Mat-2.4177 Seminar on case studies in operations research: Generating aesthetically pleasing lattice structures

Teppo-Heikki Saari // tisaari@cc.hut.fi, project manager
Jarno Leppänen // jarno.leppanen@tkk.fi
Johan Mangs // kmangs@cc.hut.fi
Teemu Mutanen // teemu.mutanen@gmail.com
Antti Savelainen // antti.savelainen@tkk.fi

May 9, 2008

Final Report

Contents

1 Introduction

1.1 Background

Arts have always depicted the phenomena and the issues of contemporary society. Since the beginning of mankind arts have always played an important role in expressing human (sub)consciousness. The scientific and technological development of modern society has prepared the way for the convergence of arts and technology.

The development of proper software and tools for creating something new has made it possible for arts to expand into digital form, and many of the art forms that we see today simply did not exist a couple of decades ago. The advances in software technology has also changed the way artists working in the fields of traditional arts, painters and sculptors to name but a few, work.

Sculptor Anna-Kaisa Ant-Wuorinen was given the job of designing a sculpture

Figure 1: "Fuuga"

for supporting structure next to Sandels building – located in Töölö, Helsinki – for its stock pile. The work is an example of combining architecture and engineering. Named after the musical instrument, "Fuuga" (see Figure 1) is made of brushed stainless steel tubes. The scaled model for the piece was designed with Rhinoceros software and the parts were weld together. Both methods were needed, because as much as exploring the dimensions of sculpting are about the final resulting art work, it is about how the sculpture is made.

Professor Ahti Salo at The Systems Analysis Laboratory in Helsinki University of Technology was contacted by Ant-Wuorinen to elucidate whether the stu-

dents had ideas for possible solutions to a problem arising in the process. The sculpture was to fulfill certain requirements. The methods of creating sculptures mentioned above were iterative, and the sculptor could not see the result beforehand. If there was a way of constructing the model fulfilling the requirements and then evaluating it digitally, the process would become easier. In addition, during the process several interesting questions concerning the aesthetics of the sculpture arose:

- What is mathematematcal beauty?
- What is technical beauty?
- What is aesthetical beauty?
- How can the questions above be applied to the sculpting process?
- Can one examine the essence and the fundamental differences and similarities of above concepts through a concrete example like creating a sculpture?

This project tries to provide means and methods of answering the aforementioned questions by developing a tool for generating lattice structures bearing a close resemblance to "Fuuga".

1.2 Project goals

The goal of the project is to develop a tool for Anna-Kaisa Ant-Wuorinen that aids her to create aesthetically pleasing lattice structures for her sculptures. The evaluation of qualitative properties in the sculpture is possible only if the artist has a chance in conceptualizing the final lattice structure. The visual model produced by the software is one way of examining the properties of lattice structures without going through the somewhat tedious process of manufacturing the physical model by hand.

The developed tool should be able to visualize the generated lattice models. It is required that the software generates the lattice structures in such a way that the mathematical minimum requirements are fulfilled. Secondly, the resulting lattices have to be shaped so that they are easily grasped visually. And thirdly, the user should have a way of influencing the result by giving input to the software.

The project was divided into two major parts. The first part consists of developing the algorithm for generating the lattice structures fulfilling the required constraints. In the second part the resulting lattices are to be examined. By identifying the properties of the lattices and describing them mathematically, the algorith is to be made more sophisticated. The user may give input to the program, and the resulting lattices express various properties demanded by the user.

Furthermore, the essence of the combination of arts and technology is examined by comparing the properties of created models. Mathematically aesthetic solution could be a more simple one than the stable and practical technological

approach. Classical artistic solutions could be symmetrical and harmonized. In that way, measurable properties can be manifested in the language of mathematics and physics, thus creating the shape of a lattice.

1.3 Project outline

We have resorted to examining parallelepiped lattice structures of size 3x4x7. The structure generated by the software is required to fulfill at least the following conditions:

- The structure is completely connected and solid.
- There is a junction in every panel point. The structure must maintain the rectangular parallelepiped form.
- There is not a single node that is connected to only one other node. This means that there is no bar that has one end point connected to a bar and the other is not.
- The projection of the structure by all three coordinate axis must form a complete grid. All of the nodes on the projection plane are connected to adjacent nodes.

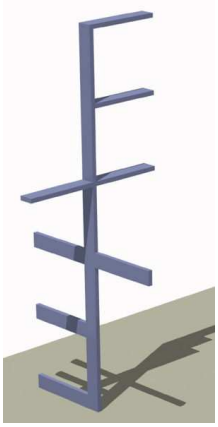

Figure 2: All the possible junction types

The lattice consists of bars that may have two orientations, vertical and horizontal. The cross-section of the bars is rectangular. This gives rise to another condition: the bars can be connected in only one plane (see Figure 2).

The apparently small size of the lattice structure still permits a vast number of different solutions, even with all the constraints applied. This poses a great challenge to the formulation of the algorithm.

2 Literature review

In the European culture, systematic studying of the philosophy of arts and aesthetics was introduced by Plato around 400 BCE. Plato finds that (within visual arts) all the pieces of art merely look like the original. If they were perfectly the same, they would be another examples of the same thing. This theory is easily applied to our project. During this project we have been interested only in the shape of the lattice. When the shape is expressed in the language of mathematics, the expressions are identical - an example of the idea of lattice.

Plato supposes that, in a sense, "all created things are imitations of their eternal archetypes, of forms". Plato sees art to be imitation in a narrower sense; arts are removed from "the reality of the forms, on the lowest of the four level of cognition, eikasia (imagining)". In this sense, lattices formed by means of mathematics could be somehow imitating the essence of the rules and exact truths of mathematics. Hence, there are utopian solutions which can be solved computationally, and this concrete solution is a rendered "image" of the original one.

By using mathematical approaches, we are able to uncover a mathematically perfect and pure solution, but it is subjective. Undefined are questions like "what is aesthetically and artistically pleasurable?". It is important to emphasize this point, because by learning a human being will be able to play with and utilize this software like an instrument. When these two approaches are both adopted, one is able to obtain better 'imitations of eternal architypes'.

Elsewhere, mathematical approaches within arts have been used at least in architecture. In St. Louis a famous monument called Gateway Arch (see Figure 3, [7]) was built in 1965. It is designed by Eero Saarinen. This curvilinear structure is known for its shape, that of which a free-hanging chain takes when held at both ends.

Figure 3: Gateway Arch in St. Louis designed by Eero Saarinen

The golden ratio is defined between two qualities "if the ratio between the sum of those quantities and the larger one is the same as the ratio between the larger one and the smaller". It has been used widely in arts since the renaissance. A famous example of the application of the golden ratio is Mona (Figure 4, [6]) Lisa. "Mona Lisa's face is a perfect golden rectangle, according to the ratio of the width of her forehead compared to the length from the top of her head to her chin."

