

DELFT UNIVERSITY OF TECHNOLOGY
FACULTY OF ARCHITECTURE AND THE BUILT ENVIRONMENT

A Topology Optimization Process for Discrete Modular Design based on Discrete Element Modelling for generating reconfigurable funicular structures

by
Qinglu Chen
(5287413)

MSc Architecture, Urbanism and Building sciences
Building Technology

Thesis Supervisor

dr. Pirouz Nourian | Architectural Engineering + Technology, Design Informatics
dr. Simona Bianchi | Architectural Engineering + Technology, Structural Design & Mechanics

Advisor

dr. Anjali Mehrotra | Department of Materials, Mechanics, Management and Design (3MD)

June 30, 2022

Abstract

The paper presents a generative design process based on topology optimization methodology for configuring masonry structures. In this approach, structures consisting of stackable interlocking blocks are modelled as discrete elements using the Discrete Element Method, approximating their mechanical behaviours. A process is devised to result in funicular structures that can be built using a limited set of modular masonry blocks with the aim to lower the environmental costs in terms of embodied carbon, monetary costs, and construction labour. Additionally, this process aims to increase the reuse and reconfigurability potential of the stackable blocks by seeking utmost modularity in the topological design of the underlying 3D tiling/tessellation.

Topology optimization is widely known as a methodology for generating geometrically elaborate structures, which typically minimize the use of the material. These approaches typically use the Finite Element Method to formulate and solve the governing differential equations for computing their objective functions, assuming that the structure to be designed is a virtually continuous distribution of material that is refinable within a continuum. However, at a more general level, the idea of topology optimization can also be applied to inherently discrete problems by creating algorithms based on the Discrete Element Modelling approach (O'Sullivan, [2011](#)), or a particle system at a quasi molecular level. The proposed approach is applicable in the design of funicular structures, with the potential for form-finding of waste-free and reconfigurable, structural geometries that are constructible using a limited stock of modular blocks.

The paper introduces a discrete topology optimization process in three steps: 1) defining a space-filling 3D tiling/tessellation consisting of interlocking blocks as a graph colouring of a voxel grid; 2) defining the objective function of a topology optimization problem based on a Discrete Element Modeling approach; 3) assembling a topology optimization algorithm using the gradient-based Optimality Criteria method adapted to work with the DEM-based governing equations, deriving an objective function and related gradient equations(O'Shaughnessy et al., [2021](#)).

The proposed methodology allows designers to find static equilibrium configurations for funicular structures defined by their desired space by minimizing potential energy between blocks. The method is validated for simply designing space discretized as interacting particles, whose optimum solutions compare to those from a typical continuum-based algorithm(Bendsøe & Sigmund, [2004](#)).

Acknowledgments

Topology optimization is widely known as a methodology for generating geometrically elaborate structures, which typically minimize the use of the material. These approaches typically assume that the structure to be designed is virtually a continuum. To apply TO in inherently discrete problems, such as compression-only structures, is state of the art. This thesis mainly focuses on developing the mathematical theory and shows the possibility of bridging Topology optimization and discrete problems.

As an architect, I witness how we design and construct, leading to further material extraction, extensive energy consumption, waste production and environmental degradation, and realize the building industry is transitioning to reduce destruction. I believe that the strategy of optimizing structure is one of those.

I want to thank my first mentor, Pirouz Nourian, for guiding my research and ongoing assistance in exploring and developing the mathematical process. It is my honour to be one of his students, and I am looking forward to more collaboration in the future.

I want to thank my second mentor, Simona Bianchi, for guiding the structural part of this research and explaining the knowledge of masonry structure. I want to thank my advisor, Anjali Mehrotra, for taking the time to explain the field of Discrete Element Modelling. Hopefully could have more opportunities to cooperate henceforth.

I would also like to thank a few people who helped with the important parts of developing this method. I want to thank Nan Bai for inspiring me on how to deal with the last step of the mathematical process. Shervin Azadi kindly gave me an idea of how to implement the tessellation pattern. I am extremely indebted to Baolian Liu's help with the output visualization. I am grateful to receive endless kindness during this period.

This research is surely a starting point, and I would like to continue with this topic if possible. Thank you all!

CONTENTS

Abstract	2
Acknowledgments	3
1. Research framework	6
1.1 Introduction	6
1.2 Background and necessity	6
1.3 Problem Statement	7
1.4 Scope and limits	8
1.5 Research objective	8
1.6 Research questions	9
1.7 Methodology	10
1.8 Planning and organization	11
2. Literature study	13
2.1 Introduction	13
2.2 Discrete element modelling	13
2.2.1 Discretizing space	13
2.2.2 Discrete Element Method	13
2.2.3 Static equilibrium as constraints	16
2.3 Topology optimization	18
2.3.1 What is structural optimization	18
2.3.2 Topology optimization methods	18
2.3.3 Topology optimization methods using discrete elements	19
2.3.4 SIMP-based Topology optimization	20
2.3.5 Discrete Element Topology optimization method	22
3. Design	24
3.1 Design methodology	24
3.2 Space-filling system	24
3.2.1 Space-filling system design	25
3.2.2 Structural analysis	26
3.3 DEM-based Topology Optimization	30
3.3.0 Notation	31
3.3.1 Colouring voxels and constructing network	32
3.3.2 Discrete Element Analysis	35
3.3.3 Discrete Element Topology Optimization	38
3.3.4 Algorithm design	42
3.3.5 Toy problems	47
4. Conclusion	55
References	60

Appendix	62
Appendix A: Topology optimization methods using discrete elements	62
Appendix B: Wind load calculation and related Eurocode	67
Appendix C: Abaqus results of two modular blocks	69

3. Design

3.1 Design methodology

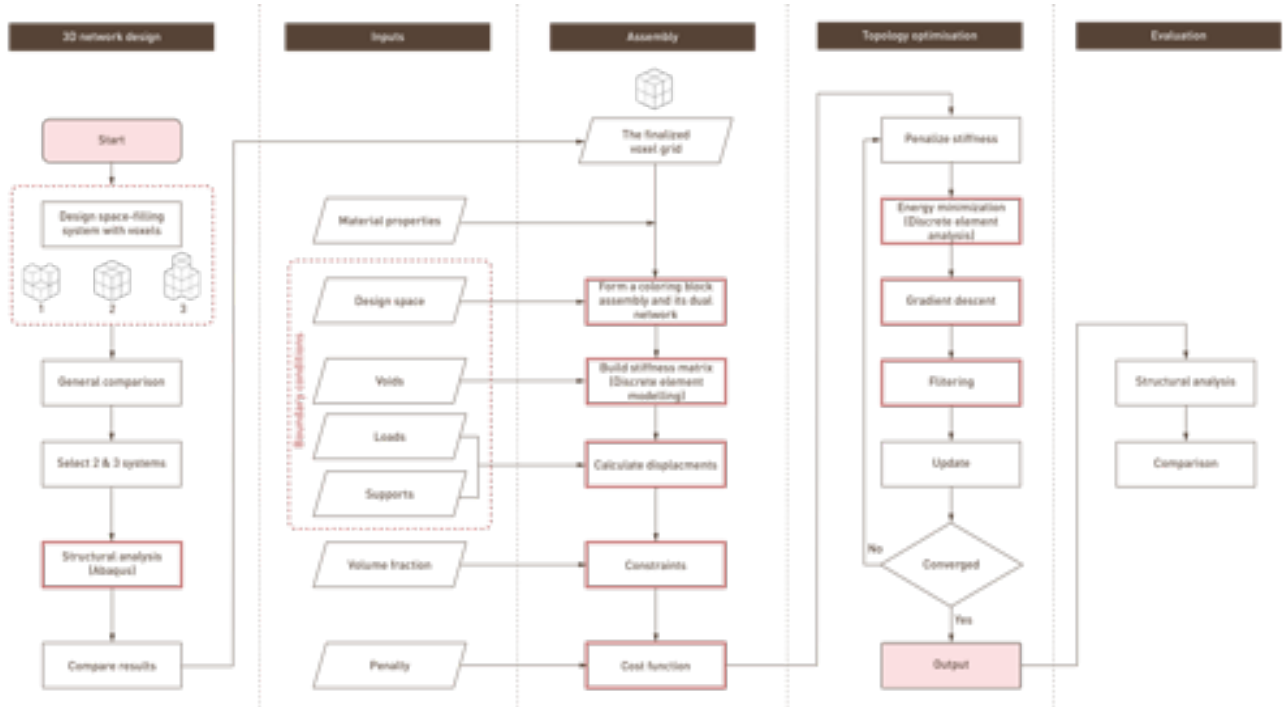


Figure 14

Design methodology

The following section of the report is aimed at presenting the proposed design methodology and preliminary results. The design framework consists of three main parts:

- The exploration of the space-filling system;
- The implementation of topology optimization;
- The topology design for the block unit (Liu.).

3.2 Space-filling system

The first part cooperates with Liu. The start point of this part is to develop a topological interlocking system that can fill the space.

The topological interlocking system

The topological interlocking system is a term meaning the elements within this system depend on its neighbour element. In this way, high bending forces are resisted, and even no additional binding material like mortar is needed. This can be achieved by designing a unique geometry and specific arrangement of the blocks.

Two fundamental principles of the interlocking system are introduced before stepping into the design process. The osteomorphic topological assembly is one of the approaches. Another method is a layer-like hexagon-based or square-based structure. The first interlocking assembly of this kind was a plate-like square-based structure of tetrahedra (Dyskin et al., 2005). However, current generative logic can only use in designing 2.5D structures, like shell structures, which means it cannot fill the three-dimensional space.

