380559154 13.89486674041387
157 UC3:  0.8704610491359136
158 Ns:  -230.27472830657112 88.48198742293887
159 UCs:  0.2935202308789479 0.09345738490964842
160 UC3:  0.8444981948839761
161 Ns:  -871.1086987167878 1302.728298544857
162 UCs:  6.031102311642706 12.226709452936042
163 UC3:  0.9121493754289701
164 loop forced to stop after  50  iterations
165 After 49 iterations 4 options were found for the last triangle
166 element ids of the best option: [83L, 19L, 8]
167 last element is cut and spare element added to database
168 updated used element dataframe:
       el_id         profile                                         k_line
    0    176          HEA300                                  0,0,0,10,0,0
    1    175          HEA300                                 10,0,0,15,0,0
    2    181          HEB220      0,0,0,4.50937298063894,0,2.15998965772605
    3     47          HEA120  4.50937298063894,0,2.15998965772605,5.54400802...
169 4    206          IPE220  4.50937298063894,0,2.15998965772605,7.18653837...
    5     21    CHS 60.3x3.2  5.54400802668972,0,0,7.18653837747108,0,2.5103...
    6     56          HEA120  7.18653837747108,0,2.51039718904441,10.5537190...
    7     83          HEA160  7.18653837747108,0,2.51039718904441,11.5666004...
    8     19    CHS 60.3x3.2  10.553719090358,0,0,11.5666004314671,0,1.72454...
    9      8   CHS 168.3x4.0        11.5666004314671,0,1.72454961912759,15,0,0
```

Figure D.7: Selection of the elements of the last triangle

D.3. Visualisation and global calculations