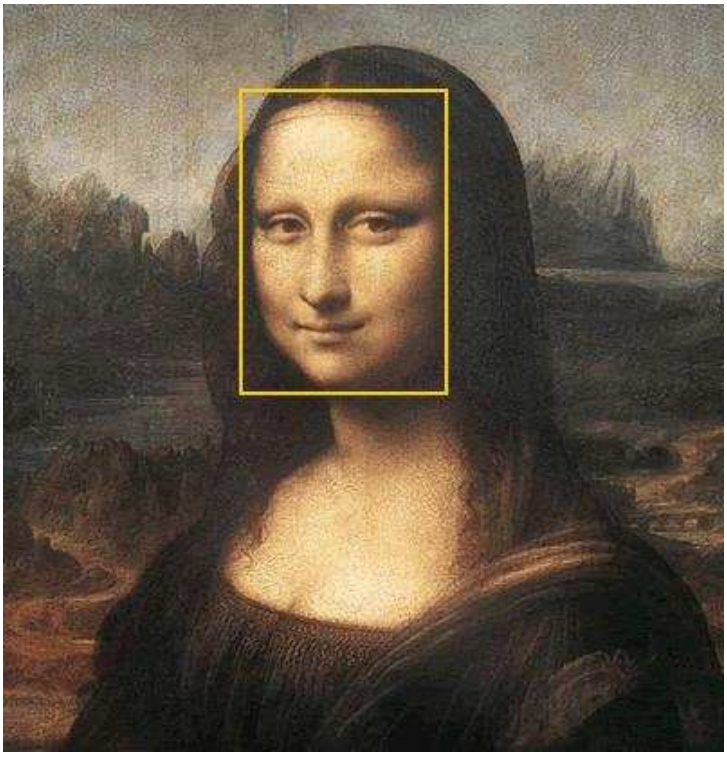

Figure 4: The golden ratio used in Leonardo Da Vinci's Mona Lisa

In academic literature the connection of art and mathematics is scattered and varied. The viewpoints from which the mathematics may be analyzed in art are numerous. These are for example how mathematics inspires art, how mathematical patterns or transformations can be seen as art, or even how mathematics itself can be seen as art. Some of the themes close to our work are studies in symmetry and projection. For example Doris Schattschneider discusses in [1] how mathematics itself generates art. Schattschneider presents how the patterns in mathematics can generate artistic patterns. Schattschneider discusses also the reserve effect, where art illuminates mathematics. These viewpoints include the idea that there is a concept called art which is in focus. In our work we focus on the tools to aid artistic design process. Today with digital technology these tools are becoming more sophisticated and thus more helpful to ordinary artists.

In literature there are very few studies on mathematical tools to help artistic design process. One previous study is presented by Sara Robinson [2] where

she discusses about the application of existing tools. The drawback in academic viewpoint is that in this case that it does not involve any new mathematics. The research viewpoint for the tools is also different, already made art is analyzed by mathematical tools and the question is what do the tools highlight. Joseph Malkevitch [3] discusses the associations between mathematics and some of the chosen arts (i.e. music, dance, painting, architecture, sculpture). His discussion focuses on projection and how some objects are seen in perspective. This is close to our work, although we view the projections as restrictions.

3 Methods

3.1 Data structures

3.1.1 Tensor

The need for applicable data structure for the lattices is self-evident for the algorithm to be able to present adequately the lattices and to compute different properties. The first data structure used was four dimensional tensor. Each of the lattice points is represented by four variables. Three of the variables are the coordinates x, y and z of the lattice point. The fourth variable represents the orientation of the bar. The possible values of the fourth variable are (see Figure 5):

- -1, the normal vector of the wider side of the bar points towards the next dimension in order, the order of the dimensions being x $\rightarrow$ y, y $\rightarrow$ z, z $\rightarrow$ x
- 0, there is no bar
- 1, the normal vector of the wider side of the bar points towards the previous dimension

and the values are mapped to orientations according to Figure 5. If the dimensions of a lattice are (a, b, c) (corresponding x, y, and z respectively), the data structure for a lattice of above size is of size (a+1, b+1, c+1). This leaves us with ab + bc + ca extra lattice points that all have value 0.

3.1.2 Directed graph

After having created the tensor data structure it was time to develop the routines for checking whether the lattice fulfills the requirements. Some of the required properties were fairly easy to implement, but the need for the structure to be connected turned out to be extremely challenging. A different approach was needed. After examining our options, graph theory was found out to be the most suitable for the purpose.

Graph is a more simple object than tensors. In addition, Matlab is tailor-made for vector handling. First a lattice is divided to nodes and directed edges from an exactly numbered node to another. Then, a matrix M ('number of nodes'-by-'number of nodes') is composed and set full of nulls. In case a bar goes from a row (m) to a column (n), set M(m,n) = 1. The orientation is determined by the order of m and n. If m ¡ n, the orientation is 1 and -1 otherwise.

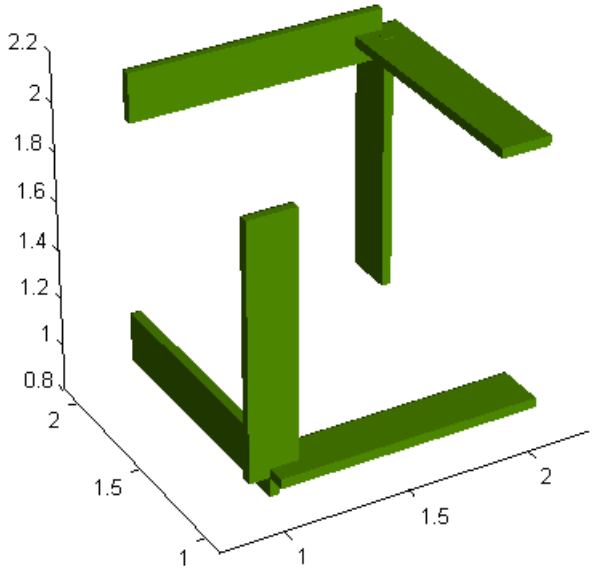

Figure 5: Different orientations of bars. The lower bars have value 1 using the tensor representation. The bars above have a value of -1.

The lower bars have value 1 using the tensor representation. The bars above have a value of -1. Figures 5 and 6 try to clarify the situation. Suppose we have a lattice of size (2, 2, 2). First, the values for the orientations of different bars needs to be agreed upon. This is shown in Figure 5. The lower three bars have value 1 in the tensor representation. The upper bars have value -1.

Because the graph is represented by a two-dimensional matrix, we need to fix a certain way of indexing the nodes, otherwise we would not be able to distinguish between the nodes. The formula for indexing is

$$\text{index} = i + (j-1) \cdot a + (k-1) \cdot a \cdot b, \quad i \in [1,a], j \in [1,b], k \in [1,c], \qquad (1)$$

where i, j, k are the 3-dimensional coordinates for the node, and a, b, c are the dimensions of the lattice. There are as many indices as there are lattice points, that is a $\times$ b $\times$ c.

