Computer-Aided Design 160 (2023) 103518

Contents lists available at ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

Structural topology exploration through policy-based generation of equilibrium representations

Ioannis Mirtsopoulos *, Corentin Fivet

Structural Xploration Lab[1], *Ecole polytechnique fédérale de Lausanne (EPFL), Switzerland*

ARTICLE INFO

Article history:
Received 17 November 2021
Received in revised form 18 March 2023
Accepted 23 March 2023

Keywords:
Design space exploration
Generative design
Rule-based design
Topology
Structural design
Strut-and-tie

ABSTRACT

Mainstream approaches to design spatial architectural forms that are structurally relevant consist either in adapting well-known and catalogued conventional types or in searching for close-to-optimum solutions of well-defined problems. Few means exist to explore structural forms detached from these routines.

The approach in this paper generates diverse non-triangulated structural topologies that do not result from optimization procedures. The process incrementally transforms interim networks of bars and forces by means of a parametric policy (–) that maintains the static equilibrium of the network at every single step, (–) that ensures growth of the network within specified (non-)convex geometric boundaries, and (–) whose high-level abstract description controls all design parameters. The successive policy application aims at decreasing the number of interim forces while increasing the number of nodes and bars in compression or tension. The entire process ends when no interim force exists anymore, which is always achievable thanks to the permanence of the static equilibrium condition. From a designer perspective, the approach opens up the generative design black box by providing geometrical and topological control and partial automation of the generation process, while not resorting to common topology patterns – e.g. triangulated bar networks.

This paper describes the conceptualization and its implementation into a computational framework, named Policy-based Exploration of Equilibrium Representations (PEER). It illustrates the potential of the approach to unveil unprecedented, unexpected, but statically-valid, structural topologies. Opportunities for further development are eventually discussed.

1. Introduction

Design is a "wicked problem" that "one cannot first understand, then solve", as stated by Rittel and Webber [1]. Designers compensate for this lack of knowledge through creative processes, one of them being *design exploration*, which frames the incremental generation of design candidates. During this process designers often build on their own experience and browse just a tiny fraction of all possible design candidates. This tendency for premature design fixation typically results in the generation of resembling designs showcasing a lack of diversity and/or creativity [2].

Paulson [3] effectively expressed how early design decisions correlate to higher project costs and hence recommends that the exploration process happens during the early stages of the project lifetime. Others, including [4,5], have also highlighted the urge to explore design at a very early stage.

Current architectural early design workflows are often disconnected from structural evaluation and/or feedback. Mainstream approaches to design spatial architectural forms that are structurally relevant consist either in adapting well-known and catalogued conventional types or in searching for a seemingly optimum solution of well-defined problems. The field of structural forms not resulting from these routines is yet to be explored and this represents an opportunity for developing new tools that would unlock the exploration of new structurally relevant architectural forms.

The computational framework presented in this article, named Policy-based Exploration of Equilibrium Representations (PEER), is meant to be used during the conceptual phase of structural design processes (Section 2.1) while implementing the principles of *Structural Design Space Exploration* (Section 2.2). Rather than rigid predetermined design processes operating on ubiquitous numerical inputs, PEER follows an incremental policy-based

* Corresponding author.
E-mail address: ioannis@mirtsopoulos.xyz (I. Mirtsopoulos).
[1] http://sxl.epfl.ch

https://doi.org/10.1016/j.cad.2023.103518

approach (Sections 2.3 and 2.4), which is inspired by the fields of shape grammars and grammar rules (Section 2.5).

2. State-of-the-art

2.1. Conceptual structural design

In the scope of this article, conceptual structural design stands for the process starting from the blank page and leading to schematic, early-stage options of static equilibrium arrangements, or more precisely, to synthetic representations of a structure's static equilibrium by means of bars in compression or tension, as was the case for e.g. Robert Maillart's design of the Salginatobel bridge in 1928 [6]. Bar networks express equilibrium representations. Their layout is developed for a chosen predominant load case but their materialization into effective structural forms, be it reticulated or continuum, during subsequent design steps is meant to resist all expected load cases. Throughout history, the radical abstraction offered by models of bar networks has proven to be highly useful and versatile for the exploration of unconventional masonry [7], reinforced concrete [8], steel [9] and timber [10] structures. The generated networks are indicative, but not deemed optimized, force flows, to be used as first inspirations, prior to comprehensive structural analyses and further form refinements.

2.2. Structural design space exploration

Mueller [5] emphasizes that conceptual design helps designers to find high quality design alternatives through the Design Space Exploration (DSE) process. The hypothesis is that DSE unveils unprecedented typologies, dodges premature design fixation, provokes creativity and facilitates decision making. Aiming at workflows that provide structural guidance within the creative process implies the rejection of non-structural design variants and subsequently the shrinkage of the solution space.

