Foreword

This technical report is part of a dual track graduation project done at the Faculty of Architecture at the TU Delft. The project is a collaboration between Architectural Engineering and Building Technology. It is the result of a semester long research project for the BT side of the graduation project which started in October 2009 and which will be completed together with the Architectural Engineering part on the 25th of June 2010.

W. Gielen, Delft
Contents

1. INTRODUCTION 9
   1.1 Assignment ................................................................................................................ 9
   1.2 Design ....................................................................................................................... 9

2. PROBLEM AREA 11
   2.1 Problem statement .................................................................................................... 11
   2.2 Research question ................................................................................................... 11
   2.3 Goals ....................................................................................................................... 11
   2.4 Method ..................................................................................................................... 12

3. BRIEF 13
   3.1 Program .................................................................................................................. 13
   3.2 Construction .......................................................................................................... 14
   3.3 Loads ...................................................................................................................... 14

4. GENERATION 15
   4.1 Input ...................................................................................................................... 15
   4.2 Parameters ............................................................................................................. 15
   4.3 Output ..................................................................................................................... 16
   4.4 Generation Tool ...................................................................................................... 17

5. ANALYSIS 21
   5.1 Importing ............................................................................................................... 21
   5.2 Modeling ................................................................................................................. 21
   5.3 Results .................................................................................................................... 24
   5.4 Conclusions and Recommendations ......................................................................... 31

6. DETAILING 33
   6.1 Input ...................................................................................................................... 33
   6.2 Demands ................................................................................................................. 33
   6.3 Output ..................................................................................................................... 35
   6.4 Detailing Tool ........................................................................................................... 35
7. APPLICATION IN THE ARCHITECTURAL DESIGN

7.1 The role of the tool in the conceptual design .............................................................. 41

7.2 The role of the tool in the final design ....................................................................... 42

7.3 The role of the tool in the final design ....................................................................... 43

7.3 The role of parametric modeling in the architectural design process ....................... 44

8. CONCLUSIONS ............................................................................................................ 47

LITERATURE .................................................................................................................. 49

APPENDIX ..................................................................................................................... 50

I. Results ......................................................................................................................... Error! Bookmark not defined.

II. Generation Tool script ............................................................................................. Error! Bookmark not defined.

III. Detailing Tool ........................................................................................................... Error! Bookmark not defined.

IV. Generation tool variant: Varying stem height ......................................................... Error! Bookmark not defined.
1. Introduction

1.1 Assignment
This research project comes forth from a preceding Architectural Engineering graduation studio. The setting for this studio is the group of cities in the Netherlands that form the network metropolis the Randstad and the development that this network will undergo in the coming years. The Dutch government has a vision, called Randstad 2040, which describes the growth of the Randstad in the future and the important role infrastructure will play, connecting the different cities, who each have their own qualities and spear points. The site for this assignment is the Prins Clausplein intersection near the city of The Hague. Here the highways A12 and A4 cross, creating a large barrier along the edge of the city which limits the growth of the city. This situation is very common in the Netherlands. As the cities grow they eventually run into the surrounding highway network. The areas around the highways have always been a kind of no-mans-land because of noise and air pollution. Hence the city is forced to grow around or jump the highway and expand on the other side, leaving the highway to run between the two parts of the city. When this happens a problem arises. The highway becomes a giant scar in the urban fabric, a barrier that splits the city in two. The situation resembles a river running through a city, bridges are needed to connect both banks. Bridging is important for creating unity and coherence: without them the new expansions will never truly belong to the city.

1.2 Design
In the Architectural Engineering graduation studio a masterplan for the area around the Prins Clausplein, the A4/Vliet-zone, was developed. One of the aspects of this vision was to create connections over and under the highway to connect both sides. For my AE graduation project I filled in one of these connections by designing a building that bridges the A4 highway, thus connecting the parts of The Hague that are separated by this highway. The building contains various supporting functions for the different target groups that use the overpass.
2. Problem area

2.1 Problem statement

While working on the design I ran into problems regarding the load bearing structure. It proved to be quite a challenge to match the structural grids of the floors of the shops above the highway, to the grid of the floors of parking under the highway. The parking layout is not quite regular, the highway cuts through the building between those floors and there are three different traffic routes to take into account (one for motor vehicles, one for bikes and one for pedestrians). I explored different options to match the two grids. The options, seen in figure 1, can be divided in four categories; table, trees, V-column and hanging constructions.

![Figure 1: Grid matching](image)

From these four categories I chose to focus on the tree constructions, as they gave the most freedom to fit both grids and didn't require a lot of reinforcements as did the table constructions. As the design began to take shape it became obvious that, apart from being different, the grids would also not be regular, and thus the idea was born to create a tool that would support me when designing the load bearing structure by generating the structure automatically based on a set of criteria set by myself. This is the topic for this research and design semester.

2.2 Research question

The research question that will be answered is the following:

*How can a non-orthogonal load bearing structure be automatically generated, taking into account the criteria set by the brief, the architectural design and the structural performance requirements?*

2.3 Goals

A set brief will be the input for a tool which will generate a structural system. This results in the first goal of this semester:
• Create a tool that generates a structural system which takes the set of design criteria into account.

This system will then be detailed by a constructing tool. This results in the second goal;
• Detail the parametrically generated structure.

This means that the research will focus on two different parametric tools, the Generation Tool and the Detailing Tool.

Apart from creating these tools there is a third, crucial, goal that makes up the research;
• Validate the structural system using a performance analysis.

The way that these analyses fit in the workflow can be seen in the figure below:

The results from the generation tool will be analyzed using a structural analysis program. The results from these tests will be used to identify problems in the construction. It can then be decided whether the problem is best solved by changing the generation tool of the detailing tool.

The deliverables of the research will be two tools which together create a load bearing system based on a predefined brief.

2.4 Method
To create the workflow described in the previous paragraph various software programs must be selected. These programs must be able to provide the options needed to create the geometry, create the custom tools, run the analyses, and produce the detailing. During the architectural semester, McNeel - Rhinoceros was used to model the design. Rhino proved to be a good starting point as it highly compatible with other cad programs, has a very flexible scripting engine and last but not least has the powerful Grasshopper plug-in. Grasshopper allows for generative modeling in Rhino using a very intuitive and easy to use interface. It also provides the ability to create and execute scripts in both VB.NET and C#. For more information about Grasshopper and its workings see www.grasshopper3d.com. The only other program that is needed to complete the workflow is the structural analysis program. For this, Oasys - GSA is chosen, as this is quite easy to link it to Rhino. It is quite easy to learn, well documented and freely available to students. The selection of these programs means that there are really only two different working environments, Rhino/Grasshopper and GSA, which makes life a lot easier.
3. Brief
The first thing needed is an extensive brief that forms the input for the tools. While the brief of the architectural design could have been used, I have chosen to create a new, more generic situation; this to ensure that the solution is bound to a very specific situation. The second reason to set up a new brief is to limit the initial application size of the tool. This makes it easier to solve problems and check for irregularities. The brief is divided into three part: program, construction and loads. Under program a general description is given of the programmatic elements on the different floors. The second part will describe the constructive and mechanical constraints. The final part will list the loads that are to be applied to the construction when analyzing it.

3.1 Program

Ground Floor
Function: Parking
Floor height: 7m
Area: 32x40m = 1280m²

On the ground floor there is a parking garage, with a standard 5-6-5m pattern. This means that the columns can only be placed between the parking spaces.

First Floor to Third Floor
Function: Offices
Floor height: 7m
Area: 37.5x42m = 1575m²

The floors above the parking garage are designed for offices and other general use. The columns are evenly spaced on a 6m by 7 columns grid.