create 3D model by copying generated truss (enable later)

cross beams will be located at connections of diagonals to bottom chord. Distance between cross beams may not be larger than 3.5 m so additional crossbeams are added when this is not the case.

Figure D.8: Generation of the temporary 3D model

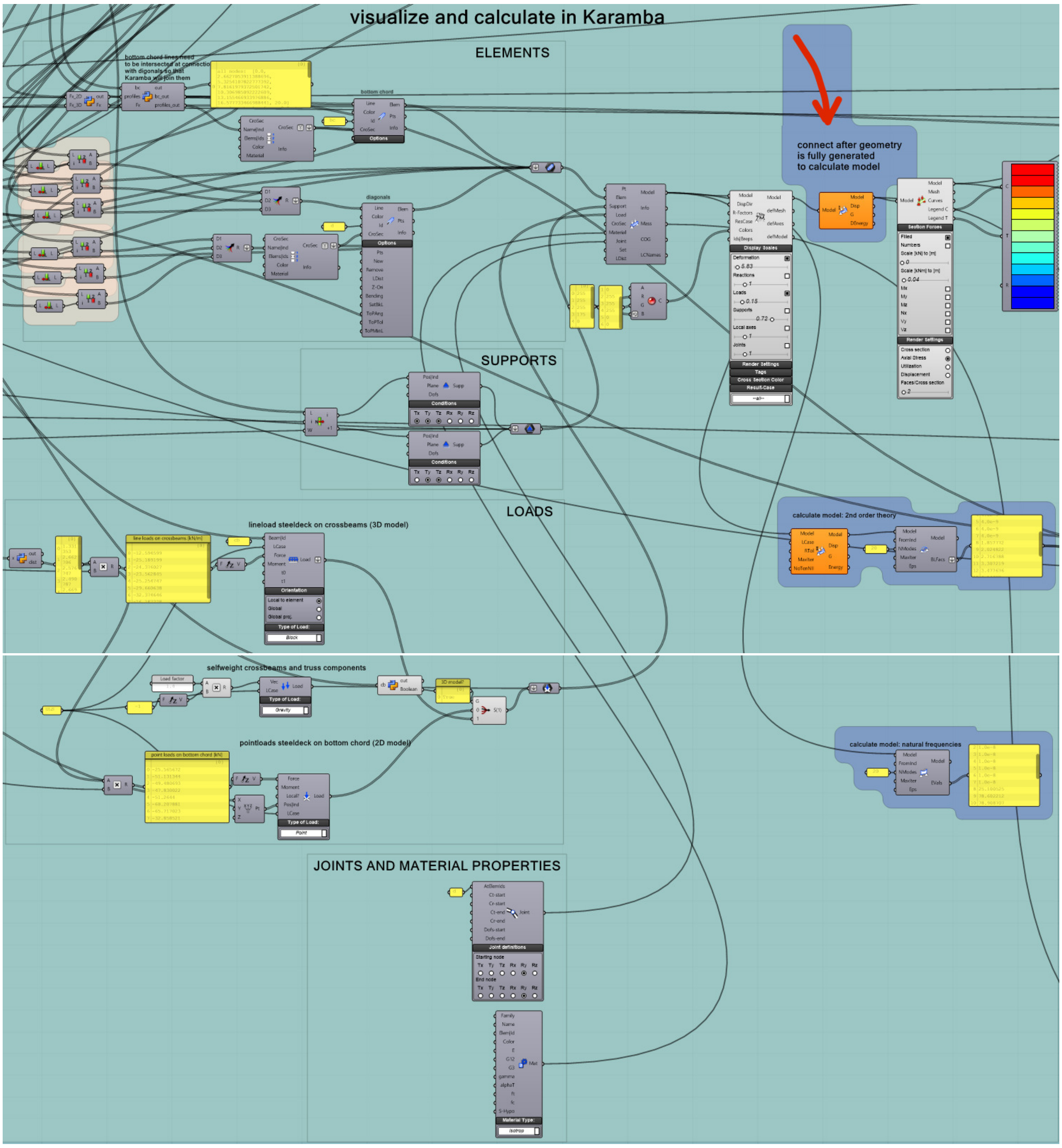

visualize and calculate in Karamba
ELEMENTS
bottom chord lines need to be intersected at connection with diagonals so that Karamba will join them
bottom chord
diagonals
connect after geometry is fully generated to calculate model
Display Scales
Deformation
Reactions
Loads
Supports
Local axes
Joints
Render Settings
Tags
Cross Section Color
Result-Case
Section Forces
Filled
Numbers
Cross section
Axial Stress
Utilization
Displacement
Faces/Cross section
SUPPORTS
Conditions
LOADS
lineload steeldeck on crossbeams (3D model)
line loads on crossbeams (kN/m)
Orientation
Local to element
Global
Global proj.
Type of Load:
Block
calculate model: 2nd order theory
selfweight crossbeams and truss components
Load factor
3D model?
Gravity
pointloads steeldeck on bottom chord (2D model)
point loads on bottom chord (kN)
Point
calculate model: natural frequencies
JOINTS AND MATERIAL PROPERTIES
Joint definitions
Starting node
End node
Material Type:

Figure D.9: Karamba visualisation and calculations

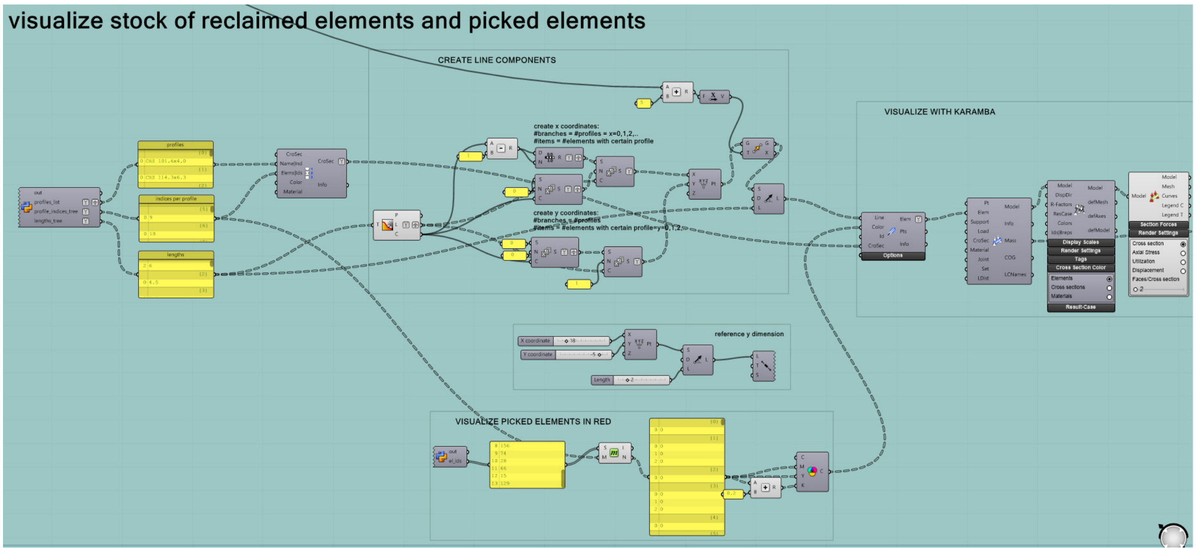

visualize stock of reclaimed elements and picked elements
CREATE LINE COMPONENTS
create x coordinates:
#branches = #profiles = x=0,1,2,..