Now we have indices for the graph that is represented by so-called 'adjacency matrix'. The direction in the adjacency matrix is represented in the same way as we would represent transfers between states in stochastic processes. Figure 6 shows how the arcs in the adjacency matrix are indexed corresponding to the bars in Figure 2. The elements in the matrix correspond to bars. For example, the bar going in the direction of z-axis from node 1 to node 5 is shown in Figure

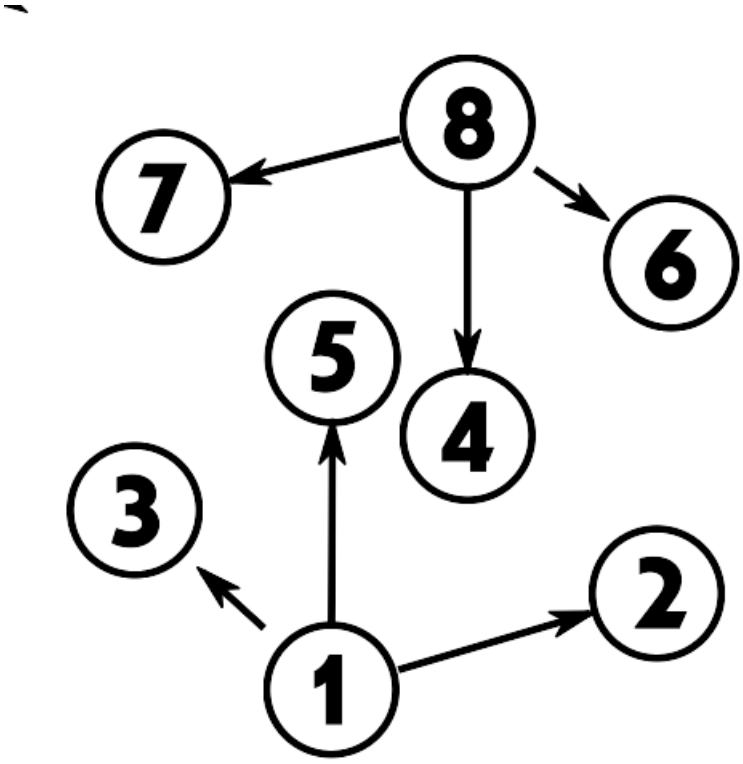

Figure 6: The arcs corresponding to the bars in Figure 5

5 by having a certain orientation. In the adjacency matrix the node is found in the first row, fifth column. If we were to have the same orientation as the bar connecting nodes 4 and 8, the number one would have been found from the fourth row and first column. The adjacency matrix corresponding to Figure 6 is

$$\begin{matrix} 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{matrix}$$

When we have big structures, the adjacency matrix is sparse.

3.2 Generating lattices

In the previous chapter the adjacency matrix was presented. This chapter will explain how the adjacency matrix is used when the tool generates structures.

The basic idea is simple, the method starts from one simple structure and edges are added to the existing structure in each step. The method is described in the following in more detail.

For the structure to be complete, the lattice is required to fulfill the conditions presented in the chapter 1.3. The first requirement, i.e. the structure is completely connected and solid, is taken into consideration by the method during the process. In graph theory there are available methods which can check for example if all the nodes are connected in a given graph [5]. There is no need to use any of these methods here because the method presented here is iterative in nature, thus the lattice is solid and connected if the base structure is solid and connected. The lattice is generated iteratively as follows:

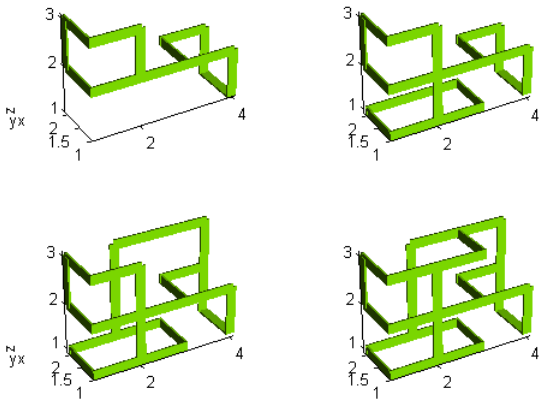

Figure 7: Four different lattices to represent the idea of the lattice generation process. In each step a lattice is added to the existing structure if the structure remains solid and connected after the lattice addition.

The method generates randomly two connected circular lattices. There is an example shown in the Figure 7. The Figure 7 shows four steps from the iterative process. These two connected lattices can be seen in the upper left corner in the Figure 7. Each of the following steps includes an addition of one randomly generated lattice to this existing lattice. This can be seen in the upper right lattice in the Figure 7. The two bottom lattices in the Figure 7 shows the two following lattice additions in the process.

There are two ideas implemented in the process which keep the structure solid and connected. The first one is that each edge can only be added to the existing lattice in correct orientation. This is needed to fulfil the second condition of a given lattice, i.e. there is a junction in every panel point. When each edge is added in correct orientation there is no need to revise the structure afterwards. The second idea implemented in the process is to formulate the lattices possibly added as such that the structure is solid and connected at all times.

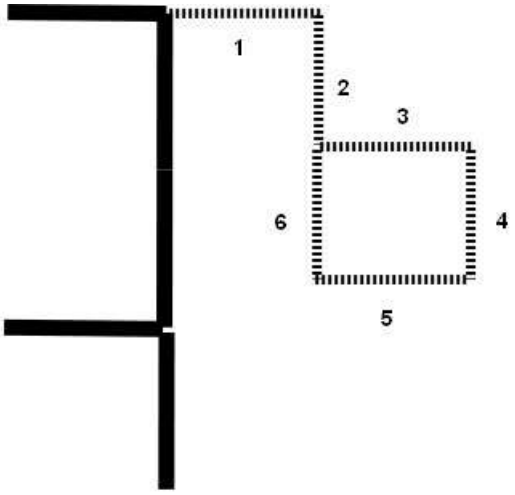

Figure 8: A diagram representation of the lattice generation process. The generated lattice ends up to itself, i.e. the edge 6 connects to the node connecting 2 and 3. The lattice is added to the existing structure.

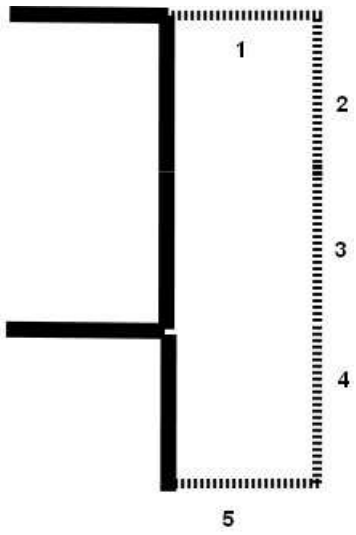

Figure 9: A diagram representation of the lattice generation process. The generated lattice ends up to the existing structure, i.e. the edge 5 connects to the existing structure. The lattice is added to the existing structure.

This second idea can be illustrated by Figures. In the Figures 8, 9, and 10 is presented simplified version of the lattice addition. The solid black edges in each

figure represent existing structures and the dotted edges represent possible additions. The dotted edges are numbered to represent the addition order. Because the added edges are selected randomly, and generated as long as there are possibilities to add another, there are three types of lattices that may be considered. In the first scenario, presented in the Figure 8, the lattice is considered added in the corner of the existing lattice, and the edges are randomly generated until the lattice ends up to itself: the edge 6 connects to the node connecting edges 2 and 3. This will stop the generation process and the lattice is added to the existing structure. The second type, presented in the Figure 9, represents the situation where the added lattice ends up to the existing structure: the node 5 connects to the edge in existing structure. The third type, presented in Figure 10, represents the situation when the last edge in the generated lattice is connected in only one other edge: the edge 5. If this case happens then the generated lattice is discarded and another lattice is generated.