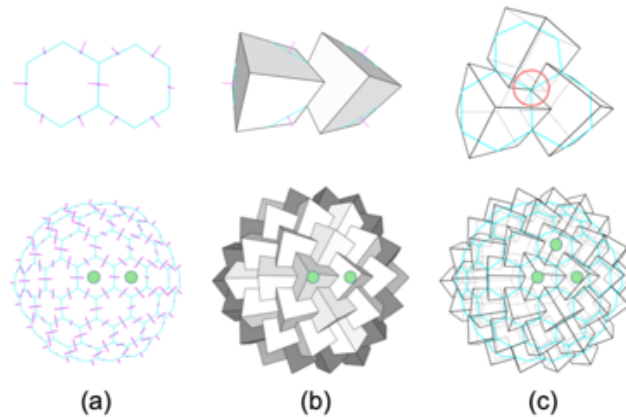


Figure 15

A design process for a layer-like hexagon-based structure (Wang et al., 2019)

3.2.1 Space-filling system design

The topological interlocking system provides an idea to design the space-filling system. Several three-dimensional tessellations are developed with the help of voxels.



Figure 16

3 types of block unit

The middle one is sound promising for compression-only structures without considering interlocking.

The generating logic behind the left one is the plane-like hexagon-based structure, where the third dimension square grid is added to achieve space-filling. The right one is developed by combining the previous two selections.

A general comparison between these three systems is made to understand their advantages and disadvantages better. First of all, systems 1 and 3 are preset interlock; consequently, they have an advantage under horizontal loads. According to the load path of these three systems, system 3 is structurally efficient and perform well in compression-only structures. Moreover, the load path of system 2 follows the edges of the trigonal trapezohedron, which is a space-filling geometry. Also, system 3 can refer to the rhombic dodecahedron.

After comparing the above grid system, 2 and 3 are chosen for the next step, structural analysis with Abaqus.

3.2.2 Structural analysis

To understand the mechanical behaviours of blocks and to finalize the general space-filling system, structural analysis is necessary. Besides the modular blocks, material properties, external loads, and the analysis limits need to be set.

Limits

Several limits of this research are preset.

- The structural loads and safety factors will be based on Eurocode standards.
 - Mortar or other connections between two blocks will not be taken into account for formulating and structural analysis.
 - The structural analysis will be a linear static analysis so that the stiffness matrix will be constant, and the solving process is relatively short compared to a nonlinear analysis on the same model.
 - The material of masonry blocks is considered as idealized isotropic material, and the block itself is assumed as a rigid body.

Material properties

The material properties are taken from the experimental result of masonry wall behaviour (Abdulla et al., 2017), where the friction behaviour between bricks is also considered. Details can be found in the below table.

Applied loads

In this simulation, applied loads include horizontal load (wind load) and vertical loads (snow load and live load). Regarding the snow and wind loads related to climate data, a location is assumed to get these data, where is Paris. The standard of classification and calculation function is based on the Eurocode. Details of wind load calculation can refer to Appendix B.



Figure 17

comparison of 3 types of system

Material Property				
	Density [kg/m ³]	1800		
	Young's Modulus [kN/m ²]	1,55E+07		
	Poisson Ratio [-]	0,2		
Drucker Prager	Angle of Friction [°]	31.79		
	Flowstress Ratio [-]	0,8		
	Dilation Angle [°]	2,85		
	Drucker Prager Hardening			
	Hardening behavior type	Compression		
	Data	Yield Stress [MPa]	Abs Plastic Strain [-]	
		7,26	0	
		7,03	0,00045	
		6,58	0,0029	
		5,9	0,0044	
4,83	0,005			
Contact Property	Tangential Behavior	Friction formulation	Penalty	
		Friction Directionality	Isotropic	
		Friction Coefficient	Slip Rate	Contact Pressure [kPa]
		1,504	0	0,5
		0,91	0	1
	6116	0	2	
	Normal Behavior	Pressure Overclosure	"Hard" Contact	
Abdulla, K. F., Cunningham, L. S., & Gillie, M. (2017). Simulating masonry wall behaviour using a simplified micro-model approach. <i>Engineering Structures</i> , 151, 349–365. https://doi.org/10.1016/j.engstruct.2017.08.021				

Figure 18

Material properties of block unit

Block & applied loads		
Voxel unit dimension Height×Weight×Length [m×m×m]	0.08×0.08×0.08	
Number of voxels for single brick [-]	8	
Self-weight per brick [kg]	$m=\rho V$	7,3728
Snow load [kN/m ²]	Characteristic value of snow load (s_k)	0.45 EN 1991-1-3 (zone A1)
	safety factor	1,5
Live load [kN/m ²]		3
	safety factor	1,5
Wind load [kN/m ²]	Basic velocity pressure (q_b)	0.35 EN 1991-1-4 (zone 2)
	safety factor	1,5

Figure 19

Applied loads of block unit

Structural analysis and discussion

A structural analysis flowchart is made to show the procedure of working with Abaqus. In between the procedure, step setting is also worth discussing. This simulation considers three steps, including the initial, static and dynamic states. Supports are assigned from the beginning, vertical loads apply in static and dynamic states, and the horizontal load is added in the dynamic step. These can help to understand better what happens when horizontal loads are taken into account.

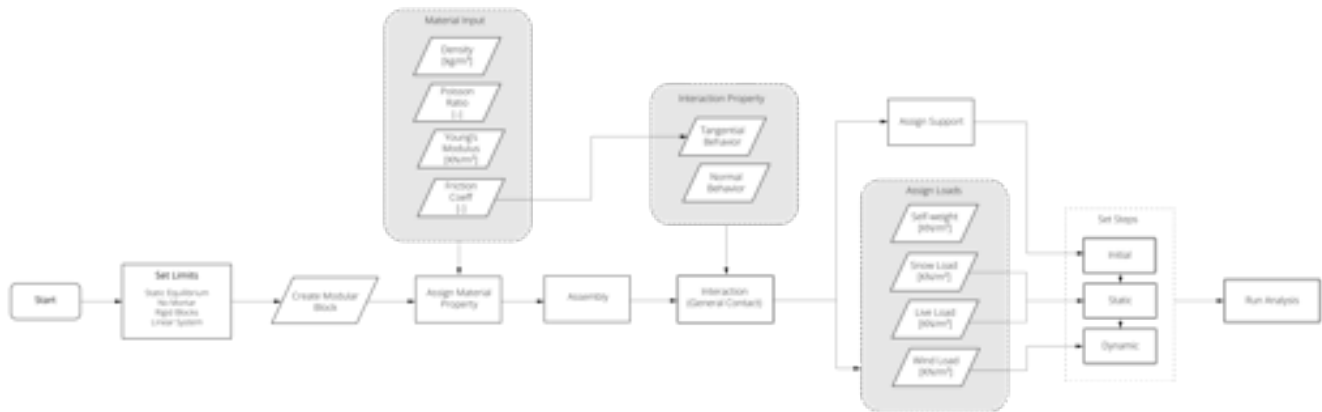


Figure 20

Structure analysis flowchart

The results of the principal stress and the displacement in x and z direction are analyzed (Appendix C). The displacement in the vertical(z) direction of B is more significant than A when comparing their value change from vertical load only(max) to vertical and horizontal load(max). However, B performs better under horizontal load. As for maximum principal stress, although B's compression is less than A, the tension change within A is far less than B. Also, considering the material used, B needs more material to achieve the same height as A. Therefore, A is more efficient than B in compression-only structures. However, B can be a good choice when applying to a location where the horizontal load is high.

To sum up, A is selected because it performs more efficiently in compression-only structures and uses less material.



			A	B
				
S [N/mm ²] (max principal stress)	Vertical Load Only	Compression	-1.1070E-03	-9.700E-03
		Tension	8.4120E-04	2.371E-02
	Vertical+Horizontal	Compression	8.5440E-05	-6.6080E-03
		Tension	2.3810E-02	1.6390E-02
U [mm] (displacement)	Vertical Load Only	Max	4.2290E-03	3.2280E-02
		Min	0	0
	Vertical+Horizontal	Max	9.1180E-02	1.8170E-01
		Min	0	0
Uz [mm] (displacement in vertical direction)	Vertical Load Only	Max	-4.2290E-03	5.7660E-03
		Min	0	-2.4020E-02
	Vertical+Horizontal	Max	1.1080E-02	1.2390E-02
		Min	-1.4640E-02	-2.5370E-02
Ux [mm] (displacement in wind direction)	Vertical Load Only	Max	4.1520E-04	2.4590E-02
		Min	-4.1520E-04	-2.4550E-02
	Vertical+Horizontal	Max	9.07E-02	1.1830E-02
		Min	0.00E+00	-1.5510E-01

Figure 21

Structure analysis results(Abaqus)

3.3 DEM-based Topology Optimization

After finalizing the graph network, the next step is to do the numerical implementation. The first step is to colouring the voxels and constructs the graph. By doing this, we can get the target graph network based on the general voxel grid. Another thing that has to define before starting the topology optimization loop is the boundary conditions, including design space, voids, loads and supports. Together with the structural network, the state of equilibrium can be found by solving a group of linear formulas. The reason for doing this is mainly to get the displacement of the blocks. Lastly, the compliance, constraints, penalization scheme and filter are formulated to complete the loop of DEM-based topology optimization.

The described process mainly includes two parts: the mathematical design and the algorithm design. The mathematical design is related to the theory behind topology optimization and discrete element modelling technique. Regarding the algorithm design, there is no existing open source built on this methodology; it is, therefore, vital to make sure all the matrix operations in the mathematical part would result in desired outputs.