Figure 2: Ground floor

Figure 3: Floors 1,2,3
3.2 Construction

Material
Steel

Sections
Stem: Circular, Diameter 500-1000mm, Thickness 10-20mm
Branch: Circular, Diameter 200-400mm, Thickness 10-20mm

Constraints
Base: encastred to floor
Floors pinned to cores

Standardization
Two different diameters
Length can be individual
Details that can accommodate the needed angle differences

Element size constraints
Not longer than a standard lorry

Construction costs
Material costs
Man hours (welds €/m)
Speed

Construction order
Stem → Branches → inter connects → outer connects

3.3 Loads

Applied loads
Dead loads: 3kN/m2
Live loads: 5kN/m2
Wind loads: 1.5kN/m2

The floors are loaded with a live load of 5kN/m2 which makes for 7.5kN/m2 when the safety factors are taken into account. When the live loads are translated into point loads on the columns they are 630kN.
The wind loads are calculated over a faced 21m high and 37m wide. These wind loads are applied as 56 point loads of 30kN.

Safety factors
Dead Loads: 1.2x
Live loads: 1.5x
Wind loads: 1.5x
4. Generation

This chapter will deal with the first of the two tools, the generation tool. The first step will be to look at the required input. What is the input from the brief that is needed to generate the structural system? The second step will be to define the parameters that control the output of the tool. These parameters are fed into the tool and directly influence the outcome. The last step will be to define the required output of the tool and is closely linked to the required input of the detailing tool. After these preliminary steps the generation tool itself will be described.

![Figure 4: The Generation Tool](image)

4.1 Input

The input of the generation is closely linked to the problem that it needs to solve: create a tree-like load bearing structure to match two structural grids. Therefore the input must be the two grids that need to be matched. The only question is in what format the grids are inputted. They could be physical points, t-parameters on a line or UV subdivision of a surface. Because of the way that script can be embedded in a grasshopper network it works best if the script expects a list of points and lets the grasshopper network determine how these points are created. This makes the script much more versatile because there is no need to create different scripts for different input types. If a different method of generating the structural grid is required, the script can simply be attached to a different grasshopper network.

There is however a big difference between the upper grid (cloud points) and the lower grid (stem points), as the upper grid is more or less a fixed situation, while the lower grid is more flexible and must be influenceable by the user of tool. While the cloud points can be linked to a subdivision of a surface, the stem points must be physical points that can be moved independently.

4.2 Parameters

After the input, the next things to define, are the parameters that control the rules that the structural system must follow. These rules are there to prevent the script to create an outcome that is not feasible. This means that they are linked to various requirements that are important when designing a load bearing system.

The two most important requirements of any structural system are stability and that it is capable of carrying the loads, inflicted on it. Stability is achieved when the system is able to transfer all the loads to the earth with a limited amount of transformation of the structure. This is done by either connecting the elements using fixed connections or using hinged connections with braces. Using these two methods the system must be able to withstand lateral and rotational forces in all three directions (x,y,z). In this case this is achieved by encastering the base of the stems, creating stiff floor slabs pinned to the cores of the building (Figure 5). This arrangement is fixed and thus every outcome of the generation tool will be stable.
Now that stability is taken care of it is time to make sure that the system is able to take the loads without failing. The system fails when the internal stresses exceed the maximum stress the material can take. The amount of stress is linked to the loads on the structure, the geometric arrangement of the elements, the sections of the elements and the material properties. Of these four only the geometric arrangement of the elements is determined by the generation tool. The loads are fixed and the section/material properties are defined later in the workflow. This means that if the parameters must ensure a feasible solution they must be able to steer away from certain geometric arrangements. It is important to remember that the values of the parameters are not yet defined. It is merely the question which parameters are needed, the minimum and maximum values will be determined later by analyzing the results of the structural analyses.

In the case of the tree there are four main geometric factors that influence the forces in the structure.

The first is the angle between branch and stem. The larger this angle, the bigger the axial force in the branch. Therefore it is important to control the maximum angle that a branch makes.

The second is the branch length which must be limited to prevent buckling, though with a limited branch angle and ceiling height the maximum branch length is rarely the limiting factor. The third factor is the stem height which, because it is encastered at the base, has to be able to withstand large bending moments.

This brings us to the last factor which is the placement of the stem in relation to the cloud points it supports. As the stem is placed more and more out of the center, the bending moments increase. The problem however is that the placement of the column is inputted by the user, which means that it cannot be changed by the script. The only way to deal with this is to either tackle the problem in the detailing tool or radically change the way the tool works. This limitation (and others) will be described in paragraph 4.4.

Apart from the load bearing requirements there are also many others, like functional, economical and constructional. Of these only the functional aspect of the stem height is incorporated in the parameters.

4.3 Output

A large part of the research for this semester was on possibilities to link multiple programs together to speed up a workflow. In this case I tried to link Rhino to GSA so that it would become much easier to run structural analyses. It is for this reason that the output of the script is closely linked to the required input of the analysis software (and to a lesser extent the detailing tool).

Looking at GSA it became clear that the easiest method would be to use Autodesk's DWG file format. GSA has the ability to read various geometry types from these files and is able to automatically assign properties to objects in different layers. The geometry required consists of simple line segments. This means that the output of the generation tool has to be lines split in two lists; one with the branches and one with the stems. These two lists can then be baked (Grasshoppers term for generating Rhino-native geometry) to two different
layers which will be used by GSA to determine the section properties for the branches and the stems.
Apart from these two required outputs the script can be asked to output a number of other things like a list of all possible branches or all the branch points. This is useful when debugging the tool and understanding the outcome.

4.4 Generation Tool

Now that all the input, parameters and outputs have been discussed, we can start to look at the actual workings of the script. While explaining the different steps, references will be made to line numbers where that step is executed in the source code, the code can be found in appendix II. The first thing to do is to describe the desired result, and then we will look how this is achieved. Finally the inherent limitations will be looked at.

4.4.1 Desired result

The main result the script must come up with is an organization of elements that form a load bearing system that matches the two supplied grids. Within that result the script must be able to achieve two sub goals. First it must be able to avoid creating results that are infeasible or undesirable. Paragraph 4.2 showed that there are a number of factors that need to be limited or controlled like the maximum branch angle. Apart from avoiding situations, the script must also be able to select the best option if there is more than one solution.

4.4.2 Workings

The first thing the script does, is to create the stems of the trees at the points that are inputted. On these stems the script places branch points. These are the points where the branches connect to the stem.

The script creates branch points which are the points where the branches connect to the stem. These points are placed above the stem points at specific intervals (lines: 293-303). These intervals are controlled by the first_stem_length and sec_stem_length parameters. All these branch points are added to a list that is later used to create the branches (line: 299). After creating the branch points the script creates the lines that form the stems (lines: 305-312).

The second step in the script creates all the valid branches from the point of view of the upper grid. This means that the script looks at each cloud point and connects it to all the branch points that give valid branches. Valid means that they are shorter then the maximum branch length and the angle is not larger then the maximum branch angle (lines: 322-336).

The third step filters all the valid branches from the point of the stems. A branch is valid if it is not the only branch of a branch point or when it is the only branch of a cloud point. This ensures that all the stems are either empty or used by more then one cloud point. Only in the case where a stem is the only option for a cloud point is it allowed that a stem has a single branch (lines: 357-386).
For the fourth step we need to create a private function which will allow us to calculate the fitness of a branch. This value can then be used to select the best branches. The creation of private functions must be done in a separate section outside the private sub that is visible by default. The private function, called valueBranch, will take a line as input and output a value. This value is calculated by the following formula:

\[
\text{valueBranch} = \text{abs}(\text{Branch angle} - \text{optimal angle}) \times \text{branch length}
\]

This means that the fitness of a branch is related to both its length and angle. The lower the valueBranch of a branch the 'better' it is.