The method adds lattices to the existing structure as far as no additional edge

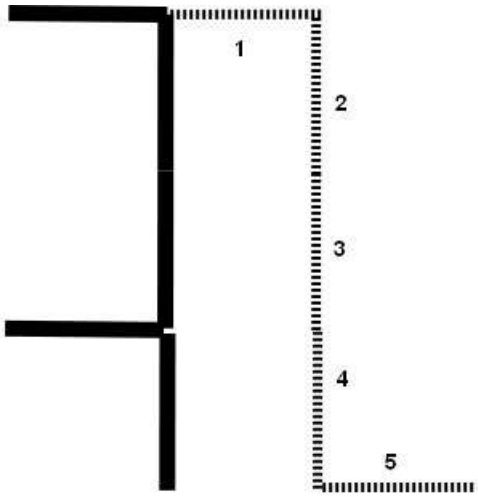

Figure 10: A diagram representation of the lattice generation process. The generated lattice is blocked, i.e. the edge 5 connects to only one other edge and no other edges can be added. The lattice is discarded.

can be added to the structure. All of the additions must fulfill the two requirements, solid and connected structure and orientation of the edges.

After the iterative process is finished and the structure is complete, the two other conditions are revised. The method to check if all of the corners of the structure have edge is simple. The other condition to be revised is the projection from each perspective. The lattice is generated in three-dimension space, by projection the lattice can be interpreted as two-dimensional lattice and viewed from three different viewpoints. The viewer in each of these viewpoints must see the lattice as whole otherwise the structure is not complete. If the generated structure does not fulfill this projection and viewpoint condition, the structure is discarded and another structure is generated.

The method does not guarantee the projection and viewpoint condition to be fulfilled. The method allows an implementation during the edge generating process that the projection conditions would more likely to be realized. This is possible to implement in the process such that before adding an edge the lattice is calculated in a way that the directions in which the edge can be added are scored. This scoring method would then tell which one of the directions will add most to the probability of the projection conditions to be fulfilled. This scoring function may also be implemented as probabilities for the directions.

3.3 Visualization

An essential part of the project was to develop a functional visualization routine. Not only because this was a requirement for the tool under construction, but because it would also serve as an excellent aid in developing the tool itself. Thus, some type of visualization routine had to be developed early on. However, as early versions of this routine would be exclusively for internal use, little effort was placed on making these early visualizations aesthetically pleasing.

Since Matlab is used to develop the algorithm itself, a logical consequence was to also produce the visualizations using Matlab. The first version of this routine visualized the lattice structure as a simple 3-dimensional wire-frame. This gave us a rough idea of what could be achieved and how it could be used. However, this wire-frame method was unable to visualize the bars as 3-dimensional objects, and thus unable to correctly reflect the true appearance of our lattice structures. Nevertheless, the visualization method itself was found to be a viable solution and this lead to the development of our next version. This version was able to correctly visualize the bars as 3-dimensional objects, and thus it was of great help when checking the correct functionality of the algorithm and evaluating its results. The next step was to complete this routine, because at the moment, this code was still unable to determine the correct lengths of the bars. Consequently, this resulted in unwanted gaps between some of the junction types. After correcting this last inaccuracy an accurate visualization routine had been produced. However, to further enhance the visual representation of our solutions, a script was created for transferring our solutions into a format that could be read by Blender - a free 3D graphics program.

3.4 Linear programming model

The problem may also be described as an integer linear programming problem (ILP) or, more precisely, as a 0-1 or binary integer programming problem (BIP). In this formulation the existence of bars in the structure is described as a set of binary decision variables and a set of linear constraints denote the set of feasible structures. We then try to optimize the structure by minimizing a linear objective function which tries to measure the 'beauty' of the structure. This function may be trivial, such as the number of bars in the structure, or a weighted combination of linear measures of the structure's "aesthetic properties".

Although the number of variables and constraints becomes rather challenging especially in case of targeted problem size (1000 variables and 4000 constraints, not including connectivity constraints), the problem may be solved in minutes

with certain objective functions and by relaxing some constraints. The problem formulation is also quite natural, as we may describe some holistic constraints which cannot be addressed easily in iterative generative approaches, such as projection constraints. What is more, by studying the set of feasible structures we may determine which structure sizes do not have feasible solutions under certain set of constraints.

3.4.1 Variables

Let us assume that the structure is of width W, depth D and height H. For the purpose of presenting the decision variables, it is helpful to assume that the structure grid is defined by the nodes $x_{i,j,k}$, i = 1,...,W+1, j = 1,...,D+1, k = 1,...,H+1.

$$\begin{array}{ll} w^m_{i,j,k} & = 1, \Leftrightarrow \text{ there is a bar from } x_{i,j,k} \text{ to } x_{i+1,j,k} \text{ with orientation } m \\ d^m_{i,j,k} & = 1, \Leftrightarrow \text{ there is a bar from } x_{i,j,k} \text{ to } x_{i,j+1,k} \text{ with orientation } m \\ h^m_{i,j,k} & = 1, \Leftrightarrow \text{ there is a bar from } x_{i,j,k} \text{ to } x_{i,j,k+1} \text{ with orientation } m, \end{array}$$

where m = 0 implies an orientation of 1 as described by the tensor description and m = 1 implies orientation of -1.

This notation may further be compacted with the following variable analogous to the tensor description:

$$z_{i,j,k,l,m} = \begin{cases} w^m_{i,j,k} & \text{if } l = 1 \\ d^m_{i,j,k} & \text{if } l = 2 \\ h^m_{i,j,k} & \text{if } l = 3, \end{cases}$$

but we use the preceding formulation for the sake of intelligibility.

3.4.2 Constraints

Overlapping bars constraints For each bar we need to forbid overlapping. The following constraints forbid overlapping bars:

$$\begin{array}{lll} w^0_{i,j,k} + w^1_{i,j,k} & \leq 1 & \forall j,k,i = 1,\ldots,W \\ d^0_{i,j,k} + d^1_{i,j,k} & \leq 1 & \forall i,k,j = 1,\ldots,D \\ h^0_{i,j,k} + h^1_{i,j,k} & \leq 1 & \forall i,j,k = 1,\ldots,H \end{array}$$

So there is one constraint for each possible bar in the structure.

Junction constraints The forbidden junction types may be barred by pairwise comparisons on bars attached to the same node. Because there exists a forbidden bar type in every direction for every bar in a junction and we don't want to include redundant constraints, we have a total of $2\binom{n}{2}$ constraints for

each node having n bars attached. The following constraints capture the forbidden junctions:

$$\begin{array}{rll} w^0_{i,j,k} + w^1_{i-1,j,k} & \leq 1 & \forall j,k,i = 2,\ldots,W \\ w^1_{i,j,k} + w^0_{i-1,j,k} & \leq 1 & \forall j,k,i = 2,\ldots,W \\ d^0_{i,j,k} + d^1_{i,j-1,k} & \leq 1 & \forall i,k,j = 2,\ldots,D \\ d^1_{i,j,k} + d^0_{i,j-1,k} & \leq 1 & \forall i,k,j = 2,\ldots,D \\ h^0_{i,j,k} + h^1_{i,j,k-1} & \leq 1 & \forall i,j,k = 2,\ldots,H \\ h^1_{i,j,k} + h^0_{i,j,k-1} & \leq 1 & \forall i,j,k = 2,\ldots,H \end{array}$$

$$\begin{array}{rcll} w^m_{i,j,k} + d^m_{i,j,k} & \leq & 1 & \forall m,k,i = 1,\ldots,W, j = 1,\ldots,D \\ w^m_{i,j,k} + d^m_{i,j-1,k} & \leq & 1 & \forall m,k,i = 1,\ldots,W, j = 2,\ldots,D+1 \\ w^m_{i,j,k} + h^m_{i,j,k} & \leq & 1 & \forall m,j,i = 1,\ldots,W, k = 1,\ldots,H \\ w^m_{i,j,k} + h^m_{i,j,k-1} & \leq & 1 & \forall m,j,i = 1,\ldots,W, k = 2,\ldots,H+1 \end{array}$$