Nowadays, DSE is inseparable from computerized workflows that facilitate and accelerate the process. Different approaches exist: parametric, computational, or generative [11]. Typically, generative workflows depend on two main kinds of solvers: deterministic algorithms, when convergence is key, and metaheuristics, when exploration is key. The former, given a particular input, always result in the same output. The latter instead yield near-optimal results very quickly. As search algorithms, they are a perfect fit for ill-structured problems [12], characterized by non-rational features, the absence of global optimality and contradicting solution paths, like design processes.

Research projects on design space exploration have focused on generating and collecting diverse sets of optimum and near-optimum solutions [5,13–15], which are likely to be discarded by conventional algorithms. The aim of these projects is to allow designers to contribute to the decision-making process, while simultaneously considering optimization objectives and other objectives that are harder to express in a mathematical way. From a designer perspective, topology is usually predefined and an evolutionary exploration mechanism controls the nodal coordinates. Explorative features may operate differently: (a) tentative design solutions are stored in a database from which they can be retrieved at any moment [13]; (b) interactive genetic algorithms are exploited to ensure the designer's active participation and the inclusion of non-qualitative criteria – e.g. aesthetics – during the search [5,16]. Of particular interest, Harding [16] contributed to design exploration mechanisms after recognizing that further interface development is required to display multiple models simultaneously, with the release of *Biomorpher*; a Rhinoceros Grasshopper plug-in that allows parametric definitions to be explored using interactive genetic algorithms (IGA) [17].

Recently, Ohlbrock et al. [18] introduced the use of a topological graph, that visualizes nodal relationships, as a manipulative design input. The approach eases design exploration through the generation of variant topological configurations using Combinatorial Equilibrium Modelling (CEM). Besides, Harding [19,20] introduced the concept of meta-parametric design, which combines graph-based parametric modelling with genetic programming. This bottom-up design approach allows the generation of variant directed acyclic graphs (DAG) that represent variant parametric models and subsequently variant architectural forms.

Alternative approaches for the exploration of the vast design space involve the incremental use of design-related policies, as described in the next two sub-sections.

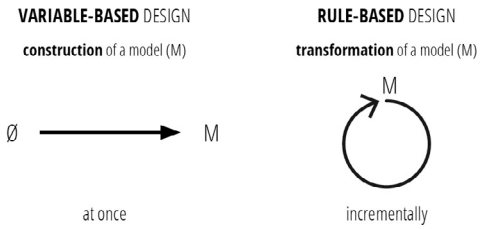

Fig. 1. Predetermined versus incremental design.

2.3. Predetermined versus incremental design

The introduction of parametric workflows has recently revolutionized the generation of architectural forms. Parametric workflows ease the instant computation, or construction, of new design variants following the alteration of a set of input parameters. However, the inter-relationships between these parameters are usually predetermined and left unaltered during the design process. Consequently, new design variants require the establishment of new inter-relationships – i.e. parametric models – a process laborious and time consuming. Moreover, the model only exists after the generation process is completed [20].

Contrary to predetermined design processes, incremental design processes transform models incrementally (Fig. 1). Incremental design starts with a valid but incomplete model and transforms it at any moment that a variation is desired and for as long as it is desired. This is an open and segmented process: while the process can be fully automated, it lets the designer interrupt, redirect, or move a step backward, at every step of the generation process. In other words, designers can provide input, interact with the output and continuously influence the design process of the model refinement itself. Incremental approaches are therefore closer to intuitive modelling processes—e.g. sculpting, painting or architecture project design. Transformative actions are applied successively, each based on an updated understanding of the problem being searched.

2.4. Numerical versus policy-based inputs

In parametric design, the inputs are typically described by a numerical set: current value, and upper and lower bounds. Their definition is often predefined in an arbitrary way constraining the design space exploration to a fraction of it. Examples of numerical inputs in parametric design are shown in Fig. 2.

On the contrary, policy-based inputs express design intentions in a qualitative, descriptive, more abstract way. Their definition includes notions of comparison between before-after states—e.g. when they describe the impact of the intentions—or currently

Fig. 2. Numerical versus policy-based inputs.

available options—e.g. when they rank values (largest, furthest etc.). Examples similar to policy-based inputs are shown in Fig. 2. Policy-based design works well with incremental design, where the model constantly changes and descriptions of action are described through policy intentions. Their combination is the basis for this paper and is believed to favour efficient design space exploration.