In the last step of the script the branches are evaluated using the valueBranch function and the best branch is selected. These branches are then collected in a list (lines: 390-408). This list, together with the list of stems is outputted and forms the resulting load bearing structure (lines: 414 and 415). The result can be seen in Figure 10.

Figure 9: Step 4

Figure 10: Results from the Generation Tool
4.4.3 Limitations

The next paragraph will explore the limitations that are inherent to the Generation Tool. These limitations can be related to the complexity of the situation, the quality of the results or to the architectural requirements. Possible solutions for these problems have been explored in separate scripts, but due to time constraints were not merged with the main generation tool.

Fixed branch point height
One of the architectural limitations is the fixed stem height that is embedded in the tool. While it is possible to change the stem height it can not be varied from tree to tree. While this is not a bad thing from a structural standpoint: branches will always have the steepest possible angle; this does give a rather rigid impression from an architectural standpoint (see Figure 11). As the idea is to make the construction appear as a collection of individual trees it would be more logical if there was some variation in stem height. In a possible solution for this problem the branch point of each tree is defined by the average length of the branches. This means that when the branches are shorter and thus steeper, the branch point is higher. This means that the structural demands are kept in mind as the branch point is moved relative to the fitness of the branches. The trees with long, shallow branches will still have their branch point on the minimum stem height. The effect of this can be seen in Figure 12. Even though the difference is not extremely large the effect is quite pleasing and suites the objective to create more individual trees (see Figure 12).

Multiple solutions possible without knowing which one is better
Another limitation of the generation tool is its inability to evaluate and compare the generated solutions. As the trees are placed by hand it is up to the user to select the best column placement. This is a problem because there is no easy way for the user to evaluate his choices. An example of this problem is illustrated on the right. In version 1 ten trees are needed to support all the points while in version 2 only nine are needed. The problem is however that it is unclear which solution is better, even though it is clear that version 1 uses fewer trees. It might be that in version 1 the resulting forces are larger which have to be countered by using larger dimensions for the stems and branches. The way that this limitation can be resolved is to let the script place the trees according to a set of criteria and values. This way the user only defines the location of where the trees are unwanted. Based on the input and the criteria the script can then evaluate different options and select the best one.
Column placement optimization needs to be done by hand which makes it hard to find the best location for the column.

A limitation that is related to the previous one is that the placement of the stem of a tree compared to the branches is also not optimized. To bring down the stresses in the stem, it must be placed as close to the center of the branches as possible. As the stem is moved away from this point the eccentricity creates a large bending moment in the column. In most situations the eccentricity cannot be avoided entirely, but it must be minimized where it can. For example in the situation below where the placement of the stem is confined to the line, the output of the script should be situation two.

The solution to this problem will lie in determining the location of the center point and then finding the closest point on the line. This can be done by iteratively dividing the line in two and checking which half is closer to the point. If this is done a number of times the result will converge to the closest point.

![Figure 16: Situation 1](image16.png) ![Figure 15: Situation 2](image15.png)
5. Analysis
This chapter will look at the structural analyses, required to validate the outcome of the Generation Tool. Testing various situations will help understand problematic situations and will allow us to set the parameters of both the Generation and the Detailing tool accordingly. To run the analyses we first need to import the results from the Generation tool. After that we will look at applying loads, creating section profiles and setting releases and constraints. Once the structure has been modeled various tests will be done to find problematic situations. The results of theses tests will then be analyzed and will lead to a set of conclusions and recommendations for both the Generation and the Detailing Tool.

5.1 Importing
When creating a workflow between different programs it is always difficult to link them efficiently. Often a lot of time is wasted moving data between programs. This is also the case when the output from the generation tool is imported into GSA. Below is the list of the steps that are needed to get a model ready for analysis starting with the output from the generation tool.

1. Baking Grasshopper to the appropriate layers, based on section properties
2. Exporting as DWG file
3. Opening in GSA
4. Set the section properties
5. Assigning constraints
6. Assigning releases
7. Applying loads with the appropriate loadcases

The list shows that there are a lot of manual actions needed to get from the output to a working analysis model. This of course slows down the iteration speed. To speed up the process a base file is prepped with the loads and loadcases, these can then be copied to each variant that come out of the generation tool.

5.2 Modeling
To get accurate results from the analysis it is important to model the construction correctly. The modeling of the construction can be divided into five sections: shape, materials, joints, constraints and loads. Each section will be briefly described below.

5.2.1 Shape
There are two aspects that are important; the first is the shape of the overall construction. This shape is what is outputted by the generation tool and contains the location, orientation and length of the members. The second aspect of shape is the section properties of the individual members. Every member needs to have a section type assigned that corresponds with the real-life situation. In this case all the members have hollow circular sections of various diameters and thicknesses.

5.2.2 Materials
The whole construction is made from structural steel. This means that the standard steel material properties already imbedded in GSA can be used. There are however still some differences between the branches and the stems. The stems have to cope with very large bending moments and to allow them to have the same slender appearance as the branches S355 steel is used instead of the more common S235. This means that the yield strength is increased from 235N/mm² to 355N/mm². This does not however need to be modeled in GSA as it only means that the maximum allowed stress increases.
5.2.3 Joints
The connection between the elements is another thing that is important to model correctly. The force distribution in a construction differs considerably when the joints are fixed compared to hinged. To minimize the bending moments in the branches the joints at their end are hinged. This is modeled in GSA by setting the rotational releases for those elements. Each element has three localized rotational axes, the x, y and z (see Figure 17).

![Figure 17: Rotational releases](image)

From Figure 17 it becomes clear that if a hinged joint is to be simulated, the y and z axes need to be released. The branches are the only elements that are connected by a hinge as the floor is treated a fixed slab.

5.2.4 Constraints
Similar to the joints are the constraints, which simulate the anchor points of the construction to world or to other construction elements which are not included in the model. As stated before (paragraph 4.2), the stems are encastred at the base. To reflect this in the model the nodes that form the end of the stems are constrained in their movements. There are six possible constraints, x, y, z, xx, yy and zz. The first three are translation constraints and the last three are rotational constraints. To simulate a fixed base all six degrees of freedom of the nodes at the base of each stem, need to be constrained. Apart from the base the floor is also constrained in its movement as it is connected to the cores of the building. To simulate this, two of the four sides of the floor slab are constrained. As these connections are pinned only the translation freedom is constrained.

5.2.5 Loads
The last important aspect to model is the loads on the structure. These loads must be correct in type, placement, direction and magnitude. Figure 18 shows what loads work on the construction. The floors of the offices all have a distributed load of 5kN/m² which results in a point load of 525kN on each branch end. The first floor transfers its loads to the floor beams which results in a linear load of 7.3kN/m. Finally a gravity load is applied on the whole structure to take into account the dead load of the construction. All these loads are assigned a unique load case number.

As stated in the brief there are some safety factors to be considered. In accordance with the Dutch norms the dead loads are multiplied by 1.2 and the live loads by 1.5. This is done by creating a combination case from the individual load cases, for example:

Combination case 1- All loads: 1.5A1+1.5A2+1.5A3+1.2A4

In this combination case all the loads are added together and multiplied by their respective safety factors.
Figure 18: Loads
5.3 Results
The first thing that GSA will be used for is to get a general idea about the location and magnitude of forces and also the required section dimensions. The problem however is that the generated structures can differ quite a bit from each other. It is therefore difficult to come up with a 'one size fits all' solution. There is however need for a standard situation that will be sufficient in most cases. The base model will be discussed in paragraph 5.3.2. To deal with the exceptional situations and to gain insight in the way different generated structures compare in performance, three specific analyses will be done. Each one will look at a specific aspect of the construction and analyze the impact on the load distributions and the resulting internal stresses. This information will then help set the minimum and maximum values of the parameters of the generation tool and also improve the detailing of the tree structures. The three aspects that will be analyzed are eccentricity, branch angles and stem length. The graphs that show the results from the analyses are located in section I of the appendix.