$$\begin{array}{rcll} w^m_{i-1,j,k} + d^m_{i,j,k} & \leq & 1 & \forall m,k,i = 2,\ldots,W+1, j = 1,\ldots,D \\ w^m_{i-1,j,k} + d^m_{i,j-1,k} & \leq & 1 & \forall m,k,i = 2,\ldots,W+1, j = 2,\ldots,D+1 \\ w^m_{i-1,j,k} + h^m_{i,j,k} & \leq & 1 & \forall m,j,i = 2,\ldots,W+1, k = 1,\ldots,H \\ w^m_{i-1,j,k} + h^m_{i,j,k-1} & \leq & 1 & \forall m,j,i = 2,\ldots,W+1, k = 2,\ldots,H+1 \end{array}$$

$$\begin{array}{rcll} d^m_{i,j,k} + h^m_{i,j,k} & \leq & 1 & \forall m,i,j = 1,\ldots,D, k = 1,\ldots,H \\ d^m_{i,j,k} + h^m_{i,j,k-1} & \leq & 1 & \forall m,i,j = 1,\ldots,D, k = 2,\ldots,H+1 \\ d^m_{i,j-1,k} + h^m_{i,j,k} & \leq & 1 & \forall m,i,j = 2,\ldots,D+1, k = 1,\ldots,H \\ d^m_{i,j-1,k} + h^m_{i,j,k-1} & \leq & 1 & \forall m,i,j = 2,\ldots,D+1, k = 2,\ldots,H+1 \end{array}$$

Dead end constraints The structure must not contain a "dead end", ie. a node, whose degree is 1. Node may however be empty, ie. having a degree of 0, or have more than one bars attached to it, ie. having a degree of 2 or more. More formally for all nodes x, it must hold that $deg(x) = 0 \vee deg(x) \geq 2$.

In linear model we may express this constraint by introducing a new binary variable e for every node and by defining the following constraints. Let y be the number of bars attached to node x:

$$y_{i,j,k} = \sum_{m=0}^{1} \left(w^m_{i,j,k} + w^m_{i-1,j,k} + d^m_{i,j,k} + d^m_{i,j-1,k} + h^m_{i,j,k} + h^m_{i,j,k-1} \right).$$

Here the terms are only included in the expression if they are defined. That means that for nodes on the edge of the structure only those bars that may really exists are considered. Now the constraints are defined as

$$\begin{array}{rcl} 2 - y_{i,j,k} & \leq & M e_{i,j,k} \\ y_{i,j,k} & \leq & M(1 - e_{i,j,k}), \end{array}$$

where M is a large number. The variable $e_{i,j,k}$ denotes the emptiness of node $x_{i,j,k}$. This construct introduced 1 new variable and two constraints per each

node in the structure.

Because the junction constraints effectively limit the maximum number of arcs attached to a node to 4, the large number M need not be larger than 4.

If empty nodes are not allowed, the previous constraints may be relaxed and a simpler constraint placed instead:

$$-y_{i,j,k} \quad \leq -2$$

This doesn't introduce new variables and creates only onw new constraint per each node.

Projection constraints Each side of the structure must form a complete grid when viewed along all three axis. This may be described as the following constraints:

$$\begin{aligned}
-\sum_{m=1}^{2}\sum_{i=1}^{W+1} d_{i,j,k}^{m} &\leq -1 \quad \forall k,\ j = 1,\ldots,D \\
-\sum_{m=1}^{2}\sum_{i=1}^{W+1} h_{i,j,k}^{m} &\leq -1 \quad \forall j,\ k = 1,\ldots,H \\
-\sum_{m=1}^{2}\sum_{j=1}^{D+1} w_{i,j,k}^{m} &\leq -1 \quad \forall k,\ i = 1,\ldots,W \\
-\sum_{m=1}^{2}\sum_{j=1}^{D+1} h_{i,j,k}^{m} &\leq -1 \quad \forall i,\ k = 1,\ldots,H \\
-\sum_{m=1}^{2}\sum_{k=1}^{H+1} w_{i,j,k}^{m} &\leq -1 \quad \forall j,\ i = 1,\ldots,W \\
-\sum_{m=1}^{2}\sum_{k=1}^{H+1} d_{i,j,k}^{m} &\leq -1 \quad \forall i,\ j = 1,\ldots,D
\end{aligned}$$

This introduces a total of $(W+1)\cdot D + W\cdot(D+1) + (D+1)\cdot H + D\cdot(H+1) + (H+1)\cdot W + H\cdot(W+1)$ new constraints.

Corner constraints The corners of the structure must not be empty. This can be simply described with the following constraints:

$$-y_{i,j,k} \leq -2 \quad \forall\{(i,j,k) \quad | \quad x_{i,j,k} \text{ is a corner point } \}$$

Connectivity constraints For the structure to be connected and realizable, there must be a path between each nonempty node of the structure. For the purpose of describing the connectivity constraint in the form of linear constraints, we may instead consider the connectivity of connected subgraphs of the graph structure — the structure is connected, if every connected subgraph and its complement in the graph has a bar between them.

Let the set of nodes in the grid graph be V. Let us define the set of edges of the subset of nodes $U \subset V$ consisting of bars z that are attached both to the set U and its complement $V - U$ as $E(U) = \{z | z \text{ is attached to both } u \in U \text{ and } v \in V - U\}$.
Now the constraints may be defined as

$$\sum_{z \in E(U)} z \quad \geq 1 \quad \forall \{U \mid U \subset V, U \text{ is connected}, |U| \geq 4\}.$$

The limitation $|U| \geq 4$ is a trivial effort to remove some redundancies which are implied by the junction constraints. They implicitly forbid connected blocks of less than 4 nodes.

These constraints however also effectively forbid connected empty blocks of 4 or more nodes, since such blocks do not have a connection to the complementing graph. We do not know how overcome this limitation through linear constraints.

There are also unnecessary constraints due to infeasible forms of connected subgraphs that would not have to be tested. This could be overcome by stating that the subset V be such that it can be implemented despite the junction constraints. For example, it is trivially clear that the only feasible form of four connected nodes is a square, which implies that connected subgraphs in the shape of the letter 'L' would not have to be tested.

The connectivity constraints are drastically different from other constraints presented above in terms of complexity: the number of other constraints was polynomial regarding structure dimensions whereas the number of connectivity constraints is exponential. Additionally, describing these constraints for the purpose of numerical calculation is somewhat difficult. However, as there are efficient ways to enumerate all connected subgroups, the task is not impossible. Still, due to the aforementioned difficulties the connectivity constraints were never actually implemented. For a target size problem where the number of nodes is 160, the number of connected subgraphs becomes quite astronomical.

3.4.3 Objective function

In the optimization model we are maximize or minimize an objective function with which we try to measure the amount of aesthetic satisfaction of the structure in whole. This is subjective in the sense that individuals may have differing conceptions on the "beauty" of the structure. The subjective function is built by linearly combining measures of the structure's visual features that aspire to measure such properties of the structure that describe the appearance of the structure but are objective in the sense that they are interpreted approximately similarly by everyone. These features are weighed according to the subject's preference. These features and their combinations are discussed further in "Mathematical description of aesthetic lattice properties".

