Structural and Multidisciplinary Optimization
https://doi.org/10.1007/s00158-019-02214-w

RESEARCH PAPER

Optimising the load path of compression-only thrust networks through independent sets

A. Liew[1] · R. Avelino[1] · V. Moosavi[1] · T. Van Mele[1] · P. Block[1]

Received: 25 September 2018 / Revised: 22 January 2019 / Accepted: 27 January 2019

Abstract
This paper presents network load path optimisation for the weight minimisation of compression-only thrust networks, allowing for the design of material efficient surface structures. A hybrid evolutionary and function-gradient optimisation process finds the optimal internal force state of the network, by manipulating the force densities of a selected number of edges based on the network indeterminacy. These selected edges are the independent sets, and are found through the Reduced Row Echelon form of the network's equilibrium matrix. It was found that networks can have certain independent sets that have a significant influence on both the stability of the optimisation algorithm, and in the final load path/volume of the structure. Finding the most effective independent sets was handled by data-driven methods, applied to many thousands of independent set trials. This provided insight into the behaviour of the underlying network and dramatically increased the rate of finding successful independent sets. The importance and weights of the network edges highlighted key areas of the network that allowed structural judgement and improvements to be made.

Keywords Reduced Row Echelon · Equilibrium matrix · Optimisation algorithm

1 Introduction

This paper demonstrates the minimum weight optimisation of statically indeterminate structures represented by networks of vertices and edges of given topology working exclusively in compression. The compression-only networks are subject to fixed boundary supports and point loads applied to vertices and can be used to represent surface structures such as shells. Such networks can be generated in engineering design through form-finding methods such as Thrust Network Analysis (TNA) (Block and Ochsendorf 2007; Block 2009; O'Dwyer 1999; Fraternali 2010; Marmo and Rosati 2017) and related tools like RhinoVAULT (Block Research Group 2014), where a network geometry is found that is in equilibrium with the applied loads and support conditions. TNA, particularly through the interactive use of RhinoVAULT, has been responsible for the design of many compression-only vaulted structures, but not with the explicit inclusion of weight minimisation by optimising the internal force state.

An example of a compression-only thrust network representing an anti-funicular shell structure is shown in Fig. 1, where the four corners are fixed from translating in all directions and the loading is based on area-weighted point loads applied at the network's vertices, used to represent the self-weight of the shell.

The external point loads that are applied to the vertices of the thrust network in Fig. 1 are taken to the supports through the force densities (Linkwitz and Schek 1971; Schek 1974) of the network edges $\mathbf{q}$, which are the internal edge forces $\mathbf{f}$ divided by the edge lengths $\mathbf{l}$. The force densities of this example is shown in Fig. 2, with the thickness of the edge lines representing the flow of the forces in the thrust

Responsible Editor: Somanath Nagendra

✉ A. Liew
liew@arch.ethz.ch

R. Avelino
maia@arch.ethz.ch

V. Moosavi
moosavi@arch.ethz.ch

T. Van Mele
van.mele@arch.ethz.ch

P. Block
block@arch.ethz.ch

[1] Institute of Technology in Architecture, ETH Zurich, Stefano-Franscini-Platz 1, 8093 Zurich, Switzerland

Published online: 13 February 2019

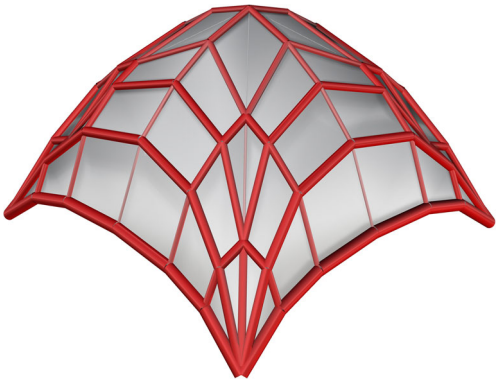

Fig. 1 An example compression-only thrust network, which is supported at each corner and subjected to area-weighted point loads applied at the vertices

network, scaled proportionally for a strength sizing of each edge having their area proportional to the axial compression force.

Due to the statical indeterminacy of the thrust networks, there are an infinite number of different ways that the forces may be transmitted through the network and maintain equilibrium. Some of these internal force distributions will be better than others when the structure comes to be materialised, and so metrics are needed to quantify how well each performs compared to its peers. The problem of optimising truss structures for the metric of mass minimisation has been studied since Michell (1904). From then, various research has been conducted in optimising network-type structures such as trusses, gridshells and thrust networks, generally with bar elements that can take compression (struts) and tension (ties). A selection of this research includes (De Wilde 2006; Jiang et al. 2018) application for the optimisation of building frames (Stromberg et al. 2018; Lu et al. 2018) to the improvement of structural trusses (Vandenbergh et al. 2006; Pyl et al. 2013; He and Gilbert 2015; Gilbert and Tyas 2003). In particular, the work by Mazurek et al. (2011), Baker et al. (2013) and Beghini et al. (2014) showed that for truss structures, based on Maxwell's formulae on load paths (Maxwell 1864), optimising for the minimal load path results in the minimal volume solution for a given stress level σ for the edges in the truss,

$$\min \sum_i V_i = \min \sum_i A_i \, l_i = \min \frac{1}{\sigma} \sum_i |f_i| \, l_i \,, \tag{1}$$

where V_i, A_i and l_i are the volume, cross-sectional area and length of edge i. Thus, for a given stress level σ, the minimum volume can be achieved by minimising the sum of the vector of edge forces $\mathbf{f}$ multiplied by the lengths $\mathbf{l}$. The load path is represented herein by ϕ, which is calculated as the product of the edge forces $\mathbf{f}$ with their lengths $\mathbf{l}$.

The determination and optimisation of a network's load path, and the associated distribution of force densities that leads to this optimum, are the focus of this paper. (1) poses no conditions on the sign of the forces, as the magnitude of the force is taken. This research considers the more challenging problem of optimising the load path specifically for compression-only thrust networks in 3D where the sign is consistently positive. Finding an optimal value of ϕ for this case is not trivial, as the search space on the force densities of the network edges is extensive. The task is further complicated when the network vertices are kept fixed in plan as an additional hard constraint. If one calculates the load path for the network in Fig. 2, the result $\phi = 1254$ is found. If instead one uses the procedures outlined in this paper, on exactly the same network with identical loads and support points, the optimised thrust network in Fig. 3 is generated.

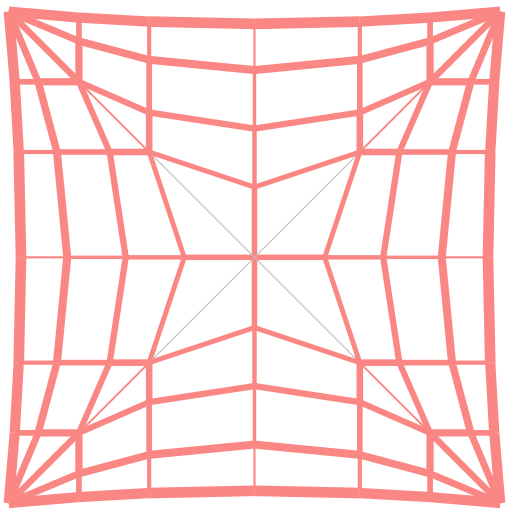

Fig. 2 Distribution of edge force densities $\mathbf{q}$ that leads to the thrust network in Fig. 1

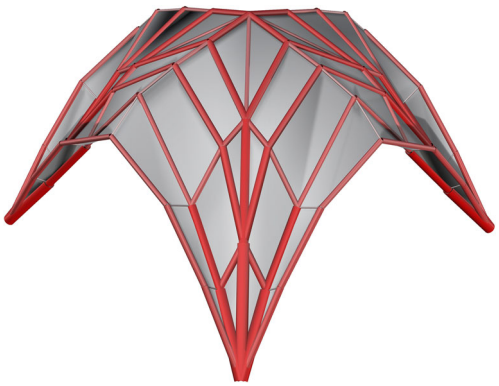

Fig. 3 Optimised thrust network featuring an improved distribution of force densities that leads to a lower volume of material for a given stress level. Notice the different geometry of the network for the same vertex co-ordinates in plan

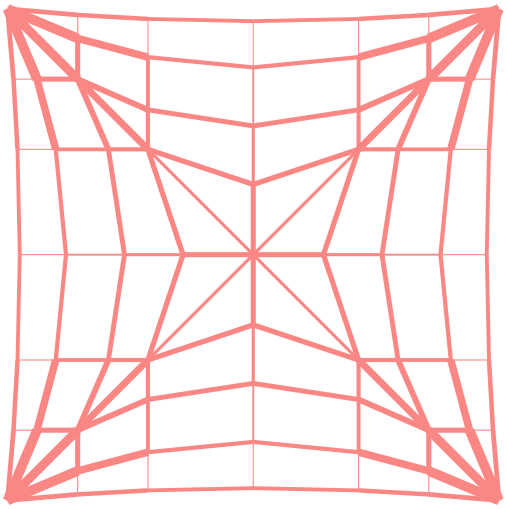

Fig. 4 The optimised distribution of network edge force densities that leads to a significant reduction in the structural weight of Fig. 3. This leads to a load path value of $\phi = 860$, an improvement of around 30% from the non-optimised case

By utilising optimal distributions of compression-only force densities, we take steps towards the minimum weight solution. Such a process is useful for example, for an architect or engineer wishing to optimise the material use in a compression-only shell structure represented by a thrust network. The thrust network in Fig. 1 was manually form-found with the user-defined distribution of force densities in Fig. 2, giving the load path value $\phi = 1254$. The load path is reduced to $\phi = 860$ for the network in Fig. 3 with the optimised force densities in Fig. 4. Both examples have the same structural height, so this economy in the material use doesn't come from simply increasing the height; instead, the thrust network is shaped to allow a more efficient path for the applied loads to be directed to the supports.