5.3.1 Location and magnitude of forces
The model that was tested for this first analysis is shown below in Figure 19.

![Figure 19: Base model](image)

To understand how the loads are distributed throughout the structure we look at the graphs that show the axial forces, shear forces and bending moments. Figure 51, located in the appendix, show the axial forces in the structure. The result is not very surprising: the branches all have considerable compressive axial force ranging for a minimum of 988kN and a maximum of 1819kN. The stems bear of course a much higher load as there are often four branches attached to one stem. In the stems the minimum axial force is 1679kN and the maximum is 4030kN. While the branches and stems all have compressive axial forces, the floor beams mostly have tensile forces. These tensile forces range from 0 to 1059kN, the few compressive forces range from 0 to 474kN.

Figure 52, showing the shear forces, teach us that they are highest in certain specific stems. These stems all have in common that they have a large eccentricity. Paragraph 5.3.3 will
provide further insight into the effects of excentricity. The shear forces range from 507 to -594kN. Finally there is the graph of the bending moments, shown in Figure 53. Here we see a situation similar to the one depicted in the shear force graph. The largest bending moments are located in the stems with a large excentricity. The magnitude of these forces range from 1022 to -1724kN.

5.3.2 Required section dimensions

Now that is clear what the location and magnitude of the forces are in the construction, the shape and size of the construction elements of the base model can be determined. As stated in paragraph 5.2.2, two different steel types are used (S235 and S355). The S355 is used in the stems as they are subjected to the biggest loads. Based on the known loads and the steel type it is possible to select the size of the sections so that the yield stress of the steel is not reached except in a few specific places in the construction. These specific situations will be dealt with in the next three paragraphs.

In the brief (paragraph 3.2) it is specified that the branches and the stems must be circular in shape. The floor beams have no demands so they can be the standard I-profile shape. When we combine the section shape with the loads and allowed stresses it is easy to select the dimensions of the sections. Below a list of the chosen sections:

Floor beams: IPE450
Branches: Cir. 300x15mm
Stems: Cir. 500x20mm

Figure 54 shows the resulting stresses when the chosen sections are used. The green bars indicate that the stress is below 235kN/mm². Therefore the S235 steel type will suffice. The orange bars mark the area's where the stress exceeds the 235kN/mm² but stays under the 355kN/mm², the limit of the S355 steel type. The blue area's show the spots where the stress is even higher and thus where the chosen sections are inadequate. When we analyze these spots we see that they only occur around trees with a high amount of stem excentricity. This means that the base model works as long as these exceptional cases are either prevented or dealt with by locally assigning different sections. How this is done will be show in the next paragraph.

5.3.3 Excentricity

This paragraph will deal with the effect an excentric placement of a stem has to the load distribution in a tree. The excentricity is measured by the distance in meters, between the stem base and the optimal point, the intersection you get when you draw lines between opposite cloudpoints. This is shown in Figure 20; the red line is the distance between the actual stem location and the optimal location. This is the optimal point, because the reaction forces from the branches are in balance. To understand the influence of excentric stems, variants with increasing excentricity will be analyzed and compared.

Figure 20: Measure of excentricity

To properly analyze the effect we have to take in consideration that the effects are not local and bound to a single tree, but that they affect a larger area of the structure. For this reason three different situation were tested. I used a variation of the base model for each of these tests. (see Figure 21). In this variant all the stems where placed in their respective optimal points. This was done to isolate the effects of placing one or two stems excentric.
In this model three different situations are simulated: the excentric placement of one stem (see Figure 22), the excentric placement of two stems in a parallel fashion (see Figure 23) and finally the excentric placement of two stems in a perpendicular fashion (see Figure 24).
All three situations were tested with an excentricity ranging from 0 to 1.6m with increments of 0.4m. The individual results of each test can be found in section I.II of the appendix. These results are combined in Figure 25 shown below.

Figure 25: Excentric stem performance (single, double (parallel) and double (perpendicular))

The graph shows a couple of things. First is the linear relationship between the stem excentricity and the resulting stresses in the stems. When the stems are placed centric the stresses are around 190 N/mm² (resulting from the axial loads on the stems) and they increase at a rate of 35N/mm² per 10 cm as the excentricity is increased (as a result from the added bending moment).

The graph also shows the difference between moving two stems perpendicular to each other compared to parallel. In the first case the resulting stresses is only 5% higher than when only one stem is moved. When the stems are moved parallel the difference is much higher, up to 46%. This means that it is important to avoid outcomes of the script where there are large and/or multiple excentric stems in a parallel fashion.

Paragraph 5.3.2 looked at the required section dimension of the elements. The base model that was used, contained excentric stems and it is therefore interesting to determine the section demands in relation to the excentricity of a stem. Figure 26 shows the stress in the stems as a result of an increasing excentricity. The graph shows the results for three different stem diameters; 500mm, 750mm and 1000mm, all with a thickness of 20mm. The horizontal dashed line shows the maximum allowable stress when using S355 steel.

The results show that when the excentricity is small, up to 0.50m, the 500mm section is strong enough to withstand the loads. When the excentricity becomes larger, up to 1.20m, the 750mm section is enough. Finally, if the excentricity is larger then 1.20m, the 1000mm section has to be used. This section is able to withstand excentricities of around 2.40m.

Figure 26: Excentric stem performance (single, double (parallel) and double (perpendicular))
5.3.4 Branch angles

Apart from the stems, it is also interesting to look at the performance of the branches of the trees. As the branches are connected by hinges at both ends they are only subjected to a normal force. The value of this normal force is related to the loads coming from the floors and the angle that the branches make. The shallower the angle the larger the normal force in the branch. To find out how the structure performs with different branch angles a few different situations are tested. These situations are; single tree, all trees and single tree excentric. For all these situations the base model with centered stems was used. In this base model all the branch angles are around 48°. This is the angle that a branch makes with the stem. In the first situation the branch angle of a single tree is changed and the resulting normal force is calculated. The branch angle ranges from 40° to 80° with steps of 10°. Figure 27 shows the resulting trees.

Figure 26: Excentric stem performance (500, 750 and 1000mm)

Figure 27: Branch angles
The second situation is similar to first except that now the branch angle of all the trees is altered. In the last situation eccentricity of the stem is added to show the effect on the normal forces in the branches. In a tree with a single axis eccentric stem there are two sets of branches with different angles and lengths. To differentiate between the two sets one is called the long branches and the other the short. The long branches inherently have a shallower angle than the short branches and this is the angle that was measured (see Figure 27).

Figure 28 shows the results of the different situations. In all situations the minimum force is around 1200kN. From that starting point the axial loads increase exponentially as the angle is increased. For the single tree situation this means that at 65° the load has doubled (2400kN) and it has quadrupled (4800kN) at 80°.

The second situation, where all the trees are changed, the trend is comparable, only a bit steeper, resulting in a maximum load of 5500kN. From this we can conclude that it is not very important to limit the number of trees with shallow branches as they do not influence each other much.

The situation with the eccentric stems is a bit different. Here it is the stems with the steeper angle that have the highest load, the 'short' branches. This is because the loads are no longer distributed evenly over the four branches.