3.3.0 Notation

Symbols

α	numerical damping coefficient
e	edges
E	Young's modulus
f	force increment
$frac$	the target ratio for volume of solid, $frac \in (0, 1)$
g	gravitational acceleration
k	contact stiffness between two blocks
λ	Lagrange multiplier
$\mathbf{M}_{e,v}$	incidence matrix with e rows and v columns
m	the number of edges
μ	the coefficient of static friction
n	the number of nodes
N	total number of blocks
p	penalty for the penalization scheme
P	centroid of the blocks
r	distance increment
r_e	inter-particle(block) distance
\bar{r}_e	equilibrium distance of two particles(blocks)
U	potential energy
v	vertices
V	sum of the solid volume within the design domain
V_0	the whole domain is solid
ν	poisson ratio
χ_e	design variables

Subscript

e	the e -th edge
f	fixed nodes
i, j	the i -th and j -th node
n	normal direction
t	tangential direction
x, y, z	the x, y, z directions

Superscript

s	strong connection between two blocks
w	weak connection between two blocks

Vector Operations

$a + b$	move a block a to specific location b in the coordinate plane
---------	---

Matrix Operations

\mathbf{a}	a $m \times 1$ matrix
\mathbf{A}	a $m \times m$ diagonal matrix belongs to \mathbf{a}
\mathbf{A}^T	Transpose matrix
\mathbf{A}^{-1}	Inverse matrix
$\mathbf{A} \odot \mathbf{B}$	Hadamard product

3.3.1 Colouring voxels and constructing network

Based on the space-filling system defined in the previous section, the next step is to implement it as a colouring voxel (or pixel) and generate its dual network for structural simulation and topology optimization purposes.

The algorithm started with defining the based voxel (or pixel) unit, the design space domain and the aggregation pattern, which was achieved by computing the translation vectors. The algorithm in this part is expected to generate a graph with nodes and edges as the centroids and the links of blocks.

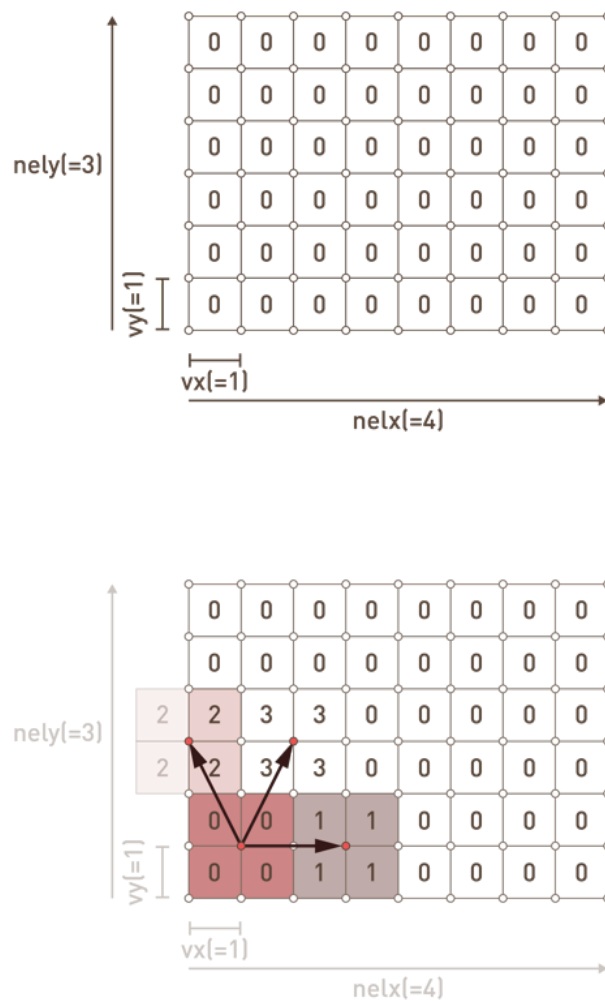


Figure 22

proposed network system

First of all, the design space domain is filled with zeros. Based on the setting, a block consists of four

voxels (or pixels) and the translation vectors, which in the 2D case are $(2, 0)$, $(-1, 2)$ and $(1, 2)$. By operating vector addition, blocks can be translated to given positions.

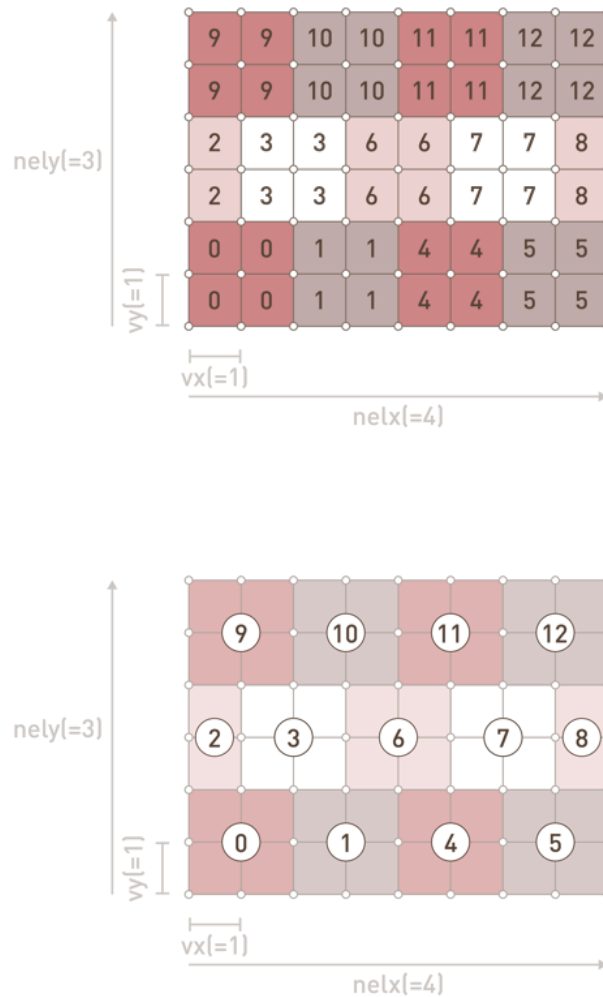


Figure 23

proposed network system

Considering this pattern as a group, the batch translation vectors can be easily obtained by dividing the design space on each axis. Then, the design domain is coloured by the setting pattern with the element id and batch id assigned.

The centroids of blocks can get from calculating the means of the grouped batch index, which represents each voxel's (or pixel's) centroid.

Based on the configuration of blocks, an adjacency matrix can be built. This can be achieved by identifying the specific difference between blocks' centroids.

Weighted adjacency matrix

In a traditional masonry system, the units are laid in mortar, and the areas between units are called mortar joints. The areas that form the vertical connections between each block (unit) are called head joints, while the bed joints create the horizontal connections. According to experiment results, the horizontal flexural strength can be assumed as approximately three times higher than vertical flexural strength when having clay as bricks and mortar as connections (Rita Esposito, 2016). The stiffness of the head and bed joints depends on the mortar properties, and if no mortar is considered, it is influenced by the surface roughness of the blocks. In our structural mechanics approximation with no mortar being considered between block units, two types of connections are defined following the same approach: 1) strong connections representing the bed joints, and 2) weak connections identifying the head joints. The relation of stiffness for these two types of connections can be assumed as:

$$k^w = \alpha k^s \quad (1)$$

where k^w and k^s correspond to the stiffness of the weak and strong joints respectively, and α is the linear relationship between two connections. In this case, all connections are assigned a weight. After constructing the adjacency matrix in coo matrix form, the graph is generated with NetworkX, a Python library for graphs and networks. A graph $G(E, V)$ with vertices V and edges E information is obtained.

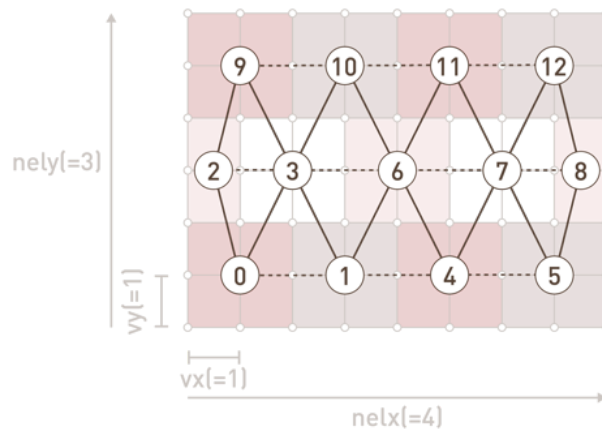


Figure 24

proposed network system

3.3.2 Discrete Element Analysis

Incidence matrix

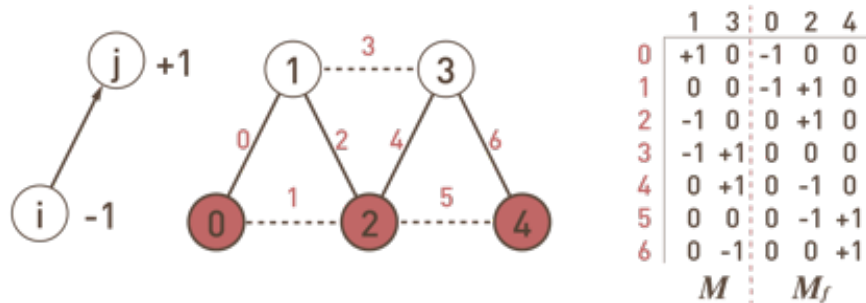


Figure 25

Incidence matrix example

Starting with the network graph $G(E, V)$ developed in the last section, an incidence matrix $M_{v,e}$, which encodes the relation of nodes and edges, can be generated for later application. The directed incidence matrix is defined by

$$M_{v,e} = \begin{cases} +1, & e[0] = v \\ -1, & e[1] = v \\ 0, & \text{in the other case} \end{cases} \quad (2)$$

where $v \in V, e \in E$. Therefore, the incidence matrix is a $n \times m$ matrix, and m equals the number of edges and n the number of nodes. Then we define the vertex with $M^{(+)} = +1$ is called source and $M^{(-)} = -1$ called target.

$$\begin{cases} M_s^{(+)} := (M_s > 0) \\ M_s^{(-)} := (M_s < 0) \end{cases} \quad (3)$$

In the force density method, the incidence matrix M , together with the diagonal matrix Q which contain the force-length ratios can operate the Gaussian transformation $M^T Q M$ (Schek, 1974). Considering the contact stiffness as a force density, this operation can obtain the stiffness matrix. Here we define M_s as a edge-node matrix. Therefore, it is a $m \times n$ matrix. Supported by the first layer of blocks, its corresponding nodes need to be fixed on the floor. Then the free and fixed nodes are distinguished. Based on the classification of free and fixed nodes, we define the matrices $M_s = [M \ M_f]$.