Rather than attempting to determine the optimal force densities of every edge in the network, the process can be made more efficient by only examining the statically indeterminate edges. This greatly reduces the complexity of the problem, as once all statically indeterminate edges are defined, there is only one solution for the distribution of force densities in all remaining determinate edges. This is a more economic approach as less degrees of freedom need to be considered, and extends upon the research by Liew et al. (2018).

This paper is presented as follows. Section 2 defines in detail the load path method and examines the statical indeterminacy of a network, and how both relate to the generation of a thrust network. Section 3 outlines the computational implementation of the load path algorithm and optimisation process. Section 4 examines the vast size of different statically indeterminate edge sets of a network, by the assistance of machine learning, to efficiently determine patterns and classify what leads to a good or bad choice of edge sets. Finally, Section 5 closes the paper with a discussion of the findings.

2 Thrust network load paths

The load path was introduced in Section 1 as the scalar ϕ, calculated as the product of the network's edge forces and edge lengths, or alternatively as the product of the network's edge force densities and the square of the edge lengths (2).

$$\phi = \mathbf{f}^{\mathrm{T}}\mathbf{l} = \mathbf{q}^{\mathrm{T}}\mathbf{L}^{\mathrm{T}}\mathbf{l}, \tag{2}$$

where the force densities are defined as

$$\mathbf{q} = \mathbf{L}^{-1}\mathbf{f}, \tag{3}$$

with $\mathbf{f}$ and $\mathbf{l}$ as the column vectors of forces and three-dimensional edge lengths of the network, and $\mathbf{L}$ as the diagonalised matrix from length vector $\mathbf{l}$.

The minimisation of ϕ is equivalent to minimising the volume of material for a truss structure with edge elements at a given stress level and for the given boundary conditions. The minimisation of ϕ as described in this research is also subject to the following assumptions: (1) there exist fixed vertical loads and no horizontal loads, with loads applied directly to the vertices as point loads, (2) the supports are co-planar fixed boundary vertices that cannot translate, here taken for convenience with z co-ordinates equal to zero, and (3) a fixed horizontal distribution of vertices, that is, the x and y co-ordinates stay fixed for every vertex throughout.

The calculation of ϕ requires a network that is in static equilibrium with the applied loads, and in this case, also with edges that are exclusively in compression. The three-dimensional framework to achieve this is by using Thrust Network Analysis (TNA). TNA is an equilibrium method, which extends the Force Density Method for the specific case of vertical loading and networks with fixed horizontal projection, and so is well suited for the task at hand. The necessary components from TNA that are used will be described in the following sections, starting with a discussion on network indeterminacy.

2.1 Network indeterminacy

When using TNA, one can generate many different equilibrium states by examining the paths through the network that static external loads can follow on their way to the supports. The different possibilities that the structure has on supporting the loads are related to the degree of statical indeterminacy of the network. If the network has high statical indeterminacy, there is a rich set of different internal force or force density distributions that can be found that lead to the structure being in equilibrium. However, to

maintain a fixed projection of vertices, i.e. to not change the x and y co-ordinates during the process of examining the different internal force states, internal forces cannot be chosen independently (Liew et al. 2018; Block and Lachauer 2014; Van Mele and Block 2014). An efficient method to tackle this problem setup is to find a vector of independent force densities $\mathbf{q}_\mathrm{i}$, for which the remaining dependent force densities $\mathbf{q}_\mathrm{d}$ can be calculated, so that vertices still remain in place in the x-y plane. This process is now described in more detail.

Consider the orthogonal network in Fig. 5, which has $m = 24$ edges, $n = 21$ number of vertices, of which $n_\mathrm{b} = 12$ are fixed boundary vertices (in blue) where translations in x, y and z are prohibited, and $n_\mathrm{i} = 9$ free internal vertices (in white).

A layout of independent edges for changing force densities $\mathbf{q}_\mathrm{i}$, with one independent edge per row and column, is shown in Fig. 6. As every edge force density in each of the groups of continuous edges is entirely independent from the force densities in any other continuous group, the state of equilibrium of the network can be described by independently choosing one force density (those in green) per continuous group of edges. This will ensure that all other force densities for the edges in red can be calculated, as equilibrium in x and y at each vertex can always be calculated for the applied loads.

For a general network with n_i number of free internal vertices, m edges and k number of independent edges for $\mathbf{q}_\mathrm{i}$, the dependent edge force densities $\mathbf{q}_\mathrm{d}$ can be calculated from

$$\mathbf{q}_\mathrm{d} = -\mathbf{E}_\mathrm{d}^{-1}\mathbf{E}_\mathrm{i}\mathbf{q}_\mathrm{i}, \tag{4}$$

in which the $(2n_\mathrm{i} \times m - k)$ non-singular square matrix $\mathbf{E}_\mathrm{d}$ and the $(2n_\mathrm{i} \times k)$ matrix $\mathbf{E}_\mathrm{i}$ are the sub-matrices of the equilibrium matrix $\mathbf{E}$. If all force densities $\mathbf{q}$ are positive, then the resulting solution is compressive. This correspond to the set of dependent and independent edges $\mathbb{M}_\mathrm{d}$ and $\mathbb{M}_\mathrm{i}$, with their respective force densities $\mathbf{q}_\mathrm{d}$ and $\mathbf{q}_\mathrm{i}$. The equilibrium matrix $\mathbf{E}$ is associated with the x-y plane and describes the horizontal equilibrium of each vertex based on the force density and geometry of the connecting edges. The equilibrium matrix $\mathbf{E}$ can be calculated by

$$\mathbf{E} = \begin{bmatrix} \mathbf{C}_\mathrm{i}^\mathrm{T}\mathbf{U} \\ \mathbf{C}_\mathrm{i}^\mathrm{T}\mathbf{V} \end{bmatrix}, \tag{5}$$

where $\mathbf{U}$ and $\mathbf{V}$ are the diagonal matrices of vectors $\mathbf{u}$ and $\mathbf{v}$, which are the co-ordinate differences in the x and y directions for the vertices of each edge. The connectivity matrix $\mathbf{C}$ has two sub-matrices $\mathbf{C}_\mathrm{i}$ and $\mathbf{C}_\mathrm{b}$, corresponding to the n_i internal vertices and n_b fixed boundary vertices, respectively. For a given network, the connectivity matrix $\mathbf{C}$ (sometimes referred to as the branch–node matrix) of size $(m \times n)$ is generally a sparse matrix and can be constructed to store the connectivity between edges and vertices, so that row i in the matrix for network edge i has two columns in the matrix marked with 1 and -1 for the indices of the vertices at the two ends, where the vertex deemed the start vertex is not governing.