The next step is to translate these results to required section dimensions. For this the loads of the single tree situations are divided by the area of different sections. The sections that are tested are circular tubes of 300mm in diameter with a varying wall thickness ranging from 5mm to 20mm. From an architectural standpoint the branches were required to be the same diameter, which is the reason for varying the wall thickness. In paragraph 5.3.2 it was decided to use S235 steel in the branches. Figure 29 shows the graph with the results. From these results we can conclude that branch angle is not a very problematic factor in the structure. Only the very slim 300x5mm section is unsuitable, but the 300x10mm section is already capable of handling angles up to 65°. One section heavier this becomes 74° which is more than sufficient for the structure.
The last analysis revisits the stems of the trees and looks at the relationship between stem length and stress. It is already clear that the excentric placement of a stem results in a bending moment in the stem. This means that if the length of a stem would be increased, so would the resulting stresses caused by the bending moment. A quick analysis was run and Figure 30 shows the results that confirm this suspicion. As the stem length is increased the stress rises. The standard stem length that is used in the base model is 3m, this means that when the stem length is doubled to 6m the resulting stresses increase by 10%. When the stem length is halved to 1.5m the resulting stresses decrease by 30%. This shows the importance of keeping the stems short.
5.4 Conclusions and Recommendations

This paragraph will draw conclusions from the analyses that were done. I will list a series of recommendations for both the generation and the detailing tool. The initial analysis of the location and magnitude of forces and the required section dimensions showed that for the most part the structure is performing well enough. By using Cir. 500x20mm S355 steel sections in the stems, most of the large bending moments are dealt with. For the branches Cir. 300x15mm S235 steel sections suffice. The floor beams have to be IPE450's to be strong enough.

There are of course specific situations where these sections are insufficient. The biggest problem to overcome is the bending moments that are a result of excentric stems. These stresses increase in a linear fashion as the stem moves away from its most optimal point. When two or more stems are placed excentric, especially when they are moved in a parallel fashion, the bending moments become quite large. To counter this, it is advisable to detail different trees with different stem sections based on the excentricity differences. Trees with an excentricity of less then 0.5m are detailed with a 500mm stem, between 0.5m and 1.2m a 750mm stem is used, and for trees with an excentricity larger then 1.2m a 1000mm stem will be used. These values do however take into account that extreme, multiple stem in parallel fashion, excentricities are avoided.

Apart from the excentricity there is another aspect of the stems that needs to be taking into account and that is the stem length. The analysis showed that increasing the length will lead to greater stress caused by the bending moments. Therefore it will be very problematic to create trees with stems of more then 4m. The opposite is of course also true, shorter stems result in lower stresses, which can be used to reduce the effects of large excentricities.

Finally we come to the conclusions regarding the effect of the branch angle. The analysis showed that the initial choice of section (300x15mm), is capable of handling angle of up to 74°. This is more then sufficient for the structure and it means that in cases where the angle is less then 65° a 300x10mm section can be used.

The recommendations that were done in this paragraph will be used to define the respective minimum and maximum parameters of the generation tool and the section dimensions in the detailing tool.
6. Detailing
The last step in the process of generating a load bearing structure is the actual detailing of structure. This chapter will look at the detailing tool which takes the wireframe output of the generation tool and transforms it into a detailed structure. The goal is to end up with the specific information regarding each individual tree, which is needed to construct it in real life. In paragraph 6.1 we will look at the input of the tool which is the output of the generation tool. Subsequently in paragraph 6.2 I will discuss demands on the detailing. These demands come from four different areas: architecture, mechanics, fabrication and construction. The Paragraph 6.3 deals with the desired output of the tool. Finally paragraph 6.4 will look at the workings of the actual tool.

6.1 Input
The input for the detailing tool is similar to the input of the analyses. The output of the generation tool is baked which results in a wireframe model of the load bearing structure. Grasshopper component can then reference these lines and use them as a description of the shape of each tree. Apart from the shape of the tree the tool must also be inputted with the information of the details itself. This can also be done by referencing geometry from Rhino.

6.2 Demands

6.2.1 Architecture
The visual expression of the architectural design is organic on the large scale with smaller elements having a mechanical appearance. This mixture must also be present in the load bearing structure. The main elements of the trees, the stems and branches, must look organic, while the joints have a more mechanical expression. A good reference for this is the Southern Cross Station in Melbourne (Figure 31).

The organic look of the stem is achieved by using circular sections and using tapered shaped columns for the stems. The mechanical elements are introduced by leaving the connections exposed.

Figure 31: Southern Cross Station, Melbourne
6.2.2 Mechanics
The mechanical demands of the construction dictate the way the construction deals with loads. This includes the section dimension, joint types and fixtures to other construction elements. The section dimensions are determined by the results of the analyses and can be found in paragraph 5.4. One of the more complex aspects regarding the detailing tool is that it has to assign the correct stem size to each tree based on the amount of stem eccentricity and assign the branch section that corresponds to the branch angle. Paragraph 4.2 showed that the joints between the floor beams and the branches and between the branches and the stem are all hinged. This means that they also have to be detailed this way. The appearance of these joints is described in the previous paragraph. Finally there is the connection between the stems and the floor which has to be encastred.

6.2.3 Fabrication
Each tree differs from another. This makes it important to determine which parts of the tree are standard and which are custom made. In general a higher the level of standardization leads to lower costs. There are however gradations. It is for example costly to use beams with a lot of different section shapes but it is not very costly to have beams with a single section shape cut at different lengths. Another situation where individualization is not much more expensive is when the custom part is made up from standard elements which are combined differently in each case. Both these examples are used to keep the costs of the construction under control while maintaining the ability to create unique trees. The first demand is therefore that the number of different sections is limited, three for the stems and two for the branches. The length of these elements can vary as needed as this does not increase the costs dramatically. The second demand is that the detail of the joints is standard, which means that is has to be able to accommodate any required angle.

6.2.4 Construction
The last set of demands comes from the construction area, which deals with physical assembly of the construction on the site. As the trees are produced in a factory, they have to be transported to the site. This means that the maximum element size must not exceed the maximum standard transport size.
Once onsite the different parts must be assembled. This leads to the second set of demands: the construction order. Figure 32 shows the desired order in which the trees are assembled. First the stems are placed and secured to their foundation. After that the branches are attached and the floors beams between the branches are placed to hold up the branches. Finally the floor beams between the individual trees are added to complete the construction. For assembly of the parts it is advised that they be joined using nuts and bolts instead of welding. It is better to do any needed welding during production and not on site. The final construction demand is that the details incorporate the needed adjustment and alignment possibilities.

![Figure 32: Construction order](image-url)
6.3 Output
The output of the tool is a 3d detailing of the trees. As the tool runs in Rhino it is logical to gather all information here. The Rhino model can subsequently be used to steer the file-to-factory process. The computer file can directly drive the CNC machines that produce the different elements. If required it can also be used to generate work drawings and provide logistical information about the different elements.

6.4 Detailing Tool
6.4.1 Column design
The demands for paragraph 6.2 are analyzed and transformed into a design for the column. The design is shown below in Figure 33.

![Figure 33: Column design](image)

The design is a nice balance of both organic and mechanical elements, owing to its round stem and branches and its simple visual hinge connections. When we zoom in on the hinges we can see how the standardization is realized. The center cross at the top of the stem is made from four identical pieces which are welded together under the required angles of the tree (Figure 34). All the branch-ends at the center are also identical and join to the center cross with a simple pin connection. The same thing happens at the other branch-end: here it is joined to the floor beams, which have similar plates welded on them at the required angle (Figure 35). In total a single tree consists of three different elements for the joints, four branches with varying lengths but similar sections and finally a single stem element.
6.4.2 Grasshopper network

While the generation tool is run in the Grasshopper environment, its main functionality comes from the custom VB script that was developed. The detailing tool runs in the same environment, but does not need any custom functionality. It is set up the way most Grasshopper networks are: input is linked to existing Rhino geometry. This geometry is then used and altered, resulting in an output that can be 'baked' to produce physical Rhino geometry. An image of the detailing tool network can be found in the appendix section III.I. This paragraph will run through the different parts of this network, the location of these different parts are shown on the image of the entire network.

The network starts with the gathering of the input. As stated in paragraph 6.1 there are two different input elements: the wireframe dictating the tree shape and the detail geometry. Figure 36 shows the network of the first input. There are two curve components called 'stem' and 'branches' which reference the line that makes up the stem and the collection of lines that make up the branches. These four branches are then split into four streams which will allow the tool to produce four unique branches.