Contact stiffness

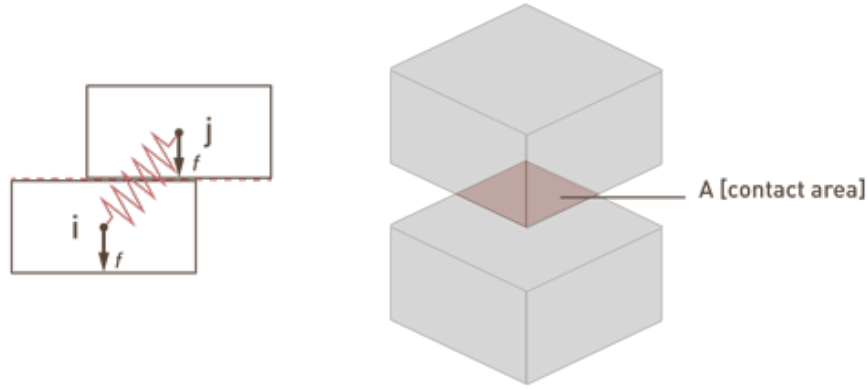


Figure 26

Interaction between two blocks

The first important aspect in the Discrete Element Method is to determine the contact state. The contacts happen when two particles interact with each other. Note that as rigid blocks are used in this procedure, all system deformability is assumed to be concentrated in the interfaces (contacts) between them. This deformability is accounted for via the contact springs, which also model the interaction (i.e. force transfer) between blocks. For the blocks made of the same material, we consider that the springs of two connected blocks are in series implicitly when calculating the spring stiffness in the edge term.

$$\frac{1}{k_e} = \frac{1}{k_i} + \frac{1}{k_j} \quad (4)$$

$$\begin{cases} k_i = \frac{E_i A_i}{l_i} \\ k_j = \frac{E_j A_j}{l_j} \end{cases} \quad (5)$$

From the pre-selected/pre-defined 3D tessellation pattern, it is known that the contact area between the blocks is the same, i.e. $A_i = A_j = A$, while the blocks are also made of the same material, and consequently have the same Young's modulus $E_i = E_j = E_e$. Therefore, the contact stiffness for a pair of blocks can be formulated as:

$$k_e = \frac{E_e A_e}{r_e} \quad (6)$$

where r_e is the distance between two blocks' centroids. Together with Eq. (1), the contact stiffness for strong and weak connection k_e^s, k_e^w can then be obtained.

Static equilibrium

Interpreting the nodes as the centroid $P_i \in \mathbb{R}^3$ of blocks, where $i = 1, \dots, n$. The fixed points with coordinates $(x_{fi}, y_{fi}, z_{fi}), i = 1, \dots, n_f$, therefore $\mathbf{x}, \mathbf{y}, \mathbf{z}$ is a n -vectors and $\mathbf{x}_f, \mathbf{y}_f, \mathbf{z}_f$ is a n_f -vectors. The length r_e form the m -vector \mathbf{r}_e . The edge vector $\mathbf{e}_{ij} \in \mathbb{R}^3$ with coordinates $\mathbf{u}, \mathbf{v}, \mathbf{w}$ can be written as

$$\begin{aligned} \mathbf{u} &= \mathbf{M}\mathbf{x} + \mathbf{M}_f \mathbf{x}_f \\ \mathbf{v} &= \mathbf{M}\mathbf{y} + \mathbf{M}_f \mathbf{y}_f \\ \mathbf{w} &= \mathbf{M}\mathbf{z} + \mathbf{M}_f \mathbf{z}_f \end{aligned} \quad (7)$$

$\mathbf{u}, \mathbf{v}, \mathbf{w}$ are therefore m -vectors.

The load vectors $\mathbf{f}_x, \mathbf{f}_y, \mathbf{f}_z$ are the forces that act on the nodes, including external load and self-weight, each load vector is therefore a $n \times 1$ matrix. With the diagonal matrix $\mathbf{U}, \mathbf{V}, \mathbf{W}$ that belongs to $\mathbf{u}, \mathbf{v}, \mathbf{w}$. The equilibrium state achieve when the net force for each blocks equal zero.

$$\begin{aligned} \mathbf{M}^T \mathbf{U} \mathbf{k}_e &= \mathbf{f}_x \\ \mathbf{M}^T \mathbf{V} \mathbf{k}_e &= \mathbf{f}_y \\ \mathbf{M}^T \mathbf{W} \mathbf{k}_e &= \mathbf{f}_z \end{aligned} \quad (8)$$

Referring to Eq.6, \mathbf{k}_e is a $m \times 1$ matrix form by scaler E and the m -vector \mathbf{r}_e .

$$k_e = \frac{E_e A_e}{r_e} \quad (9)$$

With the identities

$$\begin{aligned} \mathbf{U} \mathbf{k}_e &= \mathbf{K}_e \mathbf{u} \\ \mathbf{V} \mathbf{k}_e &= \mathbf{K}_e \mathbf{v} \\ \mathbf{W} \mathbf{k}_e &= \mathbf{K}_e \mathbf{w} \end{aligned} \quad (10)$$

Therefore, Eq.5 becomes

$$\begin{aligned} \mathbf{M}^T \mathbf{K}_e \mathbf{M} \mathbf{x} + \mathbf{M}^T \mathbf{K}_e \mathbf{M}_f \mathbf{x}_f &= \mathbf{f}_x \\ \mathbf{M}^T \mathbf{K}_e \mathbf{M} \mathbf{y} + \mathbf{M}^T \mathbf{K}_e \mathbf{M}_f \mathbf{y}_f &= \mathbf{f}_y \\ \mathbf{M}^T \mathbf{K}_e \mathbf{M} \mathbf{z} + \mathbf{M}^T \mathbf{K}_e \mathbf{M}_f \mathbf{z}_f &= \mathbf{f}_z \end{aligned} \quad (11)$$

Set $\mathbf{K} = \mathbf{M}^T \mathbf{K}_e \mathbf{M}$ and $\mathbf{K}_f = \mathbf{M}^T \mathbf{K}_e \mathbf{M}_f$, we have the equilibrium equation in the form

$$\begin{aligned} \mathbf{K}\mathbf{x} &= (\mathbf{f}_x - \mathbf{K}_f \mathbf{x}_f) \\ \mathbf{K}\mathbf{y} &= (\mathbf{f}_y - \mathbf{K}_f \mathbf{y}_f) \\ \mathbf{K}\mathbf{z} &= (\mathbf{f}_z - \mathbf{K}_f \mathbf{z}_f) \end{aligned} \quad (12)$$

The equation is now in a common form $Ax = b$. With the given external forces and the fixed points, the problem can be solved.

$$\begin{aligned} \mathbf{x} &= \mathbf{K}^{-1}(\mathbf{f}_x - \mathbf{K}_f \mathbf{x}_f) \\ \mathbf{y} &= \mathbf{K}^{-1}(\mathbf{f}_y - \mathbf{K}_f \mathbf{y}_f) \\ \mathbf{z} &= \mathbf{K}^{-1}(\mathbf{f}_z - \mathbf{K}_f \mathbf{z}_f) \end{aligned} \quad (13)$$

3.3.3 Discrete Element Topology Optimization

The Compliance

The displacement of the blocks under external and internal forces can be solved by minimizing the total potential energy of the system $U_{tot} = \sum U_{ij}$. Based on the formula of interaction potential energy, we rewrite the equation in edge terms. The elastic potential energy for each pair of blocks is

$$U_e = \frac{1}{2} k_e r^2 \quad (14)$$

where

$$r := r_e - \bar{r}_e \quad (15)$$

The displacement calculated in the previous step can generate the new centroids of the blocks, and then it can easily compute the distance r_e based on the incidence matrix and the new centroids P_i .

To apply SIMP topology optimization to described Discrete Element Method, the first thing is to define the design variable χ_e as per block quantity. To minimize the potential energy in the system, with the Eq.14, the minimization problem becomes:

$$\min_{\chi_e} : \quad c(\underline{\chi}_e) = \frac{1}{2} \sum_e^N k_e r^2 \quad (16)$$

Another critical change in the DEM framework proposed by O'Shaughnessy et al. (2021) is the

penalization scheme. In the DEM context, the stiffness k is associated with pairs of element instead of a single element. Therefore, the penalization scheme is modified as

$$k_e = \chi_{e,i}^p \chi_{e,j}^p \bar{k}_e \quad (17)$$

where \bar{k}_e is a constant stiffness, and $\chi_{e,i} \in \mathbf{M}^{(-)}$ and $\chi_{e,j} \in \mathbf{M}^{(+)}$ are the design variables of two interacting blocks. The penalty $p = 2$ provides a good convergence between the optimality of the solution, solid-void only result and numerical performance is indicated through numerical experiments done by O'Shaughnessy et al. (2021).