2.2 Reduced Row Echelon method

For networks with non-orthogonal edge layouts, it will in general not be elementary to determine what edges can form the independent set $\mathbb{M}_\mathrm{i}$, and so a robust method using the Reduced Row Echelon Form (RREF) of $\mathbf{E}$ is described in

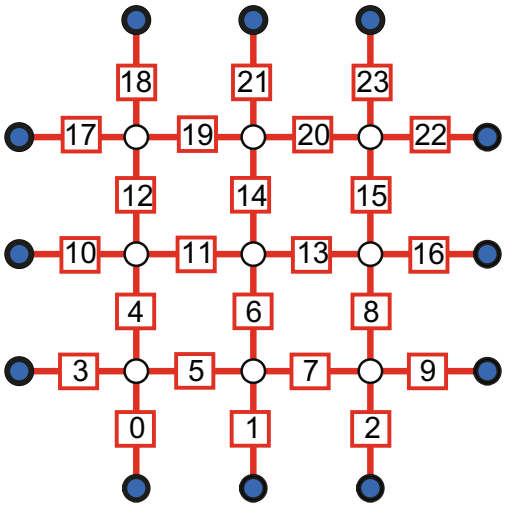

Fig. 5 A network consisting of 24 orthogonal edges and 21 vertices, with 12 fixed vertices at the perimeter in blue, and 9 free internal vertices in white

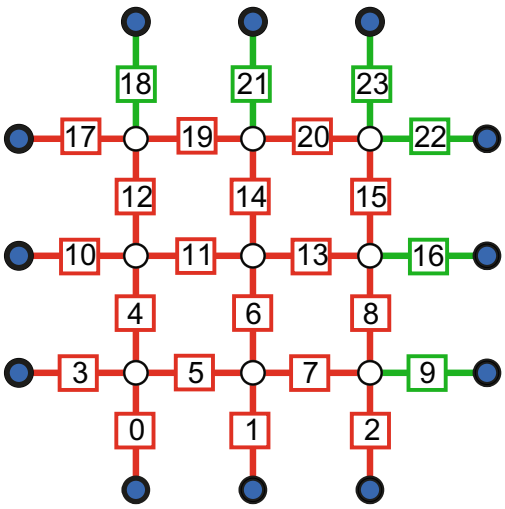

Fig. 6 Identification of a set of independent edges in green, with one independent edge per row and column of the orthogonal network

this section. The RREF of a matrix is described first, before showing how it may be used to identify the independent edges.

Converting a matrix into Row Echelon Form (REF), be it via QR factorisation or other methods such as Gauss-Jordon elimination, is used to (among other things) solve systems of linear equations by forward elimination and back substitution, to determine the rank of a matrix, and for computing the determinant and inverse of a square matrix. A matrix that is in REF will possess zeros in the lower left of the matrix and non-zero values in the upper right, with the first non-zero value of each row called the leading coefficient or the pivot, as it is selected for row operations. The REF will look in general like the following example matrix of rank 2, where elements 3 and 2 are the pivots and a_i are general values:

$$\begin{bmatrix} 3 & a_1 & a_2 & a_3 \\ 0 & 2 & a_4 & a_5 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{6}$$

The Reduced Row Echelon Form differs from the Row Echelon Form in that the leading coefficients are now unity and are the only non-zero entries in their respective columns. The RREF is unique to the matrix, whereas the REF of a matrix is not, as the latter depends on the manner of row operations that were previously performed. A matrix in RREF will in general look like:

$$\begin{bmatrix} 1 & 0 & a_1 & a_2 \\ 0 & 1 & a_3 & a_4 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{7}$$

For a selection of commonly used numerical software packages, the RREF functions that will convert a matrix into RREF are as follows: SymPy with `A.rref()` to return the $\mathbf{A}_{\text{rref}}$ matrix of $\mathbf{A}$ and pivot columns, in MATLAB as `rref(A)` for $\mathbf{A}_{\text{rref}}$, and `RowReduce(A)` in Mathematica.

By way of example, converting the equilibrium matrix of the original orthogonal network (Fig. 6) into the RREF of $\mathbf{E}_{\text{rref}}$, one arrives with the simplified matrix representation in Fig. 7, where the red elements are the unit pivots, the blue elements are the non-zero values, and the highlighted green columns are the non-pivoting columns.

For this orthogonal network and other general networks with set of edges $\mathbb{M}$, the subset of independent edges $\mathbb{M}_{\text{i}} \subset \mathbb{M}$ can be identified as the non-pivoting columns of $\mathbf{E}_{\text{rref}}$, which in this case are $\mathbb{M}_{\text{i}} = \{9, 16, 18, 21, 22, 23\}$. These are the indices of the plotted green edges of Fig. 6, corresponding to one independent edge per row and column line of the network. The subset of dependent edges $\mathbb{M}_{\text{d}} \subset \mathbb{M}$ are then the remaining pivoting columns. The RREF method is reliable for the automated selection of independent edges, however, since the shape of $\mathbf{E}$ is always $(2n_i \times m)$, its application is limited to networks containing a number of edges m greater than two times the number of internal vertices $2n_i$, i.e. for $\mathbf{E}$ having more columns than rows.

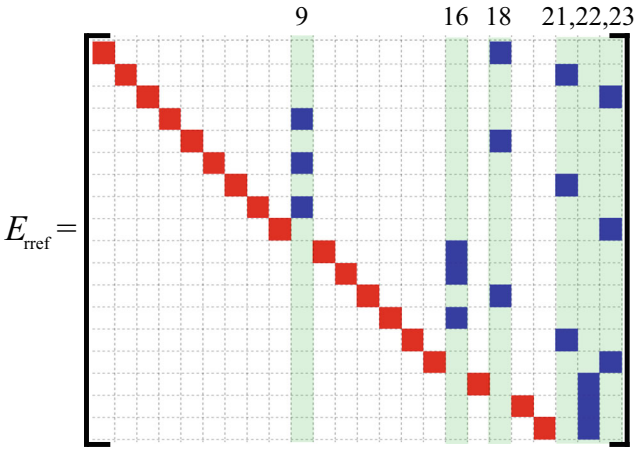

Fig. 7 Simplified diagram of the Reduced Row Echelon Form $\mathbf{E}_{\text{rref}}$ of the equilibrium matrix $\mathbf{E}$ for the orthogonal grid network of Fig. 6

For general non-orthogonal networks, such as that presented in Fig. 8, the determination of which edges will be independent is straightforward using the same process. The four additional edges that have been inserted increase the k number of independent edges by four. The addition of edges {24, 25, 26, 27} requires four more independent green edges such as {10, 19, 20, 26}, for the statical determinacy of the network.

2.3 Multiple independent sets

For the four simple networks with four edges drawn in Fig. 9, there is a single four-valent and free internal vertex (white) in the centre, which is connected to four fixed

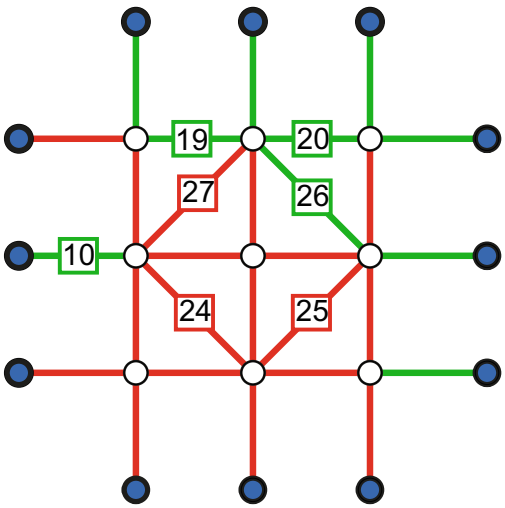

Fig. 8 Identification of four new independent edges after four additional edges are inserted into the orthogonal network

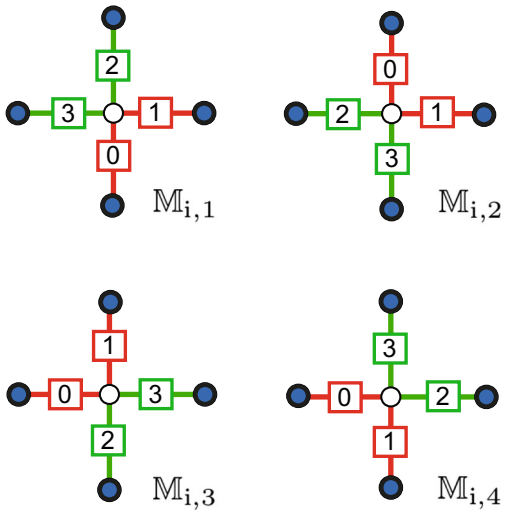

Fig. 9 Four independent edge sets $\mathbb{M}_{i,1}$, $\mathbb{M}_{i,2}$, $\mathbb{M}_{i,3}$ and $\mathbb{M}_{i,4}$ exist for this simple four-valent vertex example

boundary vertices (blue). One can see by inspection that there are four independent edge set combinations ($\mathbb{M}_{i,1}$, $\mathbb{M}_{i,2}$, $\mathbb{M}_{i,3}$ and $\mathbb{M}_{i,4}$) with each row and column of the network possessing one independent edge (green) for a total of $k = 2$.