Figure 36: Input tree (1)

The second input element, the detail geometry, is first defined in Rhino (Figure 37). The hinges and the section for the branches are modeled so they can be used by the detail tool.

Figure 37: Detail geometry

Figure 38 shows the network part that calls on this geometry. A collection of point, BRep and curve components reference different parts of the geometry so that they can be applied to the different branches. The points are used as reference points for translation and rotation actions. The curves are placed at either ends of the branch and lofted between. Finally the BReps are the geometry of the hinges. This means that no geometry is related to the stem. The stem is made up from geometry generated by Grasshopper itself. The way this is done is shown in Figure 39. First the tool compares the endpoints of the branches to the location of the stem to calculate the eccentricity. This is done by a small VB script which can be found in the appendix (III.II). Next the tool selects the stem section that matches the found eccentricity. This is also done by a small VB script (III.III). Once the stem size has been determined it is used to create curves that are lofted to create the actual stem.

Figure 38: Input detail geometry (2)
Now the stem is created, it leaves only the branches to be detailed. The branches are created in a series of steps. They translate and rotate copies of the detail geometry to match the direction and length of the line that resembles the branch in the wireframe. The first step is the calculation of the horizontal orientation of each branch expressed in an angle (Figure 40). This is then also done for the vertical angle (Figure 41). These two angles, combined with the length of the branch, are then used to do the necessary translations and rotation of the detail geometry. This results in all parts of the branch (the center cross, the branch-ends at the top and bottom, the branch, and the joint at the floor beam) to be placed at the correct spot (Figure 42). The final result is then combined with all the other parts into one component that can be baked to create the geometry in rhino.
Figure 42: Branch creation (6)
6.4.3 Results and limitations

Figure 43 shows the resulting geometry from the detailing tool. The detailing tool creates a detailed tree around the lines of the wireframe. This brings limitations: it is not possible to detail a large number of trees at a time. To achieve this, the generation script needs to be altered so that it supplies information about which branches belong to what stem. This would however require a substantial rewrite of the script and was therefore not implemented in this research.

Figure 43: Result from the detailing tool
7. Application in the architectural design

The origin of the research into tree structures lay in the architectural design that was developed for the AE semester. This means that the two are closely related. Therefore it is very interesting to see what insight the research can feedback into the architectural design. This topic is discussed in this chapter.

Figure 44: Application in the conceptual design

7.1 The role of the tool in the conceptual design

Figure 44 shows the tree structures applied in the conceptual architectonic design. The implementation works well, the generation tool shows what is possible and what is not and the detailing tool saves a lot of time modeling all the trees. The tool performs as it was designed, but here lies a problem. What happens when the architectural result is not satisfactory?

One of the first things that became clear when the trees where implemented in the design was their visual repetitiveness. Even though none of the trees are duplicates, the similar stem heights cause all the trees to look alike. The problem is that the fixed stem height is hard coded into the generation tool. This situation, where the tool dictates the architecture, is very dangerous as it means that the tool and not the user is making the decisions. The solution for this lies in the development of the tool. It is not very difficult to implement a varying stem height (see example in appendix IV). That does however mean that either all the architectural desires must be known before creating the tool, or a cyclic process must be started that updates the tools as new demands or desires arise. This last option is of course the most flexible, but it does involve a lot more work.

Another aspect that creates the sense of repetition is the placement of the trees. They are currently placed in three rows and are all about evenly spaced. This is of course not directly caused by the generation tool as it is up to the designer to dictate the placement of the columns. The designer is in turn guided by the floorplan upon which the stems are placed. However the designer is also influenced by the limitations and capabilities of the generation tool.

The generation tool has a fixed set of rules that get applied in every situation, but there might be circumstances where these fixed rules might not result in the optimum solution. For this reason the user of the tool must be very aware of the way it works and what the implied limitations are. Only this way can he prevent a situation, as stated above, where the tool is making the decisions.

A last thing that doesn't look as imagined is the connection between the floor and the branches. This is mainly because the floor is just one big flat plane. This makes for a very
abrupt change from finely detailed to completely featureless. Adding more features to the floor is possibly the best way to solve this.
7.2 The role of the tool in the final design

7.2.1 Changes from conceptual to the final design
Like in any design process, the architectural design changed. The fear described in paragraph 7.1, that the tool would determine the architecture, was a valid one. To counter this and grasp control of the architectural expression of the load bearing structure the design was changed considerably.

The main problem with the load bearing structure as it was implemented in the conceptual design was that the trees were located on one level of the parking garage and thus hidden from the rest of the design. To solve this, the branching of the stems to a finer grid was spread over multiple floors up to the roof. By doing this the architectonic expression of the branching load bearing structure is experienced fully in the most important part of the building; the shopping arcade. As the branches now reach up to the roof, they have to shape themselves to the curved shape of the skin. This will help to lose the visual repetitiveness that was described in paragraph 7.1. To further enhance the expression, the branches were changed from straight to curved elements. To improve the structural performance of the curved beams they meet the neighboring beam at the end points, creating arches. A visual representation of this can be seen below in Figure 47.

These changes to the shape of the main load bearing structure mean that the generation tool, the analyses and the detailing tool have to be reevaluated and changed. The following paragraphs will look at what changes are needed. The tools will not be rewritten as that would take up too much time.

7.2.2 Changes to the generation tool
The input of the generation tool is based on two sets of points, the first to control the location of the stems, the second to control the grid that needs to be supported. This system can remain the same when using the new structure. The stempoints are used by the user to determine where the columns may stand. The cloudpoints come from dividing the surface that describes the shape of the skin.

The second aspect that can remain the same is the principle of a stem and branch combination. This means that there are controls to influence the number of branches and the height at which the stem branches.

The main difference between the two structure types on a...
scripting level is that in the new type every cloudpoint is supported by two branches that together form one arch. This means that there is little need to find the most efficient stem for a cloudpoint to connect to. There are hardly ever more than two options. This in turn means that the whole structure can be generated from the bottom up, instead of a bi-directional method (see Figure 48).

The main difference in geometric sense is the added curvature in the structure. The curvature of a branch is related to the height the branch must reach and the horizontal distance between the branchpoint and the cloudpoint. Therefore the generation tool must be able to analyze these values and adjust the curvature accordingly.

7.2.2 Changes in the analyses
The structural principles in the new structure are very different from the earlier structure type. Now that each branchpoint is supported by an arch made from two branches the resulting forces are completely different.

Figure 49 shows the static scheme for the construction. The floors now hang between the branches instead of resting on top of them. The beams that support these floors are able to act as tension rods in the arches. These tension rods take care of all the horizontal loads in the branches leaving the stems free of any loads other than the vertical loads. This means that the entire problem with stem eccentricity is resolved. Stem eccentricity proved to be the determining factor in the resulting loads in the earlier structure type. This means that this new type will have to be analyzed to find its determining factors (i.e. branch angle, stem length). There is of course also the need to analyze the construction to determine the required section dimensions.

7.2.3 Changes to the detailing tool
The changes to the detailing tool are the last step in the reevaluation. It takes the wireframe model of the structure and creates a detailed structure from it. The detailing tool has four factors to take into account: architectural appearance, mechanics, fabrication and construction.

The desired architectural appearance has already been made clear as it was the starting point for the new structure. The wireframe is clad with tapered steel circular profiles. Important detailing points are the attachments of the floor beams to the column and the connection between the branches and the stems.

From a mechanical standpoint it is important that the connections between the elements are as described in the static scheme (Figure 49). This means hinged joints at the branch- and cloudpoints and encastred at the base of the stems. This is similar to the earlier structure.