#items = #elements with certain profile
create y coordinates:
#branches = #profiles
#items = #elements with certain profile=y=0,1,2
VISUALIZE WITH KARAMBA
profiles
indices per profile
lengths
reference y dimension
VISUALIZE PICKED ELEMENTS IN RED

Figure D.10: Visualisation of the stock of elements and the selected elements for the truss

D.4. Calculation environmental impact

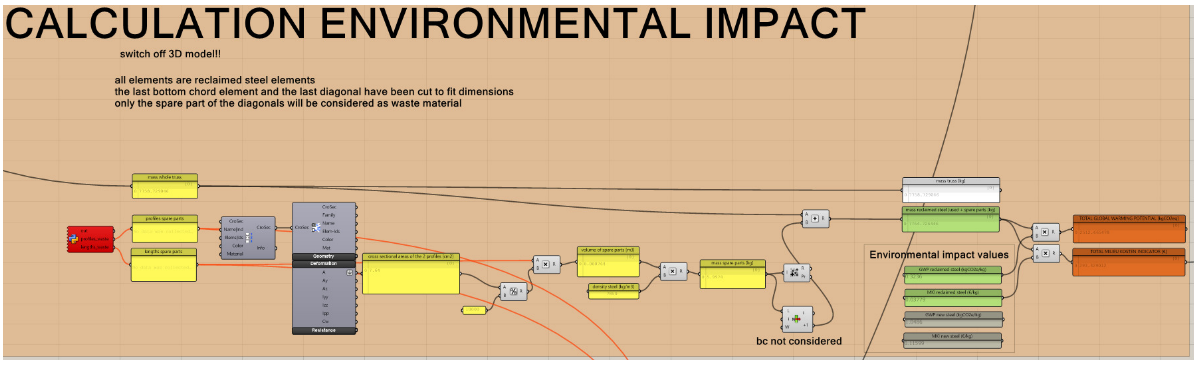

CALCULATION ENVIRONMENTAL IMPACT
switch off 3D model!!
all elements are reclaimed steel elements
the last bottom chord element and the last diagonal have been cut to fit dimensions
only the spare part of the diagonals will be considered as waste material
mass whole truss
profiles spare parts
lengths spare parts