From the RREF method, the matrix $\mathbf{E}_{\text{rref}}$ is calculated as follows, with pivots (dependent edges) as the set $\mathbb{M}_{\text{d}} = \{0, 1\}$, and non-pivots (independent edges) as the set $\mathbb{M}_{\text{i}} = \{2, 3\}$:

$$\mathbf{E}_{\text{rref}} = \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \tag{8}$$

Because this $\mathbf{E}_{\text{rref}}$ matrix is unique to the network, $\mathbb{M}_{\text{i}} = \{2, 3\}$ will always be chosen as the set of edges to represent the vector of independent force densities $\mathbf{q}_{\text{i}}$. Thus, to find the four different sets of independent edges, one can simply renumber the edges before the RREF is performed, as is shown in Fig. 9 for the four different combinations of green independent edges. In this simple example, edges 2 and 3 are always green; they are just in different locations in the network. It is very important to make the distinction between randomising the numbering of the edges and simply naively selecting edges to be in $\mathbb{M}_{\text{i}}$ at random. Simply randomly selecting edges to be in $\mathbb{M}_{\text{i}}$ is incorrect, as the situation will arise, for example in $\mathbb{M}_{i,1}$, that edges 0 and 2 would be selected as independent, which would over-define this direction and only give equilibrium if both have equal force density. Consequently, if 0 and 2 are selected together, it means that there is no control of the force densities and hence equilibrium in the direction of the edges 1 and 3. This situation cannot arise with the RREF.

Due to the relationship between the independent sets and the network edge numbering, it is preferable to not use the randomised edge numbers of each network to describe the set. Instead, a unique static key should be used to represent each $\mathbb{M}_{i,j}$ set. This can be done by using a consistent fixed network numbering system decided upon at the start, or by a set representing the midpoints p of each independent edge such that set $\{p_1, p_2, \ldots p_i \ldots p_{(k-1)}, p_k\}$ are the midpoints of the edges.

Consider an orthogonal grid network of two rows and two columns, there will be four independent edges $k = 4$ and 81 independent sets for $\mathbb{M}_{\text{i}}$. For a three rows by three columns grid with $k = 6$, this escalates rapidly to 4096 combinations for $\mathbb{M}_{\text{i}}$. It is evident for large values of k, that it becomes unreasonable to investigate every possible independent set $\mathbb{M}_{i,j}$. One might expect that each $\mathbb{M}_{i,j}$ should lead to the same final optimised load path value ϕ; however, it has been found that the choice of $\mathbb{M}_{i,j}$ has a great influence on the convergence of the solution. This is examined later in more detail.

2.4 Vertical equilibrium

For the vertical loads $\mathbf{p}_{\text{z}}$ acting on the free internal vertices, vertical equilibrium can be found by changing only the heights of the free vertices $\mathbf{z}_{\text{i}}$. This can be performed as the force density vector $\mathbf{q}$ for all edges was calculated previously for horizontal equilibrium. This calculation is made by

$$\mathbf{z}_{\text{i}} = \mathbf{D}_{\text{i}}^{-1} \left(\mathbf{p}_{\text{z}} - \mathbf{D}_{\text{b}} \mathbf{z}_{\text{b}} \right), \tag{9}$$

with the $(n_{\text{i}} \times n_{\text{i}})$ matrix $\mathbf{D}_{\text{i}} = \mathbf{C}_{\text{i}}^{\text{T}} \mathbf{Q} \mathbf{C}_{\text{i}}$ and the $(n_{\text{i}} \times n_{\text{b}})$ matrix $\mathbf{D}_{\text{b}} = \mathbf{C}_{\text{i}}^{\text{T}} \mathbf{Q} \mathbf{C}_{\text{b}}$. With the simplification of taking zero as the height of all boundary points, then $\mathbf{z}_{\text{b}} = 0$ and we then have

$$\mathbf{z}_{\text{i}} = \mathbf{D}_{\text{i}}^{-1} \mathbf{p}_{\text{z}}. \tag{10}$$

Once the $\mathbf{z}_{\text{i}}$ vector has been calculated for the network, the spatial geometry is fully defined and the length vector $\mathbf{l}$ can be found and combined with $\mathbf{q}$ to give the load path ϕ.

2.5 Symmetry

Optimising the load path of a network with a lower number of independent branches k will lead to a faster algorithm run time, as the number of dimensions in the search space of $\mathbf{q}$ will be reduced. If the network to be analysed has clear lines of symmetry, they can be exploited with great effect to both reduce k and enforce symmetry in the force densities vector. This is easily handled by adding dedicated helper symmetry edges at the symmetry interface, like those plotted in yellow in Fig. 10. These helper edges are available for the network to provide the necessary equal and opposite opposing forces in the x-y plane for horizontal equilibrium. The lengths of these helper symmetry edges as seen in plan do not matter, as the reaction forces that these edges generate are realised through their force densities and so normalised by

length. Figure 10 shows two possible network arrangements for symmetrically analysing a five by five orthogonal grid. The added symmetry edges are still part of the connectivity of the network and so still contribute to the equilibrium matrix **E**. Therefore, they may also appear within the set of independent edges $\mathbb{M}_\mathrm{i}$ during the RREF process.

The force densities of the helper symmetry edges calculated for horizontal equilibrium should not influence the z co-ordinates when the thrust network is formed for vertical equilibrium. So they may be turned off by setting the force densities to zero after horizontal equilibrium is achieved. In response to this, the $\mathbf{z}_\mathrm{i}$ co-ordinates of the free internal vertices will ignore the fixed vertices used to anchor down the symmetry edges, as they are part of the boundary vertices $\mathbf{z}_\mathrm{b}$. It must be ensured that the external loads $\mathbf{p}_z$ that are applied to the free internal vertices represent the correct tributary area loads of the considered symmetry side, so as not to double count loads for the vertices at the symmetry interface from the other side.

3 Algorithm

For a network that is in equilibrium from a particular set of independent force densities $\mathbf{q}_\mathrm{i}$, the load path ϕ can be calculated from Section 2. The aim now is to find the network's optimum set of independent edge force densities, for which the load path is minimised. The management of the network's independent sets and the convergence of the algorithm to lead to a good load path value is dependent upon a robust optimisation solver as described in Section 3.1, and effective penalisation and constraints measures for removing tension in the network edges explained in Section 3.2.

The optimisation algorithm is based on the main steps described in Section 2 for the calculation of ϕ, these are:

- Determine the independent set(s) to be analysed by the RREF method.
- Ensure horizontal equilibrium in x-y by finding the vector of dependent edge force densities $\mathbf{q}_\mathrm{d}$ from a trial vector of independent edge force densities $\mathbf{q}_\mathrm{i}$.
- Update the z co-ordinates of the free internal vertices to generate a thrust network.
- Calculate the load path ϕ for this equilibrated spatial thrust network.
- Examine the value of ϕ and the distribution of all force densities $\mathbf{q}$, to examine which, if any, are in tension.

The input data from the user for this process are the applied vertical loads $\mathbf{p}_\mathrm{z}$ at the free internal vertices, the planar x-y co-ordinates of all vertices for a fixed horizontal projection, and the connectivity matrices derived from the topology of the network.

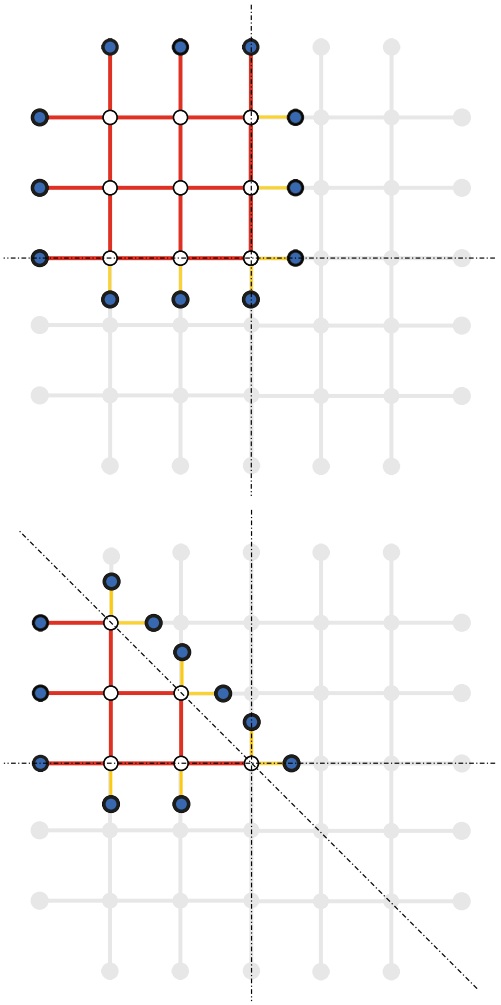

Fig. 10 Symmetry can be used on this orthogonal network so only one quadrant or triangular half quadrant needs to be analysed. This will reduce k and enforce two or three lines of symmetry

The formal statement of the optimisation problem is as follows. The objective function is the load path function $\phi(\mathbf{q}_\mathrm{i})$, which is to be minimised subject to the following two constraints: (1) that all independent force densities $\mathbf{q}_\mathrm{i}$ are greater than zero (compression taken as positive) and less than a user-defined maximum value q_max and (2) that all dependent force densities $\mathbf{q}_\mathrm{d}$ are also greater than zero (q_max need not be necessarily applied here). In order to put some bounds on the search space to prevent it from becoming too large, a value of q_max in the range of [5, 10] has been found to be effective. In mathematical notation, the optimisation can be written as follows, where the solver(s) used for this problem are described in more detail in the next section.

$$\begin{aligned} &\min\ \phi(\mathbf{q}_\mathrm{i}) \\ &\text{s.t. } \mathbf{q}_\mathrm{d} \geq 0 \\ &\quad 0 \leq \mathbf{q}_\mathrm{i} \leq q_\mathrm{max} \end{aligned} \tag{11}$$

3.1 Optimisation solver

A variety of established and custom solvers have been investigated to aid in the optimisation process, both function only, and function- and gradient-based methods. The optimisation problem is constrained as all force densities must be positive for a compression-only solution, and the problem is also multi-variate with k degrees of freedom. The implementation of the load path optimisation has been performed with solvers using the numerical and scientific NumPy (Van der Walt et al. 2011) and SciPy (Jones et al. 2001) packages and using custom algorithms for the Python programming language (Python Software Foundation 2017).