The main new aspect regarding fabrication is the added curvature to the tapered steel branches, possibilities regarding this have to be investigated.

Finally there is the construction of the structure. The construction order will be similar to the earlier structure; first the stems, then the branches and finally the floor beams. There is however a difference in the joining of the parts. The old structure relied on bolted joints while the joints in the new structure will be welded to achieve the desired architectural expression.

7.3 The role of parametric modeling in the architectural design process
7.3.1 Parametric modeling
Apart from looking at the direct relationship between the research and the architectonic design, it is also interesting to look with a larger scope. One of the main themes of the research is working in a parametric environment. It is interesting to investigate its effects on the architectonic design.
The need for a parametric environment came from the desire to create non-orthogonal architecture. As new computer aided manufacturing techniques emerged, the possibilities for non-orthogonal architecture grew rapidly. The old economic constraints, which forced buildings into orthogonal straight jackets, dissolved and gave way to new kinds of architecture. These new possibilities were very appealing, so I decided to investigate them in my graduation project.

The problem with non-orthogonal architecture is that repetition manifests itself differently from in standard architecture. Normally a design consists of a series of architectonical modules (columns, beams, facades elements) that are replicated to create the volumes and shapes that make up the building. Thus the building consists of a limited number of different parts. In non-orthogonal architecture these same modules exist but they are not simply copied to create the volumes. Instead these modules are assigned different parameters that are able to change on an individual basis. When a module is copied these parameters change to meet the specific requirements of each individual manifestation of the module.

Take for example a window module for a facade. In standard architecture this module is placed in the wall whenever a window is desired. For each type of window a new module has to be defined which means that the number of different windows is always limited to the number of defined modules. In non-orthogonal architecture the module is described by certain parameters like height, width, frame type, glass type etc. When such a module is placed in a wall it can be tailor fit to meet the desired needs of that particular window. This means that there can be an infinite number of different windows based on one single module. This freedom is in itself quite interesting but things become truly exciting when these parameters are driven by specific design ideas. In the case of the example, the size of the windows could be related to the amount of direct sunlight that reaches that part of the facade or could change over the length and height of the facade to create intrinsic patterns. It is easy to imagine the endless possibilities.

7.3.2 Application in the design

In the design there are a number of elements that were designed parametrically: the most important ones being the facade and the load bearing structure. The need to parametrically create these, came from the freeform shape of the design. This shape made it impossible to model things like the panels of the skin (Figure 50) or the columns of the load bearing structure by hand.

The way the initial load bearing structure was created, has been dealt with in the previous chapters. The load bearing structure of the final design was generated completely in Grasshopper as there was no time to rewrite the generation and detailing tools. This means that the idea of a self optimizing script was lost.

In the case of the skin a plug-in for Rhino called Paneling Tools was used. First a panel shape is defined which is in turn populated over the facade. This means that in both cases the parametric aspect was only used to create geometry that would be very labor intensive to do by hand. Automating these labor intensive steps is important to speed up the design process. Each time the design is changed certain elements have to be remodeled which takes time. By automating certain steps the iteration speed can be increased which means that more design cycles can be made in a given timeframe.

Looking back on the design process it becomes clear that this has been a very valuable aspect. The design has undergone a number of radical makeovers which would have taken too much time had the design not be modeled in a parametric environment.

Figure 50: The skin
On the other hand it does mean that the full benefit of parametric modeling has not been exploited in the design. The end result in both the facade and the load bearing structure is dumb geometry. This is a shame as a parametric environment offers a lot more possibilities. The load bearing structure could have reacted to its internal forces, as was the original plan and the outer facade could have reacted to the noise nuisance generated by the highway.
8. Conclusions
This research was set out to answer the following research question:

_How can a non-orthogonal load bearing structure be automatically generated taking into account the criteria set by the brief, the architectural design and the structural performance requirements?_

To answer this question the research had to deal with three different aspects. The first is related to the automatic generation part of the question. A method had to be developed that enabled a computer to generate an organization of elements that form a load bearing structure. The automatic generation meant that algorithms had to be developed that layed down rules and instructions for the computer to follow. The resulting load bearing structures are therefore only as good as the algorithms that create them.

This leads us to the first conclusion: developing good algorithms is hard. From the beginning it quickly became clear that there were a lot of variables that had to be taken into account and that it is difficult to determine their importance or hierarchy. Combine this with the difficulty of learning to program at the same time and things quickly become quite complicated.

To deal with this it was important to reassess the expectations. By limiting the number of variables and simplifying the algorithms it was possible to get some decent results. The generation tool had showed that it is possible to create a tool that helps a designer with specific tasks.

The other two aspects that were dealt with, lay in the realm of the load bearing structure part. The first is analysis of the structural performance of the generated load bearing structures. To create a good generating algorithm and to validate the results it was important to analyze the generate structures. Running these tests proved to be a valuable part of the research process as it gave insight into the performance of the structures and uncovered flaws in reasoning. There are two important aspects to running these structural analyses; modeling and interpreting. When the situations are not modeled correctly the outcome is worthless so it was important to get that right. This took a bit longer than it should have, because the analysis program (GSA) was unfamiliar to me. In the end though, I am confident that the models are correct and the results are valid.

When it comes to interpreting these results it became clear that there was a little bit more to it than simply checking a few numbers. Going into the results, looking at moment distribution diagrams, etc, proved to be a very important step in understanding what goes on in the structure. It is however clear that to make a more trustworthy analysis more knowledge about structural mechanics is required.

The last aspect concerned the detailing of the construction and is the final step in creating a complete load bearing structure. This aspect had the potential to become very large and overwhelming. There were a lot of different areas with their specific demands that had to be incorporated into the detailing. The main focus was however on the automatic generation of the detail. To succeed in this the situation had to be simplified in a similar method, used with the generation tool. It was just too much to incorporate all the demands. Especially the area of file-to-factory was not explored to the fullest which is a shame as it is an important part in creating these parametric details. Also the fact that the detailing tool is only capable of detailing one tree at the time is a shame. This means that there is still a lot of manual labor when generating the complete load bearing structure which is something this tool is trying to decrease. The detailing tool is however a proof of concept that, given enough time and insight, it is possible to generate these parametric details of a load bearing structure.

All in all the research showed a simple but effective approach to automatically generate a non-orthogonal load bearing structure, taking into account the criteria set by the brief, the architectural design and the structural performance requirements.
Literature


Hollander den, J., "Slankerbouwen met hogesterktestaal" *Bouwwereld*, blz 24, nr 19, 2004


Appendix

I. Results

I.1 Required section dimensions

Figure 51: Axial forces

Figure 52: Shear forces
Figure 53: Bending moments
Figure 54: Base model - Stresses
I.II Excentricity

Single excentric stem placement

Figure 55: Single - 0m - Myy

Figure 56: Single - 0.4m - Myy
Figure 59: Single - 1.6m - Myy

Double excentric stem placement (parallel)

Figure 60: Double (parallel) - 0m - Myy
Figure 61: Double (parallel) - 0.4m - Myy

Figure 62: Double (parallel) - 0.8m – Myy
Figure 63: Double (parallel) - 1.2m - Myy

Figure 64: Double (parallel) - 1.6m - Myy
Double excentric stem placement (perpendicular)

Figure 65: Double (perpendicular) - 0m - Myy

Figure 66: Double (perpendicular) – 0.4m - Myy
Figure 69: Double (perpendicular) – 1.6m - Myy
II. Generation Tool script