The penalization scheme in Eq.17 imposes that the interaction stiffness depends on $\chi_{e,i}$ and $\chi_{e,j}$. When $\chi_{e,i}$ or $\chi_{e,j}$ equals zero, the connection between two blocks would be damaged. Then Eq.26 can be rewritten as

$$\min_{\chi_e} : c(\chi_e) = \frac{1}{2} \sum_e^N \chi_{e,i}^p \chi_{e,j}^p \bar{k}_e r^2 \quad (18)$$

$$\text{subject to} : \begin{cases} \frac{V(\chi_e)}{V_0} = \text{frac} \\ 0 < \chi_{\min} \leq \chi_e \leq 1 \end{cases} \quad (19)$$

The *frac* in the first constraint is a ratio that fixed the target volume of solid blocks, where $\text{frac} \in (0, 1)$. V_0 represents the whole design domain is solid with full $\chi_e = 1$, and $V(\chi_e)$ equals the sum of the target volume. The second constraint sets a bound to χ_e from a minimum value to 1. The minimum value χ_{\min} could be 0 in principle, but a non-zero value, like 10^{-3} , is needed when applying filtering.

With the $m * n$ matrix $\mathbf{M}^{(-)}$ and $\mathbf{M}^{(+)}$ belonging to $\chi_{e,i}$ and $\chi_{e,j}$, the sum of potential energy can be easily represented with matrices. The objective function then simply becomes

$$c(\boldsymbol{\chi}) = \mathbf{r}^T (\mathbf{M}^{(-)} \boldsymbol{\chi})^p \odot (\mathbf{M}^{(+)} \boldsymbol{\chi})^p \odot \mathbf{k} \odot \mathbf{r} \quad (20)$$

where \mathbf{k} is not the global stiffness matrix but a $m \times 1$ constant stiffness matrix in the edge term, and \mathbf{r} is a $m \times 1$ matrix.

Optimality Criteria

The optimization problem in Eq. 22 can be solved with various approaches, and the Optimality Criteria method is the most popular one among those. This method provides a update scheme to update the design variable χ_e in each iteration. To minimize the compliance, an analytical expression of the sensitivity $\underline{\sigma}$ is formulated by combining the objective function with penalization scheme.

$$\boldsymbol{\sigma} = \left[\frac{\partial c}{\partial \chi_i} \right]_{n \times 1} = -p \boldsymbol{\chi}^{p-1} \odot ((\mathbf{M}^{(+)})^T (\mathbf{k} \odot \mathbf{r}^2 \odot (\mathbf{M}^{(+)} \boldsymbol{\chi}))) \quad (21)$$

After the sensitivity is computed, the algorithm can proceed to update the design variable χ_e . However, in the typical topology optimization algorithm, a process called filtering is often added to smooth the density between neighbouring blocks. To achieve this, an h th-order neighbourhood graph needs to be imposed, which can be obtained by computing the h th-order adjacency matrix, $\tilde{\mathbf{A}} = \mathbf{A}^h$.

$$\tilde{\sigma} = \tilde{\mathbf{A}}(\sigma \odot \chi) \odot (\chi \odot (\tilde{\mathbf{A}}\mathbf{1}^T)) \quad (22)$$

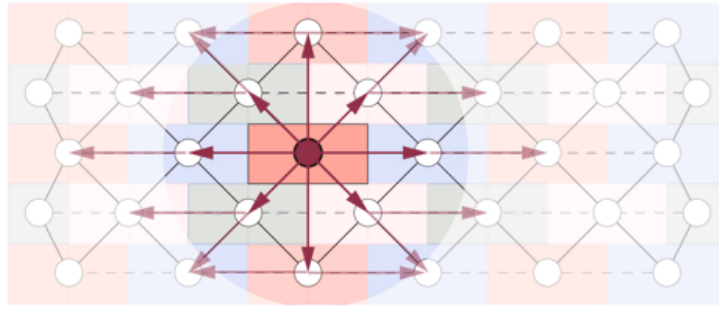


Figure 27

filtering

In Eq. (22), $\tilde{\mathbf{A}}$ represents the weighted factor that linearly reduces depending on the path length from each block e to its neighbouring blocks. In the smoothing scheme, the constraint for design variable $\chi_e = 10^{-3}$ is set as in Eq. (19).

The last step of the DEM-based optimization loop is to update χ_e by using the smoothed sensitivity.

$$\tilde{\chi} = -\tilde{\sigma} \odot \chi \quad (23)$$

Additional care needs to be taken to guarantee that the updated design variable χ_e still falls between χ_{min} and 1. This includes three steps: relativize, equalize and softening.

$$\hat{\chi} = \frac{(1 - \delta)m^{(0)}}{\mathbf{1}^T(\tilde{\chi} \odot V)} \tilde{\chi} \quad (24)$$

This step relates the design variable χ_e to the target mass fraction. Since there are three types of blocks which are full, half and quarter, mass differences for each block need to be taken into account. V is a

$n \times 1$ vector represented mass for each block, $m^{(0)}$ is total mass and δ is a removing fraction for every iteration.

To equalize, a softmax function is introduced. The softmax function takes a vector \mathbf{z} as input and normalizes it into a probability distribution. After applying softmax, each component will be in the interval (0, 1), and the components will add up to 1 (“Softmax function. Wikipedia.” [n.d.](#)). So that a equalize the distribution is obtained.

$$\tilde{\mathbf{z}} = \frac{1}{\mathbf{1}^T \mathbf{z}} \mathbf{z} \quad (25)$$

where

$$\mathbf{z} := \exp(\widehat{\chi})$$

The target mass fraction for each iteration also being considered a scalar. In this case, the total design mass is guaranteed.

$$\widehat{\mathbf{z}} = (1 - \delta)n\tilde{\mathbf{z}} \quad (26)$$

However, there is still a possibility that some of the design variables are larger than 1. Another step called softening needs to be done to make sure all design variables are between (0, 1].

while $\max(\widehat{\mathbf{z}}) > 1$:

$$\widehat{\mathbf{z}} = \frac{\mathbf{1}^T (\widehat{\mathbf{z}} - \min(\mathbf{1}, \widehat{\mathbf{z}}))}{n} \cdot \mathbf{1} + \min(\mathbf{1}, \widehat{\mathbf{z}}) \quad (27)$$

This step is set as a while loop which would help with keeping χ_e in range (0, 1].

Till here, the DETO loop is completed. The optimization loop would repeat until the difference between the new iteration and the precious iteration for each particle is small enough, for example 10^{-3} .

3.3.4 Algorithm design

The implementation of the DEM-based Topology Optimization is developed mainly based on scipy, numpy and networkX, and visualized through pyvista. The algorithm will be described with pseudocode.

User input

The algorithm starts with user input including the dimension of voxel unit in x,y,z direction and the design space. The material properties can also be defined by user, such as density, Young's modulus and friction coefficient. Also, the parameters that related to topology optimization loop, likes penalization power and volume fraction, can also be decided. After the user inputs all these values, the algorithm would translate useful information and generate a graph network with all the necessary matrices that would be applicable in the following stages.

Based on the default aggregation pattern, the centroids of blocks can be generated. Another advantage is that with the coloring voxel, the number of unit voxel that consist a block can be easily obtained, which is useful in the later displacement calculating process. Then the graph can be built with the adjacency matrix and the blocks position index (pseudocode01).

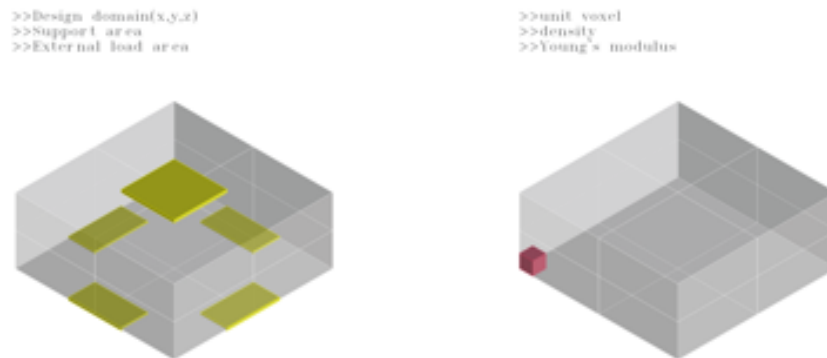


Figure 28

user input

```
BlockCentroid = np.mean(VoxelsCentroid)
BlockMass = len(UnitVoxel)*mass
WeightedAdjacencyMatrix = neighborlist*weight
Graph = nx.Graph(WeightedAdjacencyMatrix,BlockCentroid)
```

Figure 29

pseudocode01: generating network

The value `ne` is created as the number of nodes and the `chi` is a design variable vector with the target volume fraction `volfrac`.