The optimisation process starts with an evolutionary method using either the COMPAS (Van Mele et al. 2017) implementations of the Differential Evolution Algorithm (Storn and Price 1997) or Genetic Algorithm (Holland 1975; Goldberg 1989), both of which have been found to give similar and strong performance in traversing such a large search space and robustly finding minimum solutions. A population size of 300 has been used in this research to give a broad initial diversity for the Differential Evolution Algorithm, and then the population is decreased to 30 as the diversity begins to become thinner and local convergence to a solution is to be encouraged. The key solver parameters used are the Differential Evolution parameter $F = 0.8$, and the cross-over ratio parameter CR = 0.5. Starting with a population size that was too small could have issues with stagnation later on in the evolution. An evolution consisting of 500 generations was found to give a good balance between convergence rate and accuracy. A typical Differential Evolution run is shown in Fig. 11, for a population of 300 until generation 250, and then a population of 30 thereafter.

The evolution solver is followed by a function and gradient method to polish the result, using the fittest member of the population with the lowest ϕ as the starting point. This polishing uses an algorithm such as the Sequential Least Squares Quadratic Programming (SLSQP) or L-BFGS-B solvers in SciPy, and using an approximated numerical gradient. It has been observed that a combination of the two optimisation procedures (evolutionary and then followed by a gradient method) is more effective in finding a good value of ϕ than using one in isolation. This is because using exclusively a function and gradient method was sensitive to local minima and the selection of the initial starting point, as randomly selecting a starting point for $\mathbf{q}_\mathrm{i}$ could lead to divergence if this starting point lead to many tensile force densities in $\mathbf{q}_\mathrm{d}$. The sensitivity of the function and gradient methods to falling into local minima has been noted especially for large $\mathbf{q}_\mathrm{i}$ dimension, i.e. when k is large. Conversely, using exclusively an evolutionary method, although robust against falling into local minima, could take significant time to converge to a minimum solution for large k and also large population sizes.

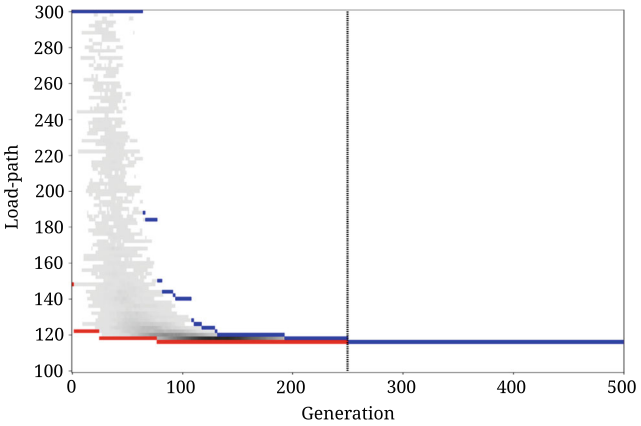

Fig. 11 Typical Differential Evolution result showing the maximum and minimum members of the population in blue and red, and the other members of the population in shades of grey based on the population density at that point. Members of the population that start above the top of the figure with $\phi > 300$ gradually return to the group as the evolution progresses

3.2 Penalisation

Whether using an evolutionary or gradient optimisation solver, there needs to be a mechanism in place to discourage negative force densities in the network to guide a compression-only solution. For the gradient solvers, this is easily enforced by applying an inequality constraint to keep $\mathbf{q} \geq 0$ throughout the solution search. For the evolutionary algorithms, a penalisation is introduced that is a function of the negative force densities $\mathbf{q}_\mathrm{neg}$ contained within $\mathbf{q}$. Different penalty functions were investigated, with the most effective found to be the addition of a continuous penalty function $\phi_\mathrm{pen}(\mathbf{q}_\mathrm{neg})$ to the load path function $\phi(\mathbf{q}_\mathrm{i})$. The function $\phi_\mathrm{pen}(\mathbf{q}_\mathrm{neg})$ distinguishes between members of the population with many large negative force densities and other better quality members with zero or only a few small negative values. The general form of the penalty function is

$$\phi_\mathrm{pen} = \sum(\mathbf{q}_\mathrm{neg} - \alpha)^\beta, \tag{12}$$

which was tested for different values of α and β. It was found that if α was too low, negative force densities were not sufficiently penalised, as a consequence α should be greater than 1 or edges with small tensile values will be made smaller and can hide within the network. So long as the value of $\beta > 2$ and was even, there was little sensitivity to the result and convergence of the final optimal solution, as the edges were sufficiently penalised compared to the value of ϕ. The final choice of penalty parameters in this research was $\phi_\mathrm{pen} = \sum(\mathbf{q}_\mathrm{neg} - 5)^4$.

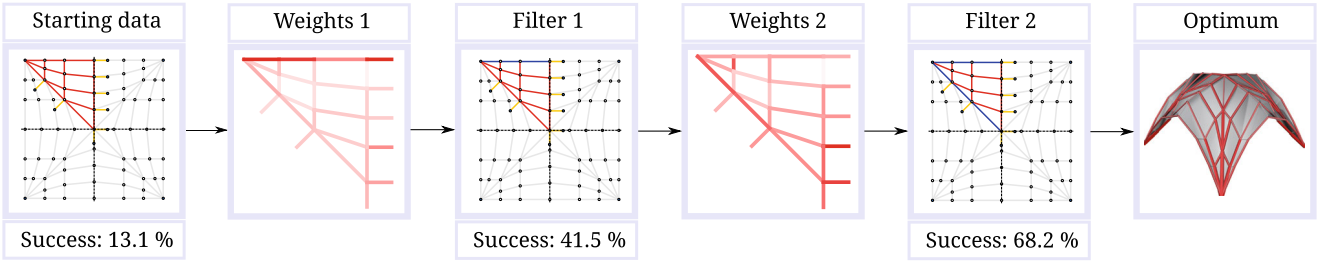

Fig. 12 Data-driven pipeline summary of Section 4, which begins with the starting network data and incrementally improves on trial independent sets before arriving to an optimum set(s) and associated minimum weight thrust network

Alternative penalty functions were also experimented with, for example by limiting the maximum length of the edges during the thrust network calculation step. This was investigated to avoids spikes in the thrust network, which could lead to very high values of ϕ as a consequence of long lengths in $\mathbf{l}$, even if paired with a very small force density in $\mathbf{q}$. These penalty schemes were not as effective as ϕ_{pen} described above, and so was not applied to the final optimisation scheme.

4 Data-driven investigation

The process of finding an independent edge set $\mathbb{M}_{\text{i}}$ by finding the RREF of the equilibrium matrix $\mathbf{E}$, was described in Section 2.2. It was then shown in Section 2.3 that, for a network with low indeterminacy (low value of k), there is still a wealth of different independent edge sets that can be uncovered. The task that this section confronts is to investigate how the choice of $\mathbb{M}_{\text{i}}$ influences the optimised load path results. Given the vast number of independent set combinations that are possible, this is not a task that can be easily answered without methods suited to handling large sets of data and inferring useful results. The use of the described data-driven methods becomes more and more compelling as the value of k increases.

This section introduces a data-driven strategy that not only informs the user of the underlying behaviour patterns of the network but also shows how this information may be used to select sets of independent edges that lead to improved values of the load path ϕ and greater stability in the optimisation process.

The pipeline that will be presented now can be summarised with Fig. 12. In Section 4.1, the network used in this case study is introduced and how key data is generated (labelled as *Starting data* in Fig. 12). In Sections 4.2 and 4.3, two data-driven methods are described, showing how they can be used to very accurately classify the data and produce importance or weight maps. These weights, shown as *Weights 1* and *2* in Fig. 12, can inform the user which edges are most important in determining a good or bad load path value ϕ. Once key important edges are identified, we can filter the independent sets (*Filter 1* and *2*), so that we arrive with better candidate sets of $\mathbb{M}_{\text{i}}$. Having filtered $\mathbb{M}_{\text{i}}$ sets that contain edges, the data-driven methods have identified as important, leads to a better pool of load path results, for which the *Success* of these sets incrementally improves. When there is a good pool of $\mathbb{M}_{\text{i}}$ sets, the best candidate leads to the optimum value of ϕ and hence the minimum weight thrust network (*Optimum*).