cross sectional areas of the 2 profiles (cm2)
volume of spare parts (m3)
density steel (kg/m3)
mass spare parts (kg)
mass truss (kg)
mass reclaimed steel used + spare parts (kg)
Environmental impact values
GWP reclaimed steel (kgCO2e/kg)
MKI reclaimed steel (€/kg)
GWP new steel (kgCO2e/kg)
MKI new steel (€/kg)
TOTAL GLOBAL WARMING POTENTIAL (kgCO2eq)
TOTAL MILIEU KOSTEN INDICATOR (€)
bc not considered

Figure D.11: Calculation of the MKI and GWP values

D.5. Generation of the solution clouds and modelling of the truss bridge designs

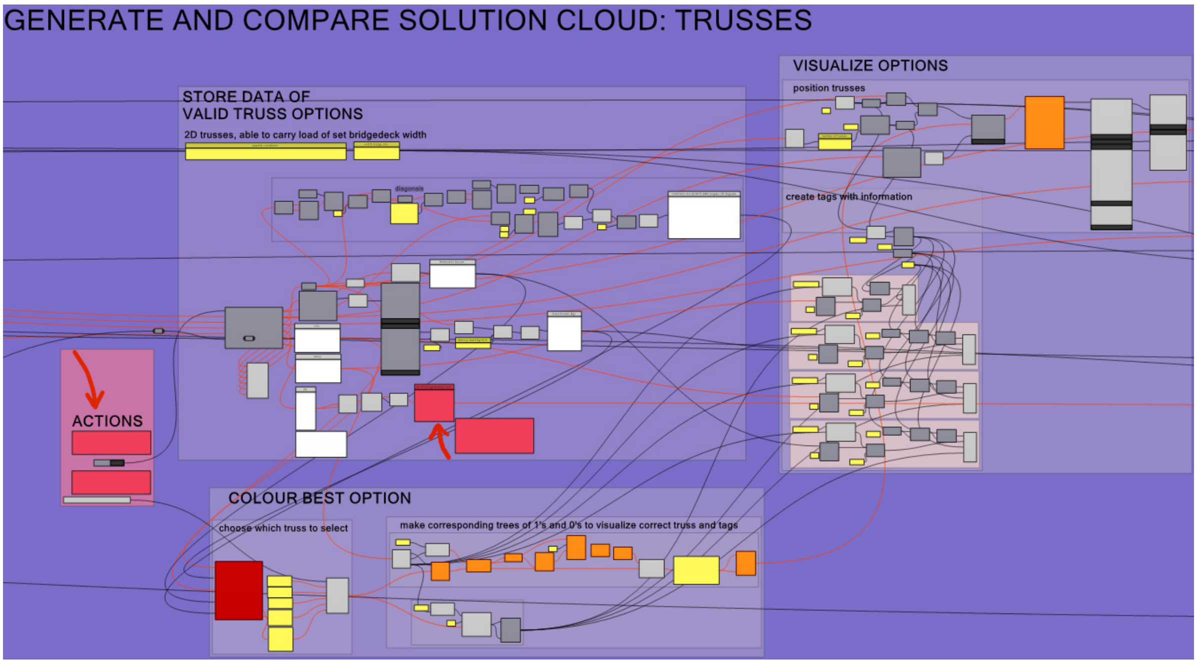

GENERATE AND COMPARE SOLUTION CLOUD: TRUSSES
STORE DATA OF
VALID TRUSS OPTIONS
2D trusses, able to carry load of set bridgedeck width
diagonals
VISUALIZE OPTIONS