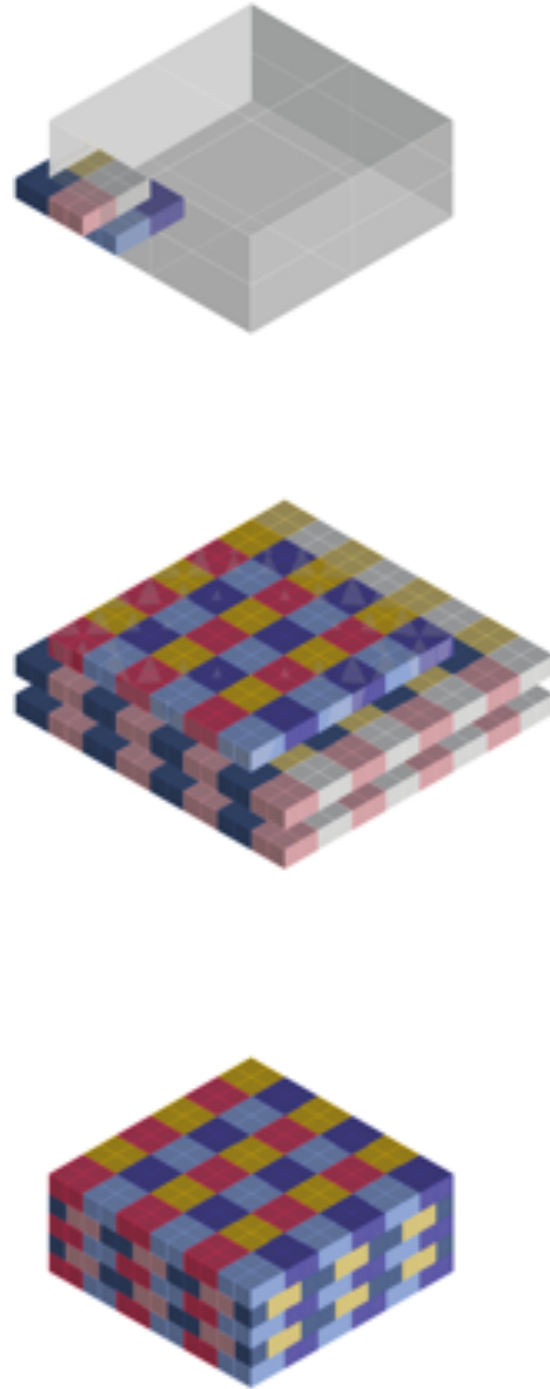


Figure 30

generating blocks based on the default pattern

```

# the number of vertices
ne = len(Graph.nodes)
# the number of edges
nedges = len(Graph.edges)

# Allocate design variables (as array)
chi = volfrac.repeat(ne)
#Create supports
fixed = [support1,support2,...,supportn]
free = BlockCentroid.setdiff(fixed)
#Set load
f = SelfWeight[BlockID] + ExternalLoad[load1,load2,...,loadn]

```

Figure 31

pseudocode02: supports and load case

The node IDs chosen by the user for applying supports and external loads would be used to get the small displacement of the nodes and solve the DEM-based minimization problem. The total force includes external force and self-weight applied on each node. A list of supports is chosen as a **fixed** vector and therefore the nodes in this list would not move during the structural analysis. Those nodes in the **free** list are free to move in x,y,z directions(pseudocode02).

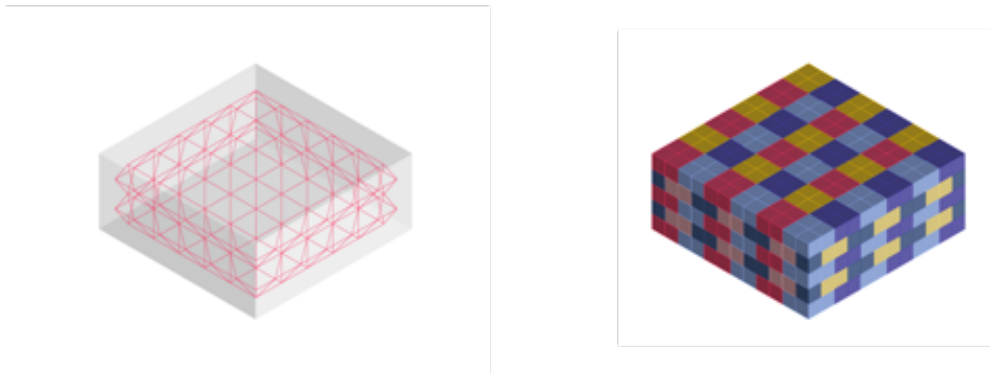


Figure 32

The structural network (dual network)

One of the most important matrix in this algorithm is the incidence matrix and the order of the nodes index needs to reorder based on the support IDs selected by user(pseudocode03).

Calculating displacement

The next step is to get the displacement values. Based on the proposed mathematic methodology, the

```

#incidence matrix
M = nx.incidence_matrix(Graph)
orderid = np.append(free,fixed)

#reorder the column of incidence matrix
M_s = M[orderid]
M_free = M[free]
M_fixed = M[fixed]

```

Figure 33

pseudocode03: incidence matrix

contact stiffness for the strong connection can be calculated and the rest of the stiffnesses are a linear relationship with it. Now, all the necessary matrices and vectors are ready. The displacement of blocks can be calculated by using the preset scipy function `sp.linalg.solve(A,b)`.

Optimization loop

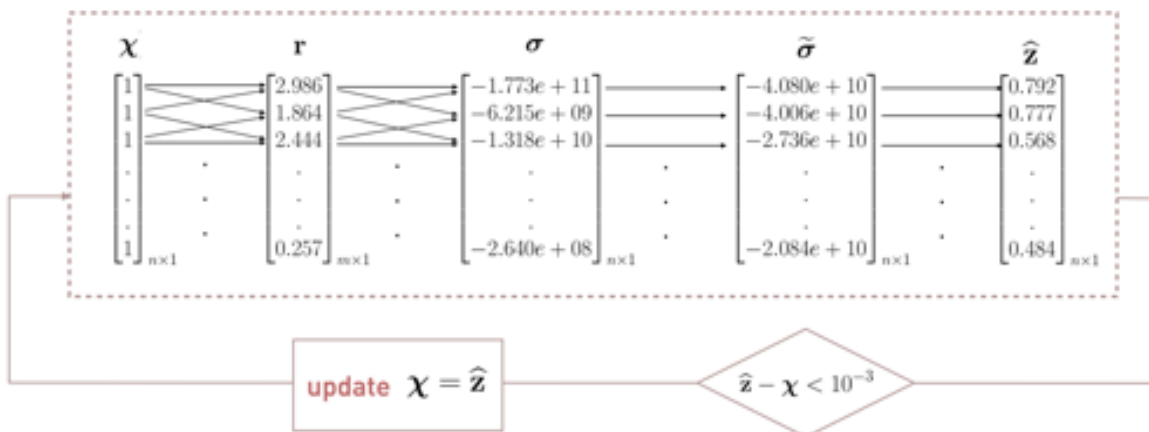


Figure 34

the optimization loop example

After all the matrices are known and compute in the right form, the topology optimization loop can start. To find the lowest compliance, the objective function is used and achieve with the set volume fraction. The gradient of the cost function is calculated for each pair of blocks. The solver is built based on the proposed method, which would return a new χ for the next loop until the convergence.

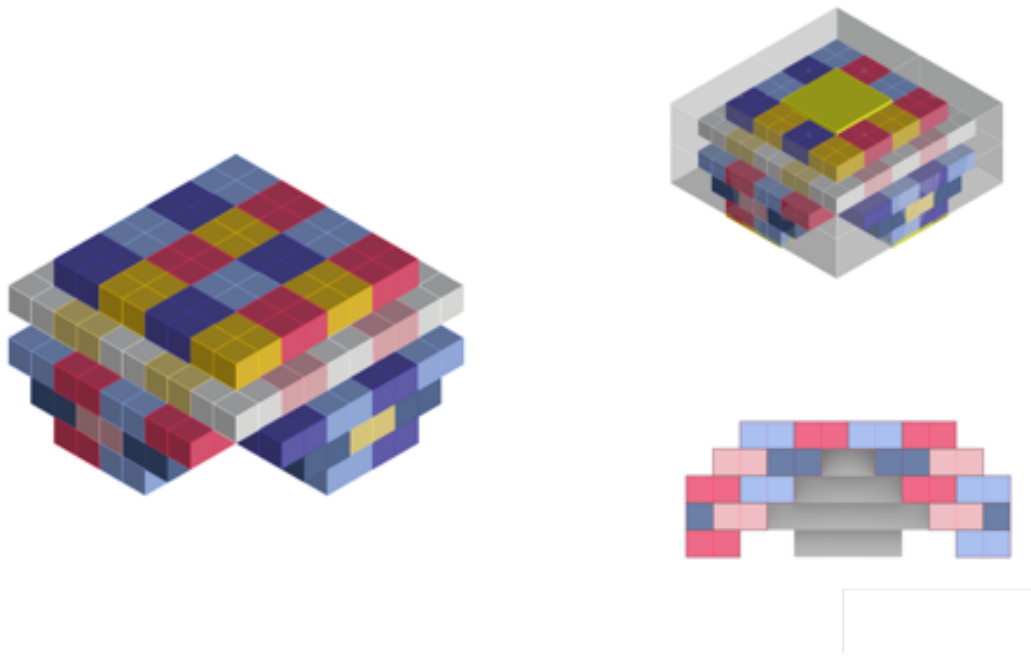


Figure 35

the optimization result for this example

3.3.5 Toy problems

This section will elaborate on the results obtained from proposed DETO method. Since there is no existed commercial software that can be used to check the validity of the results, the results would be compared with the vault generated by using the force density method in COMPAS (Mele & many others, [2017-2021](#)).

Toy problem 1

The lattice domain for the first toy problem is $0.8 \times 0.6 \times 0.4m$ which generated by $0.1 \times 0.1 \times 0.1m$ voxel unit. The material properties include density and Young's modulus is based on the standard brick properties. The external load applied in this example is the snow load. Details for the inputs can be found in fig.36.