4.1 Case study

The network that forms the basis for this data-driven case study is drawn in plan in Fig. 13, composing of 140 edges and loading at vertices based on tributary areas. It has the lines of symmetry exploited by the methods outlined in Section 2.5, which converts the network study to a triangular segment of 29 edges, of which five are independent. This network is thus relatively simple with $k = 5$, yet we shall see that it still shows sensitivity to the selection of $\mathbb{M}_{\text{i}}$.

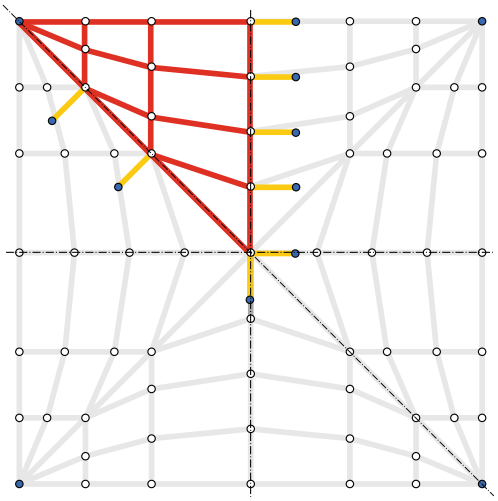

Fig. 13 Network under investigation, with arc patterns spanning between four corner supports. The original 140 edges shown in grey were reduced to the examination of the 29 edges in red, with symmetry applied through the helper symmetry edges plotted in yellow

For this network, batches of 5000 sets ($\mathbb{M}_{\mathrm{i},1}$ through to $\mathbb{M}_{\mathrm{i},5000}$) of independent edges were analysed, with various information extracted from each result for the data-driven learning methods as described next. A binary vector $\mathbf{b}_i$ was created for each trial, of length m and for the set i, to represent the network's $m = 29$ edges. In this vector $\mathbf{b}_i$, there are k elements equal to unity, representing the independent edges that form the set $\mathbb{M}_{\mathrm{i},i}$. For each of the trials, the binary vector element $\mathbf{b}_{i,j}$ is formally given by

$$\mathbf{b}_{i,j} = \begin{cases} 1 \text{ if edge } j \text{ is an independent edge,} \\ 0 \text{ if edge } j \text{ is a dependent edge.} \end{cases}$$

Each binary vector $\mathbf{b}_i$ is then stored in a group of all 5000 trials, as row vectors in the training binary data matrix $\mathbf{X}_\mathrm{b}$, which is of size $(5000 \times m)$. The labels to this data, which are the results relating to the optimised load path values that these trials gave, are stored in a vector $\mathbf{y}$ of length 5000. Each element i of this labels vector represents one trial and takes a binary value for $\mathbf{y}_i$ to show if the load path ϕ was "good" or "bad" according to the rule

$$\mathbf{y}_i = \begin{cases} 0 \text{ if } \phi(\mathbb{M}_{\mathrm{i},i}) \leq 200, \\ 1 \text{ if } \phi(\mathbb{M}_{\mathrm{i},i}) > 200, \end{cases}$$

where $\phi(\mathbb{M}_{\mathrm{i},i})$ is the load path value for the set i of independents $\mathbb{M}_{\mathrm{i},i}$. The observed success of these 5000 trials is calculated as the percentage of good trials where $\mathbf{y} = 0$, i.e. the number of trials where ϕ was less than a threshold of 200. Note that the value of 200 is tied to the structure under examination and will vary for different network topologies, the values and locations of the applied loads and the boundary conditions present, it is not a constant for the optimisation process or solver. For this case study, the value of 200 was found to represent a suitable cutoff point for describing a good solution. This base success percentage is equal to 13.1%.

For each of the 5000 trials, an image was saved where the network was imprinted as pixel data with information about the $\mathbb{M}_\mathrm{i}$ set. These data were stored as $p \times p \times 3$ matrices in a training image data array $\mathbf{X}_\mathrm{i}$ of size $5000 \times p \times p \times 3$, where p is the number of pixels in the two image directions in plan. The red-green-blue colour channels were encoded with red for compression, green for the helper symmetry edges (this was not changed for this study), and finally and most importantly, the edges selected in the $\mathbb{M}_\mathrm{i}$ set activated the blue channel. An example selection of two good and two bad bad sets is shown in Fig. 14, showing how the colour channels can combine to indicate multiple properties: red and green (magenta) show compression and an independent edge, while green and blue (cyan) show a symmetric edge that is also an independent edge.

Qualitatively before introducing the data-driven methods, there are some observations to be made between the two good and two bad $\mathbb{M}_\mathrm{i}$ sets in Fig. 14. One independent edge

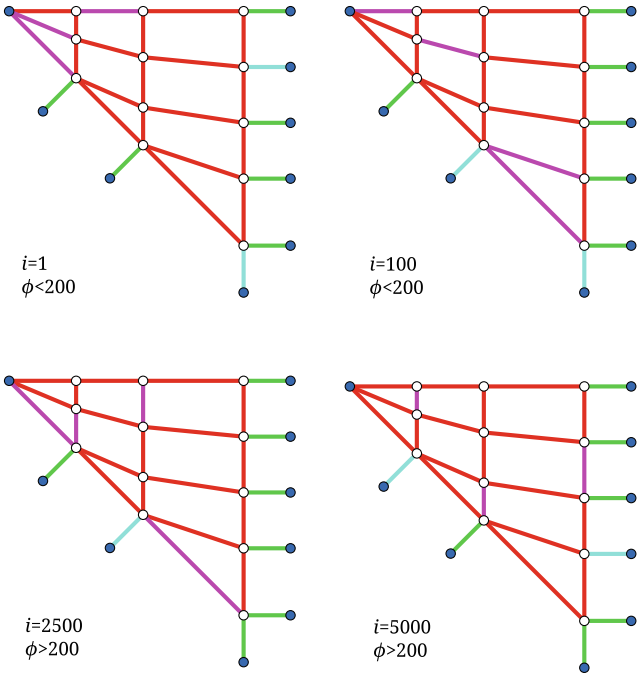

Fig. 14 Example RGB images generated for four example independent sets, the top two lead to a good value of ϕ below 200, while the bottom two led to divergence in the optimisation process

is found at the top boundary of both of the two good sets, and this is not true for the two bad sets. Also, it appears that a good set may contain the bottom symmetry edge, and should not include vertically orientated edges. The latter observation makes structural sense, as this would not control the forces along the arches, which is the main action taking the applied loads from the way the network has been drawn. These manual observations can be made for such a small sample of trials, but robust methods are needed to handle the entire data. For this, two different data-driven methods were tested on the binary $\mathbf{X}_\mathrm{b}$, image $\mathbf{X}_\mathrm{i}$ and label $\mathbf{y}$ data. These methods were a random forest classifier and a neural network, both of which are now described in the following two sections.

4.2 Random forest classifier

The first data-driven method uses a random forest (RF) classifier, which allows for supervised learning based on decision trees. The decision trees created by the RF algorithm randomise the features (the independent edges) to be analysed, which reduces significantly the errors and problems with over-fitting for a large number of trees and extensive data-training (Breiman 2001). The RF approach also allows for a quick assessment on the features' importance, information describing which of the data features were most influential to the final classification (good or bad set).

The random forest classifier used in this research is from the Python machine learning library sklearn (Pedregosa et al. 2011). The meta-parameters of the classifier were chosen after several pilot tests, for which the number of estimators (the number of decision trees) was set to 80, the depth of the trees was limited to 20, and other meta-parameters were kept to their default values. The accuracy of the classifier is controlled through a cross-validation method with degree five, which divides the data randomly into five portions, training the machine with four and then testing against the fifth. The data used for the training is the binary data of 5000 samples ($\mathbf{X}_\mathrm{b}$ and $\mathbf{y}$).

After the training process was complete, the RF classifier was used to make predictions on a new (testing) set of 5000 trials. The classifier receives the new data as before in binary vector format, and using its previous training, decides if the set of independent edges are likely to be a good or bad set. If the prediction is that the set will be good, i.e. that it will lead to $\phi_{\leq}200$, then the set is analysed. The success rate is then calculated as the percentage of testing trials that were predicted correctly. The achieved accuracy of the RF classifier in classification of this new testing data was equal to 97%.

The prediction of whether the $\mathbb{M}_\mathrm{i}$ set is going to be good or not is costless, as it can be calculated with much less computation time than the direct optimisation of ϕ and observing the result, and so avoids the time spent on a calculation that leads to an unsuccessful ϕ.