A trivial linear objective function whose coefficients are all ones measures the number of bars in the structure. If the dead end constraint is not relaxed, it also puts weight on the number of empty nodes, because we had to introduce a variable desribing whether a node was empty or not per each node. Minimizing

this type of objective function turned out to be quite efficient for numerical calculations and it was used for most of our optimization calculations.

3.5 Sequential linear programming algorithms

Because of the difficulties arising in describing the connectivity constraints, a sequential approach, where unfeasible solutions to the linear programming problem with relaxed connectivity constraints are constrained and the problem is reiterated, was also considered. This approach was not however explored further due to resource limitations.

4 Results

During the project we formulated several different methods of creating lattice structures. Because of the tight schedule, we managed to implement only two of the methods. When the coding of verification routines for the constraints was finished, creation by randomly generating the lattice structures was examined. This method created the lattice structures by drawing the orientations and locations for the bars at random, whether the lattice fulfilled the given constraints, was only checked afterwards. However, this algorithm was unable to find any solutions in decent time. A more intelligent generating routine was needed.

In the next phase, an algorithm based on graph theory was developed. This method proved to be much more effective as it was able to generate solutions within a reasonable time frame. The resulting lattices were found to be too similar, though. The structures lacked variation. Because the interesting solutions are extreme cases and they look very different from each other, we needed to find a new approach in order to fully investigate the limits of possible structures. To solve this problem a linear integer programming method was developed.

4.1 Realized outcomes of the project

As mentioned above, some of our initial ideas had to be disregarded in order to bring the project to a closure within the set timeline. However, these cutbacks in workload were mostly due to overoptimistic views of available resources and an aspiration to produce a functional tool for solving the problem at hand. However, even though these ideas were not developed further, they are still an integral part of the project and its solution. These unrealized ideas will be listed and examined in a later section.

We begin this overview of the realized outcomes by stating a somewhat obvious, but still very fundamental result; the fact that we are able to produce and display feasible lattice structures of preferred size. This alone stands as a proof-of-concept that it is possible to generate feasible lattice structures utilizing our current method based on graph theory. Moreover, our current results showcase ways in which we are able to create visualizations of produced lattice structures. These results by themselves complete two of the three main goals of the project.

To summarize the results concerning the visualization routine, we can conclude

that two different ways to illustrate the lattice structures were implemented. Both methods are able to correctly depict the structure, thus giving an accurate view of what the final lattice would look like if was constructed. The first method implemented is a somewhat simpler version, as it only utilizes Matlab's internal plotting routines to create a virtual model of the structure on hand (Figure 11). This technique requires little effort and makes it easy to use since both generation and visualization are made with Matlab. Nevertheless, as mentioned earlier we have also implemented a procedure that enables the user to export these lattice structures to a 3D animation program (Blender). Consequently, giving the user complete freedom to depict and examine the structures in any setting imaginable. Moreover, this program can easily be used to render striking images of completed structures for closer evaluation and comparison, thus assisting the user to single out the best possible lattice structure from a larger group.

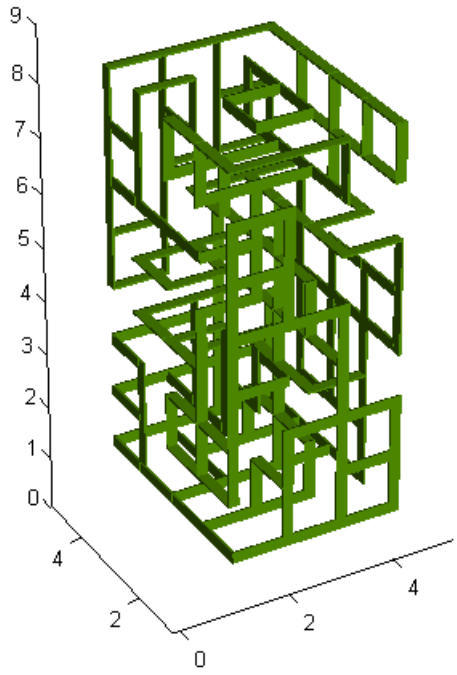

Figure 11: A feasible (3,4,7) structure

Visualizing the lattice structures has also given us a better understanding of how the current algorithm works and how it tends to create new structures. It has been discovered that, even though every structure (path) created is in fact random, the final outcomes still tend to look somewhat similar (e.g. many small loops, crowded lattice structures). The reason for this phenomenon can

be explained by the way our algorithm functions as a whole. Thus, further work is needed in order to move away from these similarities. A closer look at these challenges will be made in the next section. Nonetheless, visual evaluations have been able to yield new insights into structural features that might appear undesirable and awkward. Hence, enabling us to address these issues by designing algorithm updates that, either removes or minimizes the occurrence of such features.

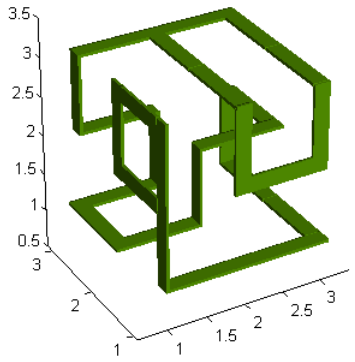

Figure 12: A connected (2, 2, 2) solution minimizing the object function

As designed, the algorithm can be launched from an existing structure, hence enabling the user to influence the final, feasible, lattice structure. However, this existing structure needs to be connected in order to guarantee connectivity of the final lattice.

Since the algorithm is unable to find extreme solutions, an entirely different approach was developed in order to address this issue. By converting the entire problem into a linear integer programming problem enabled us to define an object function to control the outcome. Then by either minimizing our maximizing this function we were able to yield feasible structures that represented extreme cases of what was possible within the given constraints. In this case the objective function was defined as the number of bars in the structure. A feasible lattice structure of size (2, 2, 2) that minimizes this objective function is shown in Figure 12. Solving this linear integer programming problem was found to be very computationally intensive. Thus, to speed up the process, the problem was calculated with Xpress-MP, as it turned out to be much more efficient than Matlab (using either Matlab's bintprog-routine or the GLPG-package [4]) in producing solutions.

Linear programming formulation gave further insights into the properties of the lattice structures. By inspecting the polytopes formed by the linear constraints, it was discovered that the smallest possible structure size which has a feasible

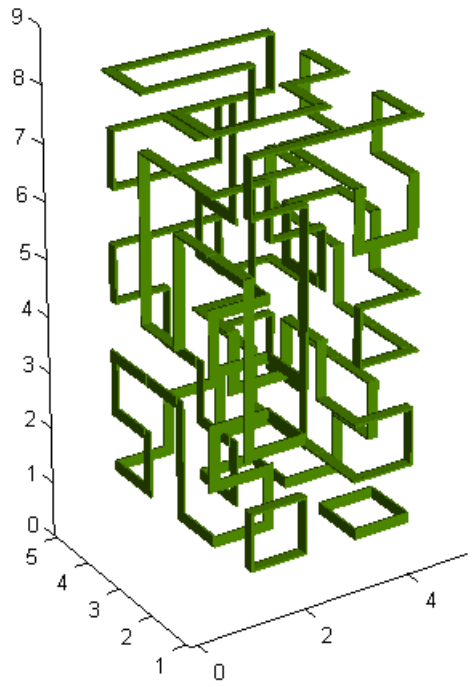

Figure 13: A non-connected (3, 4, 7) solution minimizing the object function

solution is 2x2x1. If corner constraints are relaxed, even a 1x1x1 structure has a feasible solution.