Block & applied loads		
Voxel unit dimension Height×Weight×Length [m×m×m]	0.1×0.1×0.1	
Number of voxels for single brick [-]	4	
lattice domain [m×m×m]	0.8×0.6×0.4	
Snow load [kN/m ²]	Characteristic value of snow load (s _s)	0.45 EN 1991-1-3 (zone A1)
	safety factor	1,5
Material Property		
Density [kg/m ³]	1800	
Young's Modulus [kN/m ²]	1,55E+07	

Figure 36

Toy problem 1: material properties, design domain and applied load

Fig.37 shows the area for supports and applying external load. The blocks at the four bottom corners are fixed. Also, the penalty and mass fraction are determined at the beginning. Numerical experiments demonstrated that $p = 3$ provides a good compromise.

```
>>Design domain: (6,8,4)
>>Void: None
>>Support area
>>External load: snow load
>>External load area
>>mass fraction: 0.7
>>penalty: 3
>>
```

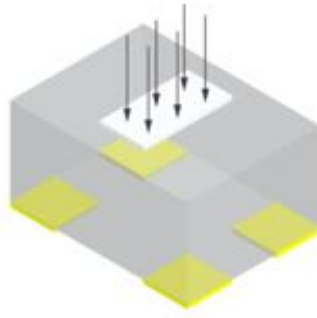


Figure 37

Toy problem 1: inputs of the design space, support areas and load region

The output of the algorithm is a list of nodes that can be visualized by using the script developed in GHpython. The result for this test case is obtained after 9 iterations. Although the result is not symmetrical, it seems logical. And the compliance got minimized.

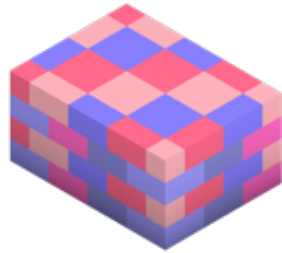


Figure 38

Toy problem 1: input geometry

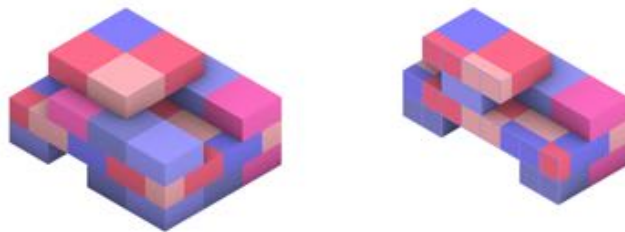


Figure 39

Toy problem 1: output geometry and its section

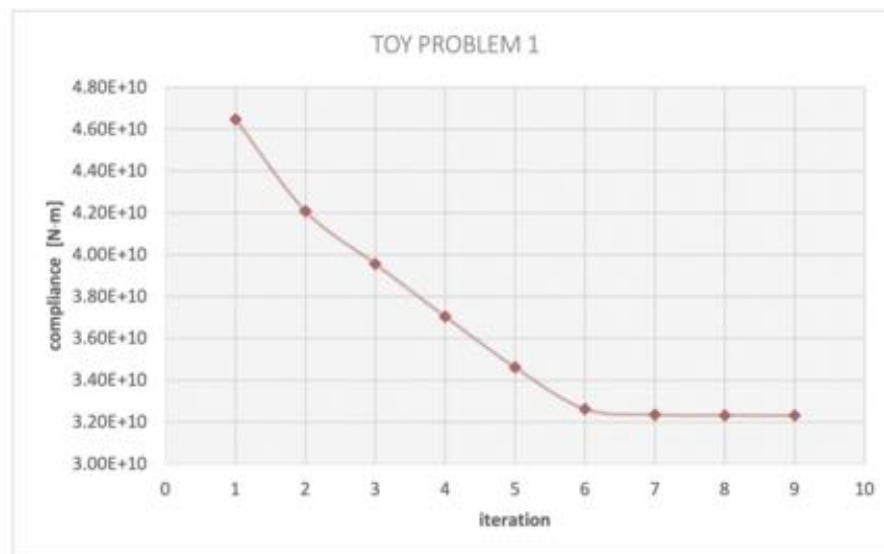


Figure 40

Toy problem 1: compliance outputs

Toy problem 2

In order to further develop the algorithm that can be used in architectural design, the void needs to be added. Similar to previous test, the material properties are the same. For the external load, live loads is also added in this case. The design domain is $2 \times 1 \times 1m$ in this case.

Block & applied loads		
Voxel unit dimension Height×Weight×Length [m×m×m]	0.1×0.1×0.1	
Number of voxels for single brick [-]	4	
lattice domain [m×m×m]	2×1×1	
Snow load [kN/m ²]	Characteristic value of snow load (s _k)	0.45 EN 1991-1-3 (zone A1)
	safety factor	1,5
Live load [kN/m ²]		3
	safety factor	1,5
Material Property		
Density [kg/m ³]	1800	
Young's Modulus [kN/m ²]	1,55E+07	

Figure 41

Toy problem 2: material properties, design domain and applied load

After filling the design space with the tessellation pattern, the next is to compute the void. This needs to be done before constructing the dual graph. The centroid of blocks are detected and if they locate within the void region, the blocks would be deleted. Then the network can be built.

With penalty equals 3 and 0.7 as mass fraction, the result seems reasonable. This is achieved by fixing ceiling and the area that applying external load. The compliance is convergence at the 13rd iteration. As it was shown in the horizontal sections, at the lower level where normally consider as columns, still not symmetric. This can probably caused by the tessellation pattern.

However, since the ceiling is fixed in this case, those blocks that above the void height can not be removed. When trying to release the fixed ceiling, the design variables χ_e result in almost the same density except for the fixed blocks. And it is not reasonable to remove the blocks with lower density

```
>>Design domain: (20,10,10)
>>Void
>>Support area
>>External load: snow load + live load
>>External load area
>>mass fraction: 0.7
>>penalty: 3
>>
```

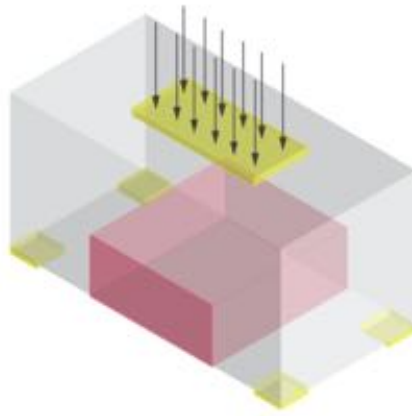


Figure 42

Toy problem 2: inputs of the design space, void, support areas and load region

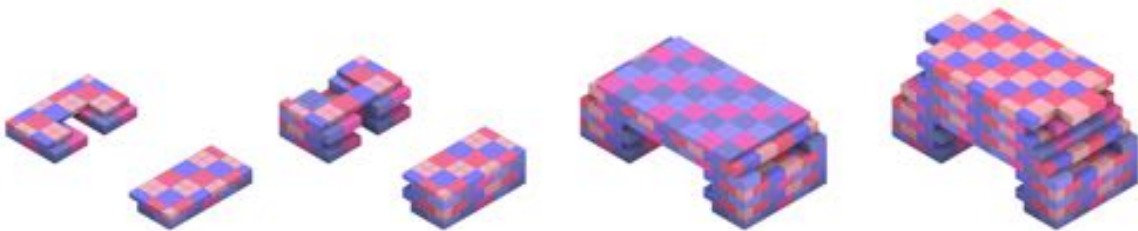


Figure 43

Toy problem 2: horizontal sections through the topology optimization result

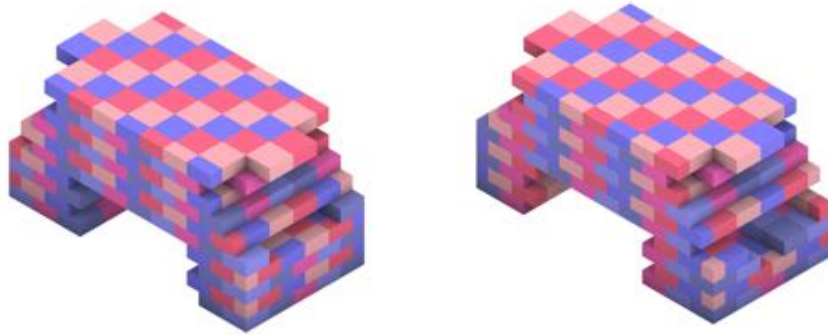


Figure 44

Toy problem 2: output geometry

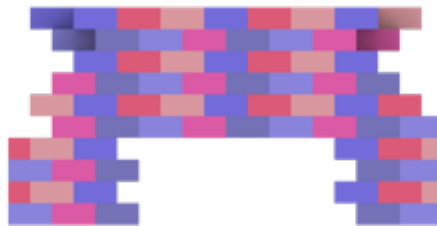


Figure 45

Toy problem 2: output geometry elevation

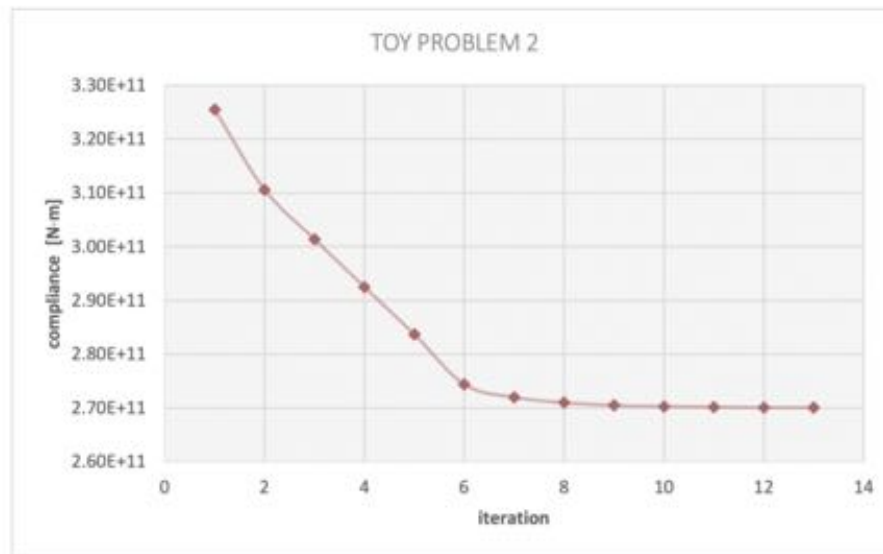


Figure 46

Toy problem 2: compliance outputs

when the density is really closed with each other. The reason why this situation occur might be the last step of the topology optimization loop, namely softening after using the softmax function. The last part of the mathematical theory needs to be redeveloped to avoid this sort of problems.