4.3 Neural network

Two neural network (NN) models were created in TensorFlow (2015) to examine the binary training data $\mathbf{X}_\mathrm{b}$ and the image training data $\mathbf{X}_\mathrm{i}$.

The image data model that worked on $\mathbf{X}_\mathrm{i}$, consisted of the following pipeline. The Python package skimage was used for cropping, scaling and general processing of the image data into a 380×380 pixel format. The processed image data were shuffled into batches of 200 images and fed into the neural network at each step for a total of 1000 steps. Based on the $380 \times 380 \times 3$ RGB image array and batch size, this represents input tensors of shape $200 \times 380 \times 380 \times 3$. The input pixel layer was connected to a layer of 1024 neurons with ReLu activation. This neuron layer was then connected to the two output logits representing the good and bad classification labels. Loss was calculated via softmax cross entropy alongside the AdagradOptimizer. At each step, the weights across the neural network were stored and then processed to display the weights/activations of each pixel which led to each classification.

The neural network model for the binary data $\mathbf{X}_\mathrm{b}$ used the same trials as the RF classifier. This model again composed of shuffling the input data, ReLu activation, two output logits for the binary classification and softmax cross entropy and used the AdagradOptimizer with a learning rate of 0.1. Two hidden layers containing 10 neurons each connected the 29 input features representing the edges of the network. Batches of 500 were used in this model for 2000 steps.

After training the neural network models on the 5000 trials, they were both able to classify new sets of previously unseen (testing) data to an accuracy of 98 to 100%. This means that they were able to correctly identify new independent sets leading to a good or bad value of ϕ. From the results of the RF and NN classifiers, we can see that both have a similar prediction accuracy.

4.4 Weights and importance

From the results of the random forest and neural network classifiers, it is clear that there is an underlying behaviour from the independent sets that will control whether a good ϕ will occur or not, otherwise the classifiers would not have been able to classify the data so accurately. Additionally, the method of classifying the data did not appear to be influential, as the success rates are similar in the range of 97% and 100%, which is a significant increase to the base 13.1% from the raw data.

The random forest classifier can also allow for a quick assessment of the importance of the underlying features during the learning process, via a predefined method that returns an array with the normalised importance of each feature. An insight into the importance factors of the edges allows for the identification of which edges should appear (or not) in the $\mathbb{M}_\mathrm{i}$ set, to preferentially bias the set to give a low value of ϕ. The edge importance factors for the training data $\mathbf{X}_\mathrm{b}$ and $\mathbf{y}$ is plotted in Fig. 15a, with darker shades of red highlighting a greater importance. One can see that the straight edges at the topmost free boundary of the network have been identified very prominently as important for the labelling of a set, as well as some edges near the corner.

From the image training data $\mathbf{X}_\mathrm{i}$, an equivalent plot to the importance factors can be made by showing the weights of the pixels. These weights are shown in Fig. 16 and are similar to the importance factors from the RF, but also inform us if an edge was important for a good (red) or bad (blue) result, not just that it was influential. The same border edges at the top are highlighted, and their weights are similarly identified, with another symmetry edge suggested to omitted for a good solution.

Given that Figs. 15a and 16 show a strong emphasis on the top border edges (the reason for this will be described later), the $\mathbf{X_b}$ data was split into a subset containing the trials where the independent set of edges included at least one of the edges on the boundary. It was then found that

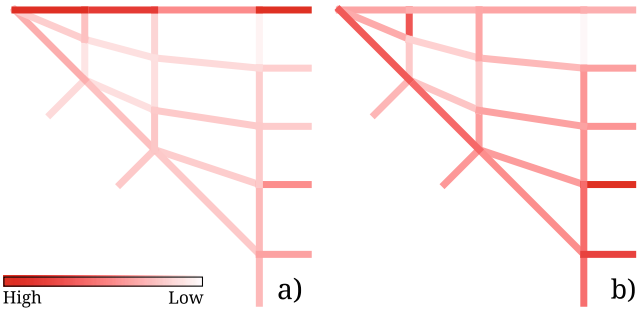

Fig. 15 Plot showing the RF edge importance factors for labelling the $\mathbf{X_b}$ data. In **a**, the first important edges were identified along the top border, while in **b**, the second level of important edges were identified along the main diagonal

the success increased from 13.1 to 41.5%, thus quantifying the importance of activating one of these edges to achieve a lower ϕ. One can continue the classification and importance factors process, but now to assess the next most influential edges given one of the top border edges is pre-selected. This is plotted in Fig. 15b, showing the highest importance factor on the edge located along the main diagonal of the network near the corner support. When this edge is present in $\mathbb{M}_i$ along with an edge at the top border, the success ratio now increases further from 41.5 to 68.2%. This edge can be understood as being structurally important for the definition of the thrusts for the main arches of the thrust network, and so controlling these edges gives stability to the main load-bearing arching paths.

4.5 Interpretation

From the investigation of the load path algorithm stability (or success rate) in connection with the identified importance or weights of the edges, it was found that the straight border edges were influential. This is because for these edges of the compression-only network, the thrusts from the connected edges have no possibility to transmit forces to the free border as their connection is perpendicular. The border edges that were identified as being so critical to the optimisation then generate their own separate arch in the vertical plane, disconnected from the main structure. Therefore, if no independent edges were selected in this straight line, the optimisation process was not effective as it could not control the stability of this arch. Based on this observation, and relaxing the constraint that the border edges need to form a straight line, the altered network presented in Fig. 17 was investigated, where the four straight edges were given a slight arc.

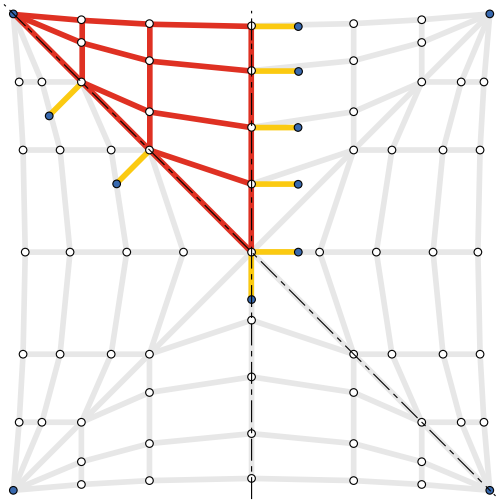

Fig. 17 Adjusted network introducing a slight curved arc at the free border edges. Identical to the original network, the 140 edges plotted in grey were reduced to the 29 edges in red by using the eight helper symmetry edges.

This new network was analysed for 5000 trials to generate new binary $\mathbf{Z_b}$ and image $\mathbf{Z_i}$ training data. The load path optimisation performance is significantly increased by introducing the curvature at the borders, with the observed success percentage rising from the original (straight) 13.1% to the new (curved) 53.2%. When we train the random forest classifier on the data and test the model on new trials,

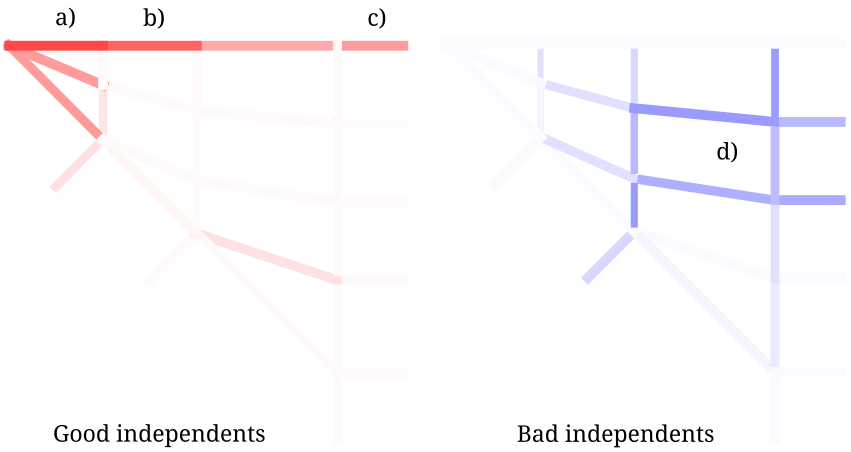

Fig. 16 Plot showing the weights of each edge (converted from pixel data) in obtaining a good (red) or bad (blue) load path for labelling the data $\mathbf{X_i}$. Edges **a**, **b** and **c** were critical in finding a good solution, while edge **d** would lead to poor solutions.