271 Private Sub __Internal_RunScript(ByVal stem_start_pt_list As List(Of On3dPoint), ByVal cloud_pt_list As List(Of On3dPoint), ByVal max_angle As Integer, ByVal num_nodes_stem As Object, ByVal first_stem_length As Object, ByVal sec_stem_length As Object, ByVal max_branch_length As Object)
272 'Custom code not written and not copyrighted by Robert McNeel & Associates
273 '<custom code>
274 Dim stem_conn_pt_list As New List(Of On3dPoint)
275 Dim branch_l_list As New List(Of OnLine)
276 Dim branch_valid_l_list As New List(Of OnLine)
277 Dim branch_best_l_list As New List(Of OnLine)
278 Dim stem_l_list As New List(Of OnLine)
279 Dim stem_full_l_list As New List(Of OnLine)
280 'Const first_stem_length As int32 = 20
281 'Const sec_stem_length As Int32 = 6
282 'Const num_nodes_stem As Int32 = 3
283 'Const max_branch_length As Int32 = 20
284 Dim pt As On3dPoint
285 Dim i As Int32
286 Dim ptc As On3dPoint
287 Dim distance As Int32
288 'Create all nodes on the stems
289 'For Each pt In stem_start_pt_list
290 Dim stem_prev_pt As New On3dPoint
291 stem_prev_pt = pt
292 For i = 0 To num_nodes_stem - 1
293 Dim stem_pt As New On3dPoint
294 stem_pt.x = pt.x
295 stem_pt.y = pt.y
296 stem_pt.z = pt.z + first_stem_length + i * sec_stem_length
297 stem_conn_pt_list.Add(stem_pt)
298 stem_l.from = stem_pt
299 stem_l.To = stem_prev_pt
300 stem_l_list.Add(stem_l)
301 stem_prev_pt = stem_pt
302 Next
303 Dim stem_full_l As New OnLine
304 Dim stem_full_pt As New On3dPoint
305 stem_full_pt.x = pt.x
306 stem_full_pt.y = pt.y
307 stem_full_pt.z = pt.z + first_stem_length + i * sec_stem_length
308 stem_full_l.from = pt
309 stem_full_l.To = stem_full_pt
310 stem_full_l_list.Add(stem_full_l)
311 Next
312 Dim ptc As On3dPoint
313 Dim distance As Int32
314 'Create all valid branches from the point of view of the cloud_points
315 'Valid = length < max and angle < max
316 'All branches are added to a list
317 'For Each ptc In cloud_pt_list
318 For Each pt In stem_conn_pt_list
319 distance = pt.DistanceTo(ptc)
320 If ptc.z > pt.z Then
321 If distance < max_branch_length Then
322 Dim branch_l As New OnLine
323 branch_l.from = ptc
324 branch_l.To = pt
325 branch_l_list.Add(branch_l)
326 End If
327 End If

62
End If
End If
Next

' Check how many branches connect to each node
Dim l As OnLine
Dim branch_count As Int32
Dim cloud_count As Int32

For Each pt In stem_conn_pt_list
  Dim lastCloud_pt As New On3dPoint
  branch_count = 0
  For Each l In branch_l_list
    If l.To = pt Then
      branch_count = branch_count + 1
      lastCloud_pt = l.from
    End If
  Next

  If branch_count > 1 Then
    For Each l In branch_l_list
      If l.To = pt Then
        Dim l_ok As New OnLine
        l_ok.from = l.from
        l_ok.To = l.To
        branch_valid_l_list.Add(l_ok)
      End If
    Next
  Else
    If branch_count = 1 Then
      Dim l_toch_ok As New OnLine
      l_toch_ok.from = lastCloud_pt
      l_toch_ok.To = pt
      branch_valid_l_list.Add(l_toch_ok)
    End If
  End If
Next

' Filter all valid branches from the point of view of the trees
' Valid = more then one branch at each node or when it is the only branch of a point
If branch_count > 1 Then
  For Each l In branch_l_list
    If l.To = pt Then
      Dim l_ok As New OnLine
      l_ok.from = l.from
      l_ok.To = l.To
      branch_valid_l_list.Add(l_ok)
    End If
  Next
Else
  If branch_count = 1 Then
    Dim l_toch_ok As New OnLine
    l_toch_ok.from = lastCloud_pt
    l_toch_ok.To = pt
    branch_valid_l_list.Add(l_toch_ok)
  End If
End If
Next
' Create the value rating for each branch and pick the best one

Const STARTBESTVALUE As Int32 = 10000
Dim bestFoundValue As Double
Dim foundValue As Double
Dim bestFoundLine As New OnLine
Dim line As New OnLine

For Each ptc In cloud_pt_list
    bestFoundValue = STARTBESTVALUE
    For Each line In branch_valid_l_list
        If line.from = ptc Then
            foundValue = valueBranch(line)
            If foundValue < bestFoundValue Then
                bestFoundvalue = foundvalue
                bestFoundLine = line
            End If
        End If
    Next
    If bestFoundValue < STARTBESTVALUE Then branch_best_l_list.Add(bestFoundLine)
Next

A = stem_conn_pt_list
B = branch_best_l_list
C = stem_l_list
D = branch_l_list
E = stem_full_l_list

' </custom code>
End Sub

' </Custom additional code>

Private Function valueBranch(ByVal aBranch As OnLine) As Int32
    Dim len As Double
    len = aBranch.Length
    Dim cosValue As Double
    cosValue = (aBranch.from.z - aBranch.To.z) / len
    Return abs(cosValue - Cos(0.01745 * 25)) * len
End Function

' </Custom additional code>
End Class
III. Detailing Tool

III.I Detailing Tool network
III.II Excentricity calculation script
Private Sub RunScript(ByVal cloud_pt_list As List(Of On3dPoint), ByVal stem_pt As On3dPoint, ByRef A As Object, ByRef B As Object)
    Dim i As New Int32
    Dim j As New Int32
    Dim r_pt As New On3dPoint
    Dim performance As New Double
    Dim dir_l As New OnLine
    i = 0
    j = cloud_pt_list.Count()
    While i < j
        r_pt.x = r_pt.x + cloud_pt_list.Item(i).x
        r_pt.y = r_pt.y + cloud_pt_list.Item(i).y
        i = i + 1
    End While
    r_pt.x = r_pt.x / j
    r_pt.y = r_pt.y / j
    performance = r_pt.DistanceTo(stem_pt)
    dir_l.Create(stem_pt, r_pt)
    A = performance
    B = dir_l
End Sub

III.III Stem selection script
Private Sub RunScript(ByVal stem_perf As Double, ByVal dia_min As Double, ByVal dia_max As Double, ByRef A As Object)
    A = dia_min
    If stem_perf > 0.4 Then
        A = 0.6
    End If
    If stem_perf > 0.8 Then
        A = dia_max - 0.25
    End If
    If stem_perf > 1.2 Then
        A = dia_max
    End If
End Sub
IV. Generation tool variant: Varying stem height

Section of the script that generates the varying stem height based on the average distance between the stem point and the cloud points. A higher average distance means that the cloud points are further apart resulting in shallower branches. This means that if the stem length of this tree is lowered it has a positive effect on the branch angles.

....
'create the branch point
    Dim branch_pt As New On3dPoint
    Dim avg_dist As New Double

    avg_dist = 0
    branch_pt.x = stem_pt.x
    branch_pt.y = stem_pt.y

    For j = 0 To tree_cloud_pt_list.Count() - 1
        avg_dist = avg_dist + stem_pt.DistanceTo(tree_cloud_pt_list(j))
    Next

    avg_dist = avg_dist / tree_cloud_pt_list.Count()  
    'MsgBox(avg_dist)
    'branch.pt.z = stem.pt.z + min_stem_height + (-1 * max_added_stem_height / (max_influence - floorheight) * avg_dist) + max_added_stem_height * (max_influence / (max_influence - floorheight))
    branch.pt.z = stem.pt.z + min_stem_height + (max_added_stem_height / (2 * avg_dist - magicnumber))
    'Create the stem of the tree
    Dim stem_1 As New OnLine

    stem_1.Create(stem_pt, branch_pt)
    stem_1_list.Add(stem_1)
...