4.2 Challenges with the algorithm

Although the current algorithm is able to produce feasible solutions within a satisfactory time frame, there are still a number of issues that need attention in order to meet all our goals. Primarily, these issues are consequences of the way the code is constructed, or more specifically, the way it finds new paths. So, albeit the algorithm itself works, adjustments should be made in order to insure improved results, e.g. wider variety of solutions and more possibilities to influence the lattice structure's appearance.

As already established, the current algorithm is unable to produce a wide variety of solutions. This in return means that we are unable to investigate and produce feasible solutions that are interesting in a sense that they embody extreme features (e.g. minimum amount of bars) of what is possible within the give limitations.

Moreover, the algorithm is unable to guarantee that we will find a feasible solution once it has launched. The reason for this is twofold. First, the algorithm

does not check the projection requirements until it has completed the whole lattice (i.e. when no more bars can be placed inside it). Second, the algorithm is unable to guarantee that all the 8 corner points will be used, hence requiring that we also check these afterwards. Thus, if we want to find feasible solutions, we need to run the algorithm multiple times. On the other hand, this second issue could be circumvented by launching the algorithm from a structure that occupies all the corner points.

Concerning the linear integer programming problem, there is one issue that needs to be addressed. As of now, the method lacks a check for connectivity. Thus, when finding solutions for big structures, the probability for it to be non-connected is very large. An example of this is shown in Figure 13, the solution minimizes the objective function but is clearly not connected at the base.

4.3 Unrealized outcomes of the project

Projects are complicated constructs that require certain devotion from all the participating parties. To make things happen, things need to be scheduled so that the project goals are reached. We might have overestimated the resources in this case. A lot of ideas that were to be implemented were scrapped due to the lack of resources. The following paragraphs try to elaborate on certain ideas we had in mind that we did not implement.

First, the problem of generating suitable lattices was seen as an integer optimization problem. General linear integer model consists of target function (to be minimized/maximized) and linear constraints. The linear integer model was examined, but not implemented in the algorithm. The resulting lattice structures could be variated by using different object functions.

In an optimization approach, the value of every variable representing a bar is restricted to be either 0 or 1. Then those variables are chosen to minimize the cost function, subject to set of linear restrictions (see project outline). The requirement for connectivity is not linearly formulable, which requires one to check the connectivity of the solution afterwards. Theoretically, it is possible to optimize properties of the lattice. In practice, the general linear integer model formulation requires some 4500 different constraints for a lattice structure of size 3x4x7. The problem is a really complex one, and even if a feasible solution is found, it's still doubtful that the connectivity requirement will be realized.

Objective function consists of bar variables. Their sum of them can be minimized or maximized, which would make the lattice structure lighter or heavier. If we maximize the sum of bars next to each other, we make more turns and vice versa. We managed to create an optimization program, which is able to take linear and quadratic object functions as an input. The problems appear when the size of the lattice grows and the objective function is not linear or quadratic. The current algorithm works independently without any input, except the dimensions. It generates lattices that fulfill the requirements despite the fact that there is no cost function affecting the result. When a cost function is applied, the algorithm could generate lattices with certain properties. For example, the properties could be the number of bars or junctions.

Objective function could be given as an input in some form to the random path algorithm. For example, the number of junctions could be maximized either by steering the generating algorithm by supporting the junctions or by a brute-force back-tracking technique. With back-tracking the algorithm could generate loops, and when the algorithm is stuck in a dead-end it would take a couple of steps backwards. The back-tracking would demand a data base, which would contain all the possible paths or taken steps in case steps are wanted to take backwards.

There is also another way to generate lattices that was thought about. Every small-sized layer could be generated one by one, and the different layers would be connected to each other to form a connected structure. This would be a totally different execution, because the software should be in collaboration with user. Though, the existing software could be used, because the constraints should be checked every time when a layer would be added.

5 Discussion

5.1 Mathematical description of aesthetic lattice properties

In this section it is time to recognize some repetitious characteristic of the forms of the lattices and how they can be produced. These methods to produce lattices with some features introduced below are just examples. In our framework, the lattices' properties can be influenced in two ways: by changing the objective function on the one hand, and by revising the random lattice constructor on the other.

The optimization is separated in linear and non-linear parts. The constraints are in linear form in the optimization program presented above. The program is able to solve linear and quadratic objective functions. However, within this approach the constraints will remain linear and the objective function will range between linear and quadratic. To be exact, the sum of bar variables yields linear functions, whereas the product of the bar variables yields quadratic functions.

An intuitive example of the characteristics of lattices is the number of bars. This can be regarded as the total weight or the total cost of the lattice. In an optimization approach, this could be applied by setting the objective function as the sum of the bars.

One usual feature of the lattices produced by the random path algorithm, is plenty of 1 by 1 by 1 sized loops (shown in Figure 14). This often makes the lattice fore-seeable and divided to layers. Automatically, when loops start to accumulate somewhere, there is no other way to continue the existing structure because of the conditions presented above.

However, in the linear optimization approach, the number of loops can be easily controlled by fixing the objective function. In this approach, one has to de-

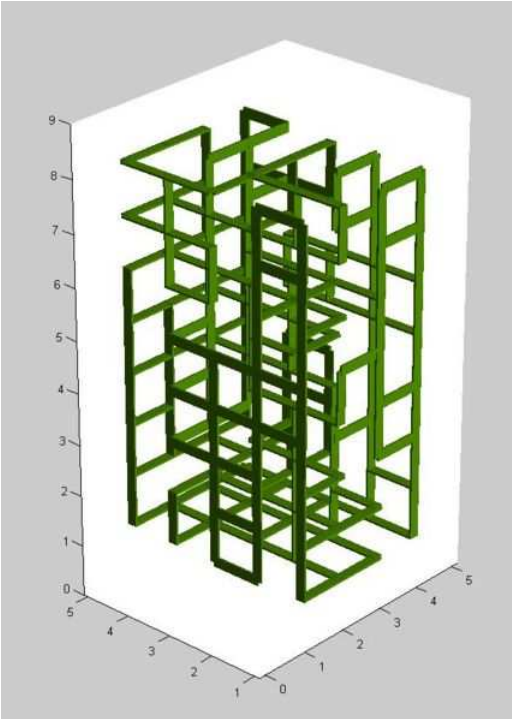

Figure 14: A lattice solution showing small loops

termine where either the long canes or the frequent loops could be, and then maximize the sum of those bars one by one. In this way, the construction will become more elongated, but within the limits of conditions.

Secondly, in the non-linear optimization approach there could be the products of the opposite bars in the objective function. See, if another one of them was missing, it would product null. In this way, the bar variables can easily get depending on each other.

In the random lattice constructor approach, the algorithm should be revised. If the algorithm would remember the previous turn, the next selection could be weighted not to select the same and vice versa.

Symmetrical lattices could be generated by the optimization method. In the objective function the possible symmetrical bar variables should be multiplied by each other. By maximizing the sum of products, symmetrical lattices could be produced.

Not only 1 by 1 by 1 sized loops, but also broadly cyclical structures can be favoured. In an optimization way, the critical bar variables of cycles must be multiplied by each other and then they all have to exist to turn a profit in the cost function. The algorithm could remember an exact number of previous turns

and then weight the next one according to the preceding steps. In this way, there could be cyclical behavior in the lattice.