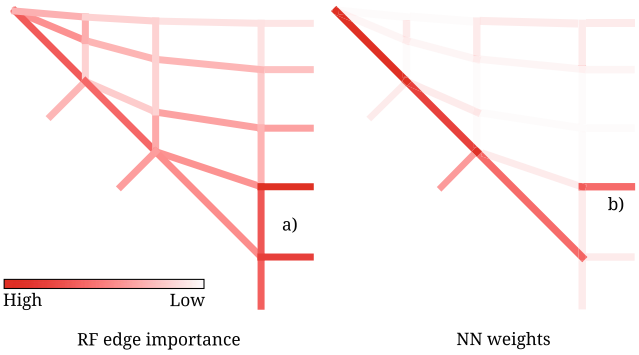

Fig. 18 Plot showing the RF edge importance factors for labelling the $\mathbf{Z_b}$ data for the network with introduced curved border

we obtain a similar classification accuracy as before of 96%. Proceeding with the same analysis of the importance factors, one can study the feature importance of this network. The result for analysing the data $\mathbf{Z_b}$ is presented in Fig. 18, showing a strong influence from the edges on the main diagonal and the edges around the (global) centre of the network at label *a)*. When an edge at *a)* is selected in conjunction with a diagonal edge to appear in the $\mathbb{M}_i$ set, the success percentage further increases to 80.8%. Notice that none of the edges at the now curved border are considered important to have in $\mathbb{M}_i$. The NN edge weights in Fig. 18 show very prominently a preference to including a main diagonal edge into the independent sets, as well as edge *b)*, which was similarly picked up by the RF importance.

5 Conclusions

This paper presented the load path optimisation method for minimising the volume of compression-only thrust networks at given constant stress level. There is a practical construction incentive to optimise such thrust networks, as they can represent material savings and structural efficiency in the design of compression-only structures such as shells by emphasising membrane action and reducing the demand on reinforcement. The research outlined a process to generate a compression-only thrust network and calculate its load path, under given boundary conditions and a network defined in-plan, and then how to minimise the load path by solving a constrained optimisation problem.

The optimisation works by finding optimal force densities for sets of independent network edges, which define the statical indeterminacy pattern of the network, and allows the optimisation to progress more efficiently with fewer degrees of freedom and so a smaller search domain. The number of independent edges to consider can be further reduced by utilising helper symmetry edges, which represent the equal and opposite forces at a symmetry line. A method to find the independent sets of the network was presented through the use of the Reduced Row Echelon form of the network's equilibrium matrix.

There exists such a wide territory of different independent sets that data-driven methods were employed to quantitatively analyse many thousands of cases. Two separate data-driven methods, a random forest classifier and neural networks, gave similar and accurate results when classifying the independent sets. It was shown that with a very high level of accuracy (96–100%), a trained model can reliably predict if an independent set will lead to a good load path value or a divergent optimisation. This showed that there were underlying features within the indeterminacy of the network, which drive good and bad optimisation solutions, and that randomly selecting an independent set is not an effective strategy. It was found that networks can have certain independent edges that have a significant influence on both the stability of the optimisation algorithm, and in the final load path/volume of the structure. Thus it was key to identify which edges held this influence on the optimisation's success.

These embedded features could be brought to the surface by plotting the importance or weight each independent edge had in labelling sets as good or bad. These visual outputs provided insight to infer structural patterns and highlight influential edges in the network. By doing so, it was possible to filter edges that were deemed to be influential, and so drastically increase the quality of the independent sets from 13 to 68%. The importance and weights also highlighted areas that allowed structural judgement and changes to be made, for which a modified network from the base case study was created. The new network broke the identified straight edges forming at the boundary and introduced a structural arch. This further improved the success of the sets from 13 to 53%, eventually leading to a 81% success rate in picking good sets.

Given the reliability and speed of classifying and determining useful information about the weights and importance of each edge, there is great scope in using trained data-driven models to aid the user in finding the best weight optimised network for their problem.

6 Replication of results

All of the source code used in this research is freely available online at github.com through the following open-source respositories: (1) compas-dev/compas for the network datastructure and evolutionary optimisation solvers, (2) BlockResearchGroup/compas_loadpath for the complete load path optimisation algorithm code, and (3) BlockResearchGroup/compas_ml for all machine learning code using TensorFlow. The specific examples used in this

research are found in compas_loadpath/data with files arches_flat.json and arches_curved.json.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

References

Baker WF, Beghini LL, Mazurek A, Carrion J, Beghini A (2013) Maxwell's reciprocal diagrams and discrete Michell frames. Struct Multidiscip Optim 48(2):267–277

Beghini LL, Carrion J, Beghini A, Mazurek A, Baker WF (2014) Structural optimization using graphic statics. Struct Multidiscip Optim 49(3):351–366

Block P, Ochsendorf J (2007) Thrust network analysis: a new methodology for three-dimensional equilibrium. J Int Assoc Shell Spatial Struct 48(3):167–173

Block P (2009) Thrust Network Analysis: Exploring three-dimensional equilibrium. PhD dissertation, Massachusetts Institute of Technology, Cambridge

Block P, Lachauer L (2014) Three-dimensional funicular analysis of masonry vaults. Mechan Res Commun 56:53–60

Block Research Group (2014) ETH Zurich, RhinoVAULT - Designing funicular form with Rhino. [Online] Available at http://block.arch.ethz.ch/brg/tools/rhinovault

Breiman L (2001) Random forests. Mach Learn 45(1):5–32

De Wilde WP (2006) Conceptual design of lightweight structures: the role of morphological indicators and the structural index. High Perform Struct Mater III: WIT Trans Built Environ 85:3–12

Pedregosa et al (2011) Scikit-learn: machine learning in Python. JMLR 12:2825–2830

Python Software Foundation (2017) Python Language Reference. Version 3.6

Fraternali F (2010) A thrust network approach to the equilibrium problem of unreinforced masonry vaults via polyhedral stress functions. Mechan Res Commun 37:198–204

Gilbert M, Tyas A (2003) Layout optimization of large-scale pin-jointed frames. Eng Comput 20(8):1044–1064

Goldberg DE (1989) Genetic algorithms in search, optimization & machine learning, 1st edn. Addison-Wesley Professional, Boston

He L, Gilbert M (2015) Rationalization of trusses generated via layout optimization. Struct Multidiscip Optim 52(4):677–694

Holland JH (1975) Adaptation in natural and artificial systems, 1st edn. The University of Michigan, Ann Arbor

Jiang Y, Zegard T, Baker WF (2018) Form-finding of grid-shells using the ground structure and potential energy methods: a comparative study and assessment. Struct Multidiscip Optim 57:1187–1211

Jones E, Oliphant T, Peterson P et al (2001) SciPy: Open source scientific tools for Python. [Online; accessed 2017-02-13]

Liew A, Pagonakis D, Van Mele T, Block P (2018) Load-path optimisation of funicular networks. Meccanica 53:279–294

Linkwitz K, Schek HJ (1971) Einige Bemerkungen zur Berechnung von vorgespannten Seilnetzkonstruktionen. Ingenieur – Archiv 40:145–158

Lu H, Gilbert M, Tyas A (2018) Theoretically optimal bracing for pre-existing building frames. Struct Multidiscip Optim 58(2):677–686

Marmo F, Rosati L (2017) Reformulation and extension of the thrust network analysis. Comput Struct 182:104–118

Maxwell JC (1864) On reciprocal figures and diagrams of forces. Philos Mag J Ser 4(27):250–261

Mazurek A, Baker WF, Tort C (2011) Geometrical aspects of optimum truss like structures. Struct Multidiscip Optim 43(2):231–242

Michell AGM (1904) The limits of economy of material in frame-structures. Lond Edinb Dublin Philos Mag J Sci 8(47):589–597

O'Dwyer DW (1999) Funicular analysis of masonry vaults. Comput Struct 73:187–197

Pyl L, Sitters CWM, De Wilde WP (2013) Design and optimization of roof trusses using morphological indicators. Adv Eng Softw 62-63:9–19

Schek HJ (1974) The force density method for form finding and computation of general networks. Comput Methods Appl Mech Eng 3:115–134

Storn R, Price K (1997) Differential evolution - A simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 11(4):341–359

Stromberg LL, Beghini A, Baker WF, Paulino GH (2018) Topology optimization for braced frames: combining continuum and beam/column elements. Eng Struct 37:106–124

TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems (2015) Version 1.7.0. Online: https://www.tensorflow.org

Van der Walt S, Colbert C, Varoquaux G (2011) The NumPy array: A structure for efficient numerical computation. Comput Sci Eng 13:22–30

Van Mele T, Block P (2014) Algebraic graph statics. Comput-Aided Des 53:104–116

Van Mele T, Liew A, Mendéz T, Rippmann M et al (2017) COMPAS: A framework for computational research in architecture and structures. [Online; accessed 2017-07-06]

Vandenbergh T, De Wilde WP, Latteur P, Verbeeck B, Ponsaert1 W, Van Steirteghem J (2006) Influence of stiffness constraints on optimal design of trusses using morphological indicators. High Perform Struct Mater III: WIT Trans Built Environ 85:31–40