Also, the traditional OC method that used in SIMP was tried. However, this result in a worse situation including the compliance increasing and the design variable can not be controlled between 0 and 1. The result for this happened still can not be figured out.

4. Conclusion

The graduation topic is Topology Optimization for Discrete Funicular Structure, within the scope of the Genesis Lab topic Structural Topology Optimization. It is part of the research framework of Chair Design Informatics, within the track of Building Technology. This graduation project seeks to use the computational method to bridge structure mechanics, architecture design and computer science. It proposes a method for architects to generate funicular structures by providing a more efficient and integrated workflow. The aim of this project is to develop a topology optimization methodology as a generative design process for configuring masonry structures consisting of stackable interlocking blocks modelled as discrete elements using the Discrete Element Method by approximating its mechanical behaviours.

Topology optimization is widely known as a methodology for generating geometrically elaborate structures, which typically minimize the use of the material. These approaches typically use the Finite Element Method to formulate and solve the governing differential equations for computing their objective functions, assuming that the structure to be designed is a virtually continuous distribution of material that is refinable within a continuum. However, at a more general level, the idea of topology optimization can also be applied to inherently discrete problems by creating an algorithm based on Discrete Element Modelling.

The relationship between research and design

The first phase is broad research on topology optimization applied in structural design and understanding discrete element modelling. From these initial studies, the topic is able to narrow down and have a potential method to formulate the mechanical behaviours of the block. Since topology optimization was born for continuous material, most papers are based on it. The DEM-based topology optimization is state of the art, with very little research done so far. The Discrete element topology optimization provided a framework for combining DEM and topology optimization. However, applying to building scale has not been explored before, which makes this graduation topic full of challenges and experimental. Therefore, it is essential to understand convex optimization algorithms and find a proper solver for this project. Also, the mathematics and algorithms of Discrete element modelling need to be studied in order to approximate the mechanical behaviours between pairs of particles.

Method and argumentation

In achieving the objective of this, several sub-objectives are created. The first sub-objective is about how to configure the interlocking blocks. Several network systems are explored with the concept of the topological interlocking system for designing the space-filling system. Then, structural analysis is done to understand the mechanical behaviours of blocks and to finalize the general space-filling system. A connected cloud of particles represents a redefined structural network to discretize space by colouring the voxel grid. The current colouring method considered all the blocks to have the same dimension, which might cause structural inefficiency. The way to colour voxels can have a better solution to improve the structural performance. For example, the height of blocks at the lower layer could be larger.

In the numerical implementation process, the main target is to find the displacement value and stiffness matrix. Various methods to formulate the function are explored, and in the final version, a method developed based on the force-density method is used. The benefit of this method is that the self-weight



Figure 47

improve colouring voxel configuration

can easily be added to the nodes, which is usually a complex problem to solve in the FEM-based Topology Optimization method. To apply discrete element modelling, the formulation of the stiffness matrix is separated into three degrees of freedom, which might make the code more complicated. Another approach based on FEM to get the three-dimensional stiffness matrix can be explored to simplify the code.

After the numerical implementation, where each particle stores geometrical and mechanical data of blocks, the second phase is to assemble a topology optimization algorithm, using the gradient-based method adapted to work with the DEM-based governing equation and objective function and related gradient equation. The K matrix in compliance with Eq.20 only accounts for the stiffness acting on the edge direction, which is the normal stiffness. It would be better to take the shear stiffnesses in the perpendicular direction of the normal vector into account to have a more accurate result. Also, the constraints for the topology optimization process are standard operations. It is also possible to consider the friction and shear constraints since, in reality, the blocks are interlocking. According to the limit analysis of masonry, friction and shear constraint applied at block interfaces provides better insight into what happens in the contact area and give a better result.

The gradient-based method proposed in this research has a issue that the design variables might end up to the same value that related to the volume fraction. The mathematical function still needs to improve to avoid this problem. Another topology optimization solver in the typical SIMP method is the most popular and fundamental one, called Optimality Criteria, with a shorter calculation time. This technique is also be tested by changing part of the algorithm to adapt to proposed DETO method. However, OC method also fail in minimizing the compliance.

Discussion

Since the scale of toy problem 1 is too small to have a comparable with other results, we compare the result of toy problem 2 with a case that similar to this, which generated by using a commercially available software Ansys based on SIMP method. This case was done by Ivan during her thesis (Avdić, [2019](#)). Although the design domain is slightly different, it is easily to tell that the geometry is similar. The loaded area remained unchanged, which is almost the same as toy 2 result. The columns is visible and the upper half of the geometry seems alike. However, the lower part seems have more material removed around the ceiling, which is also what I expect to happen. Therefore, for the mathematics and algorithm design, it is fair to say that still have to be improved.

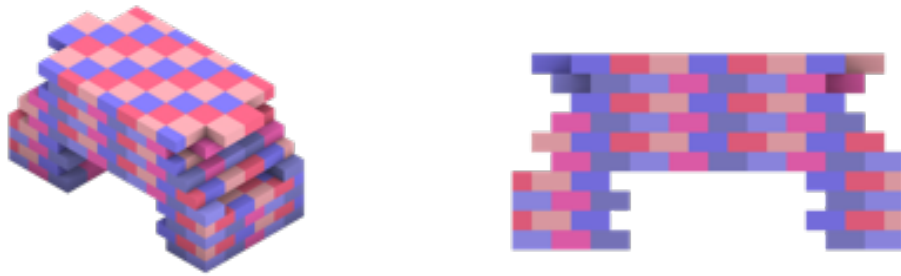


Figure 48

toy problem 2 output

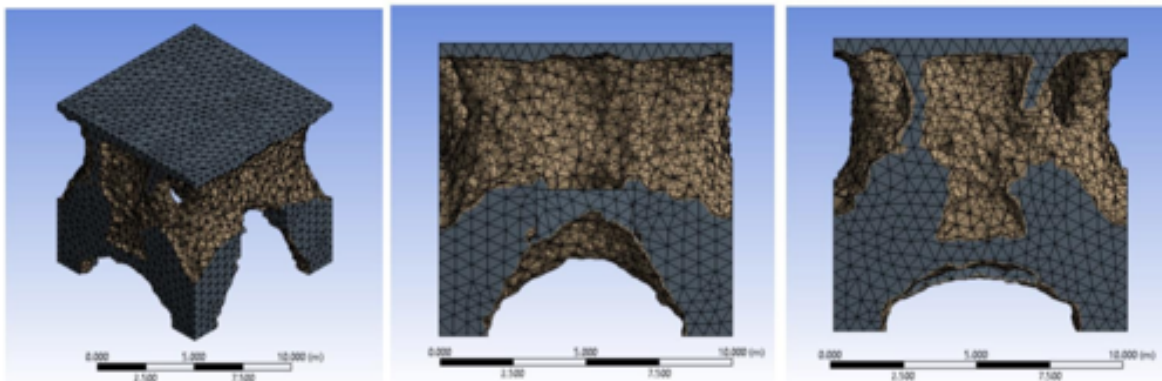


Figure 49

Ansys results for the case generated by Ivan (Avdić, [2019](#))

Applicability

The main research question was "How to implement discrete building blocks into the topology optimization to design funicular structures for architecture applicable for later construction processes?". This thesis provides a solution to it. The proposed methodology allows designers to find static equilibrium

configurations for funicular structures defined by their desired space by minimizing interaction energy between blocks. The masonry structure can be constructed basically following the configuration of each layer, which is the output of the Discrete Element Topology Optimization. Designers can also optimize the blocks' shape to decrease further the material used.



Figure 50

application

The objective of this research is to design a topology optimization method based on the DEM approach. The process is devised to result in funicular structures that can be built using a limited set of modular masonry blocks with the aim to lower the costs of production in terms of embodied carbon, monetary costs, and construction labour on the one hand and to increase the reuse and reconfigurability potential of the stackable blocks by seeking utmost modularity in the topological design of the underlying 3D tiling/tessellation.

It is urgent to transit the construction industry toward a more sustainable future in this era. This graduation topic from the Building Technology track gave me an opportunity to explore such a novel and experimental topic, which is full of challenges and unknowns. I am full of passion for developing an approach that can provide a more sustainable solution for masonry structural design. Since this topic has not been explored on the building scale, it is important to first understand the theory behind both Topology Optimization and Discrete Element Method technique. Before this thesis, I knew nothing about topology optimization, graph theory, DEM, FEM and the relevant mathematical operations. Also, there is no python code I can refer to, so I learned all the necessary python libraries to achieve my objective. To sum up, I really learned a lot during my thesis project and still have deeply interested in this topic. As

I discussed in the previous part, there are many places that can be improved; I think I would continue working on it to enhance the mathematical design and have a simpler code.