5.2 Possible other approaches

5.2.1 Constraint programming

In constraint programming framework we describe the relations of the variables in terms of logical or agebraic constraints. After the formulation solutions to the constraints can be enumerated. This differs from the linear programming approach mainly in that we are not trying bto optimize any objective function. [8]

Utilizing constraint programming as a generative approach would have been interesting to explore further since the constraints can be expressed quite naturally by means of logic constraints. However, the connectivity problem is also very evident here and cannot be solved trivially.

5.2.2 Genetic algorithms

During the course of the project, there were talks about different approaches of how to integrate the interactivity brought in by user input into the software. Also different methods of creating various lattice structures were needed. Had we had more time, we probably would have implemented a GA based algorithm also. It would have been interesting to compare the results of different approaches.

Genetic Algorithms (GAs) are adaptive heuristic search algorithm based on the evolutionary ideas of natural selection and genetics. As such they represent an intelligent exploitation of a random search used to solve optimization problems. Although randomised, GAs are by no means random, instead they exploit historical information to direct the search into the region of better performance within the search space. The basic techniques of the GAs are designed to simulate processes in natural systems necessary for evolution, specially those follow the principles first laid down by Charles Darwin of "survival of the fittest.". Since in nature, competition among individuals for scanty resources results in the fittest individuals dominating over the weaker ones.

GAs simulate the survival of the fittest among individuals over consecutive generation for solving a problem. Each generation consists of a population of character strings that are analogous to the chromosome that we see in our DNA. Each individual represents a point in a search space and a possible solution. The individuals in the population are then made to go through a process of evolution.

GAs are based on an analogy with the genetic structure and behaviour of chromosomes within a population of individuals using the following foundations:

- Individuals in a population compete for resources and mates.
- Those individuals most successful in each 'competition' will produce more offspring than those individuals that perform poorly.

- Genes from 'good' individuals propagate throughout the population so that two good parents will sometimes produce offspring that are better than either parent.
- Thus each successive generation will become more suited to their environment.

A population of individuals is maintained within search space for a GA, each representing a possible solution to a given problem. Each individual is coded as a finite length vector of components, or variables, in terms of some alphabet, usually the binary alphabet 0,1. To continue the genetic analogy these individuals are likened to chromosomes and the variables are analogous to genes. Thus a chromosome (solution) is composed of several genes (variables). A fitness score is assigned to each solution representing the abilities of an individual to 'compete'. The individual with the optimal (or generally near optimal) fitness score is sought. The GA aims to use selective 'breeding' of the solutions to produce 'offspring' better than the parents by combining information from the chromosomes.

The GA maintains a population of n chromosomes (solutions) with associated fitness values. Parents are selected to mate, on the basis of their fitness, producing offspring via a reproductive plan. Consequently highly fit solutions are given more opportunities to reproduce, so that offspring inherit characteristics from each parent. As parents mate and produce offspring, room must be made for the new arrivals since the population is kept at a static size. Individuals in the population die and are replaced by the new solutions, eventually creating a new generation once all mating opportunities in the old population have been exhausted. In this way it is hoped that over successive generations better solutions will thrive while the least fit solutions die out.

New generations of solutions are produced containing, on average, more good genes than a typical solution in a previous generation. Each successive generation will contain more good 'partial solutions' than previous generations. Eventually, once the population has converged and is not producing offspring noticeably different from those in previous generations, the algorithm itself is said to have converged to a set of solutions to the problem at hand.

How would a GA be used to generate the lattices?

1. randomly initialize a population of lattices
2. determine fitness of population using a suitable fitness function
3. repeat
   (a) select parents from population
   (b) perform crossover on parents creating population
   (c) perform mutation of population
   (d) determine fitness of population
4. until best individual is good enough

In order to get the algorithm working, we would need a fitness function to evaluate the fitness of a lattice. If the iteration is supervised by the user, that is, the user selects the parents from the population, a fitness function is not required. Furthermore, the crossover method is needed. In case of a lattice structure required to fulfill certain qualifications, designing such a method is not a trivial task at all.

6 Concluding thoughts

The project has been a long process of generating something of real value, something that does not exist prior to the beginning of the project. During the course of the process, managing the productivity and schedule becomes of extreme importance. Successfully project managing is a skill that is learned by doing. With it comes also a great responsibility.

Let us begin with the downsides. The partitioning and scheduling of the project was not done properly. We could have achieved a whole lot more with greater efficiency had the project been partitioned to smaller and more meaningful pieces. This way a more precise distribution of work between group members would have been possible. It would also have balanced out the seemingly mismatched tasks. In the end we had one member doing the coding and the rest of the group doing something that was not really exactly specified.

Although the means of communication were exeptional compared to other project groups (ie. wiki), communication was somewhat tangled. People had their own schedules and a couple of project group meetings were held with only some two members present, the other being the project manager every single time. It seemed that people were somehow expecting them to be given specific tasks, despite the fact that a list of tasks what a member could do with his spare time existed. This policy always leads to a situation where the tasks are taken care of just moments before the deadline. And thus the role of project manager becomes even more emphasized.

There were times when nothing particular was done to advance the project. Perhaps this was due to the mismatched tasks. Some might have thought that there is nothing else to be done, as the coding was mostly done by one person. Constructing complex data structures and programming requires some experience and knowledge, and it is easier to let one person with a clear picture in his mind to handle the situation, than to interfere the process by dividing it to tasks that require enormous effort from less experienced members.

The estimated risks realized to a certain degree. The schedule was lagging behind, but delays were not severe in reality. We were unable to extend the functionality of the program by adding features to it that would take user input to consideration when generating lattices. The currect version of the algorithm is only usable within MATLAB computing package. Requiring one to buy very expensive software in order to make other applications work is not that elegant of a solution.

But when the upsides are taken into account, the project was a success. We managed to create what we were supposed to, a tool for sculptor to aid her designing and evaluating lattice structures. We managed to do some research on the future prospects of studying the lattice structures, and we managed to get some insight and inspiration for future development. The research continues, and the results and ideas can be applied in the future when developing the algorithm further. It was also delightful to experience sudden break-throughs during the final moments of the project. The formulation of the integer optimization problem and its results gave a whole lot of new information for further analyzing.

One thing exceeds all the other matters. It must be clear to every one participating in the project that the results must satisfy the needs of the client. This was the case in our project.

7 References

1. Doris Schattschneider, Mathematics and Art - So Many Connections, Math Awareness Month - April 2003. Mathematics and Art - So Many Connections.

2. Sara Robinson, Can Mathematical Tools Illuminate Artistic Style? Society for Industrial and Applied Mathematics, March 2005. Can Mathematical Tools Illuminate Artistic Style?

3. Joseph Malkevitch, Mathematical tools for artists. AMS, Feature Column from the AMS, April 2003. Mathematical tools for artists

4. GLPK (GNU Linear Programming Kit), http://www.gnu.org/software/glpk/

5. R. Diestel: Graph Theory (3rd ed.), Springer

6. http://library.thinkquest.org/trio/TTQ05063/monalisa.gif, picture retrieved May 9th, 2008

7. http://www.visitingdc.com/images/st-louis-arch-address.jpg, picture retrieved May 9th, 2008

8. http://en.wikipedia.org/wiki/Constraint_programming