position trusses
create tags with information
ACTIONS
COLOUR BEST OPTION
choose which truss to select
make corresponding trees of 1's and 0's to visualize correct truss and tags

Figure D.12: Generation of the solution cloud of valid truss options

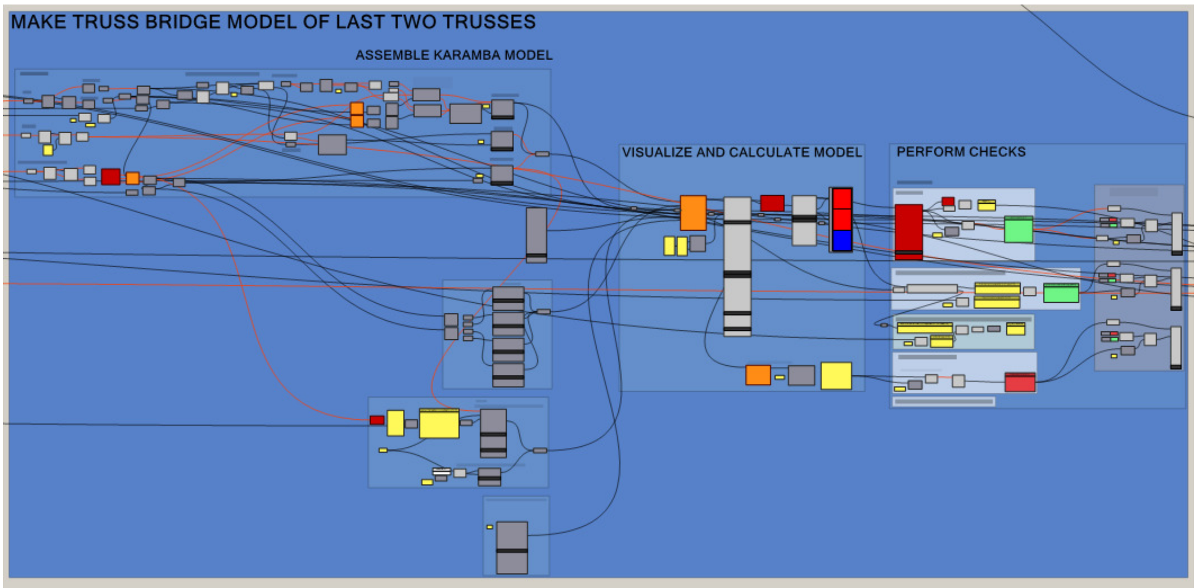

MAKE TRUSS BRIDGE MODEL OF LAST TWO TRUSSES
ASSEMBLE KARAMBA MODEL
VISUALIZE AND CALCULATE MODEL
PERFORM CHECKS

Figure D.13: Creation of truss bridge model and structural checks with Karamba

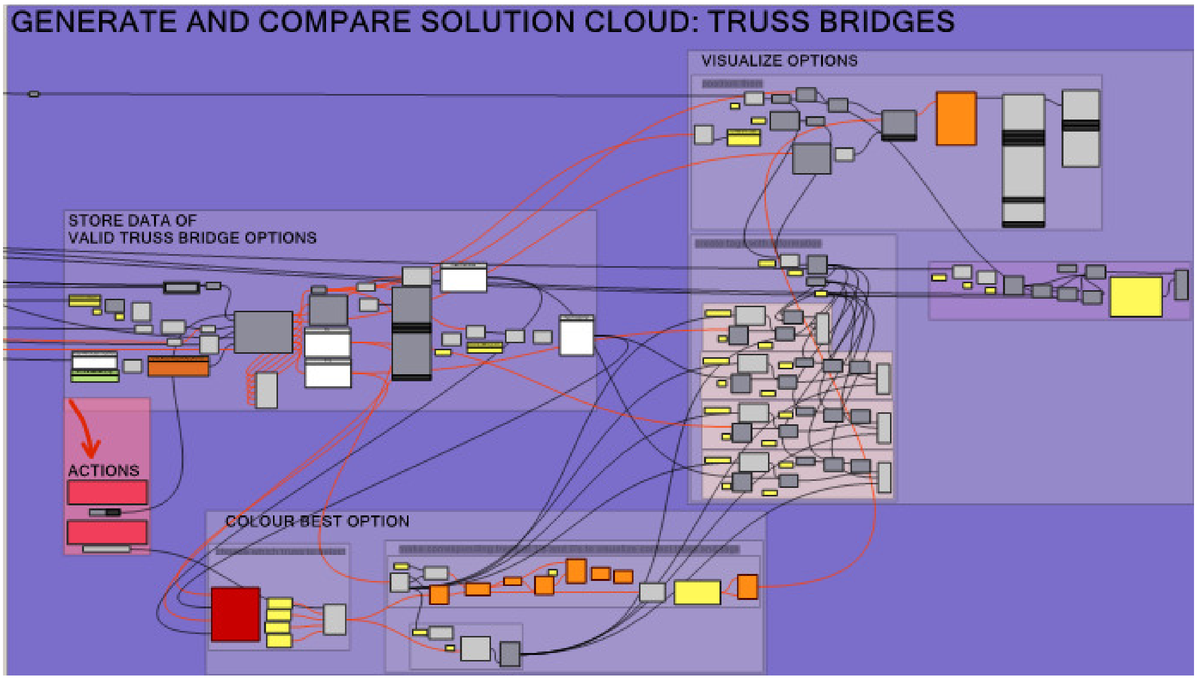

GENERATE AND COMPARE SOLUTION CLOUD: TRUSS BRIDGES
VISUALIZE OPTIONS
STORE DATA OF VALID TRUSS BRIDGE OPTIONS
ACTIONS
COLOUR BEST OPTION

Figure D.14: Generation of the solution cloud of valid truss bridge options