2.5. Shape grammars and grammar rules

Stiny and Gips [21] introduced the term of shape grammars for design, inspired by Noam Chomsky's theories on generative grammars in language: *"[Chomsky's] idea was that a grammar had a limited number of rules that could generate an unlimited number of different things, and the resulting language was the set of things the rules produced"*. Thinking of design as the resulting language, a set of design rules assembled in various configurations can generate infinite design variants. Stiny and Mitchell [22] used shape grammars to generate a wide range of plans that follow design principles found in Palladian Villas. Shea [23,24] applied shape grammars to the synthesis of triangulated trusses, whose main topological features were predefined. Simulated annealing was exploited on complete bar networks to obtain optimal structures. Geyer [25] applied grammar rules at a component level for the design of buildings. O'Neil [26] combined grammar rules with interactive evolutionary algorithms for the design of a shelter using linear beam elements. Beirão [27] created urban-specific grammars for the generation of urban patterns. Chakrabarti [28] reviewed the application of graph grammars, an abstract generalization of shape grammars, for design synthesis. Through grammar rules, Byrne et al. [29] have shown that Grammatical Evolution is capable of creating surprising and innovative designs. Zimmermann et al. [30] integrated a spatial grammar for wheel spoke design in a workflow that concludes with the consideration of additively manufacturing the designed products. Tomei et al. [31] proposed shape grammars for the topology optimization of grid shells and diagrid tall buildings. Mueller [32] applied structural grammars both randomly and manually to generate diverse sets of structural systems. More recent implementations have coupled vector-based graphic statics with grammar rules [33] and have proven their suitability to generate structurally-aware forms during the conceptual, early design stage.

Grammars and rules in most of the above cases are context specific. However, they always generate an unlimited number of deliverables, whatever the syntax. In terms of design, this strength of rules is recognized as a way to revolutionize and redefine the term *parametric design*. Their value in the field of structural conceptual design is showcased in this paper, where rules are coupled with structural awareness in order to explore the design space in a semi-automated way.

3. Overview

3.1. Intentions and scope

Stemming from the above statements, this paper presents PEER; a new incremental, design framework to generate network topologies in static equilibrium (Fig. 3). The (structural) topologies are not necessarily triangulated and do not result from optimization procedures. The framework improves state-of-the-art alternatives in the following ways:

- it breaks down the generation process to the node-by-node network creation;
- it maintains static equilibrium of the network at every intermediate step of the generation process;
- it ensures growth of the network within specified (non-) convex geometric boundaries;
- it transforms networks by means of a policy that is defined by high-level abstract rules.

Following DSE principles, PEER allows designers to generate fast design variants that satisfy specific criteria – i.e. static equilibrium and bounding volume – while handling design decisions by means of policies that affect the overall topology of the resulting structure. When handled manually, the incremental generation allows the designer to consider arising design challenges before they percolate in the final solution. When automated, the incremental generation allows the designer to process routines that can be stopped or cancelled at any step of the generation. PEER does not assume knowledge of structural mechanics by the designers. Particularly, the designer is free to make sequences of decisions among the provided options, without questioning static equilibrium. All generated design candidates satisfy static equilibrium. The offered freedom could be compared to free drawing of structural systems, which implies a lot of background knowledge, it is cumbersome and therefore generation of numerous variations is not always possible.

The PEER framework is not meant to be a toolkit for numerical analysis of structures once a structural form is known. Rather, it is intended to be used beforehand, for the generation of spatial forms in static equilibrium. The process is said to be semi-automated because the designer decides when and for how long the generation is automated, at any time of the process. Topology is neither predefined nor bound to existing structural typologies. Instead, topology is defined during the decision making process and is the main output of the exploratory process.

3.2. Conceptualization

The design process consists in transitioning from a disconnected network of force vectors in interim equilibrium (Fig. 3, top left) to a connected network of bars and force vectors in global static equilibrium (Fig. 3, top right). The starting point is a set of forces (applied loads and support reactions) applied onto a bounded territory, called *design domain* ($\mathcal{D}_\mathrm{d}$). The set of forces together with their nodes of application – i.e. anchor points – form a disconnected network of force vectors. At this stage, the disconnected network is guaranteed to be in interim static equilibrium thanks to the introduction of equal and opposite interim forces applied at the same anchor points.

The goal of subsequent step-wise transformations is the incremental elimination of interim forces in the network through the addition of bars and other interim forces. Each transformation – e.g. Fig. 4 – consists of: the introduction of a new node P; the elimination of selected interim forces; the creation of new bars in compression (struts) or tension (ties); and if necessary, the introduction of new interim forces that retain static equilibrium.

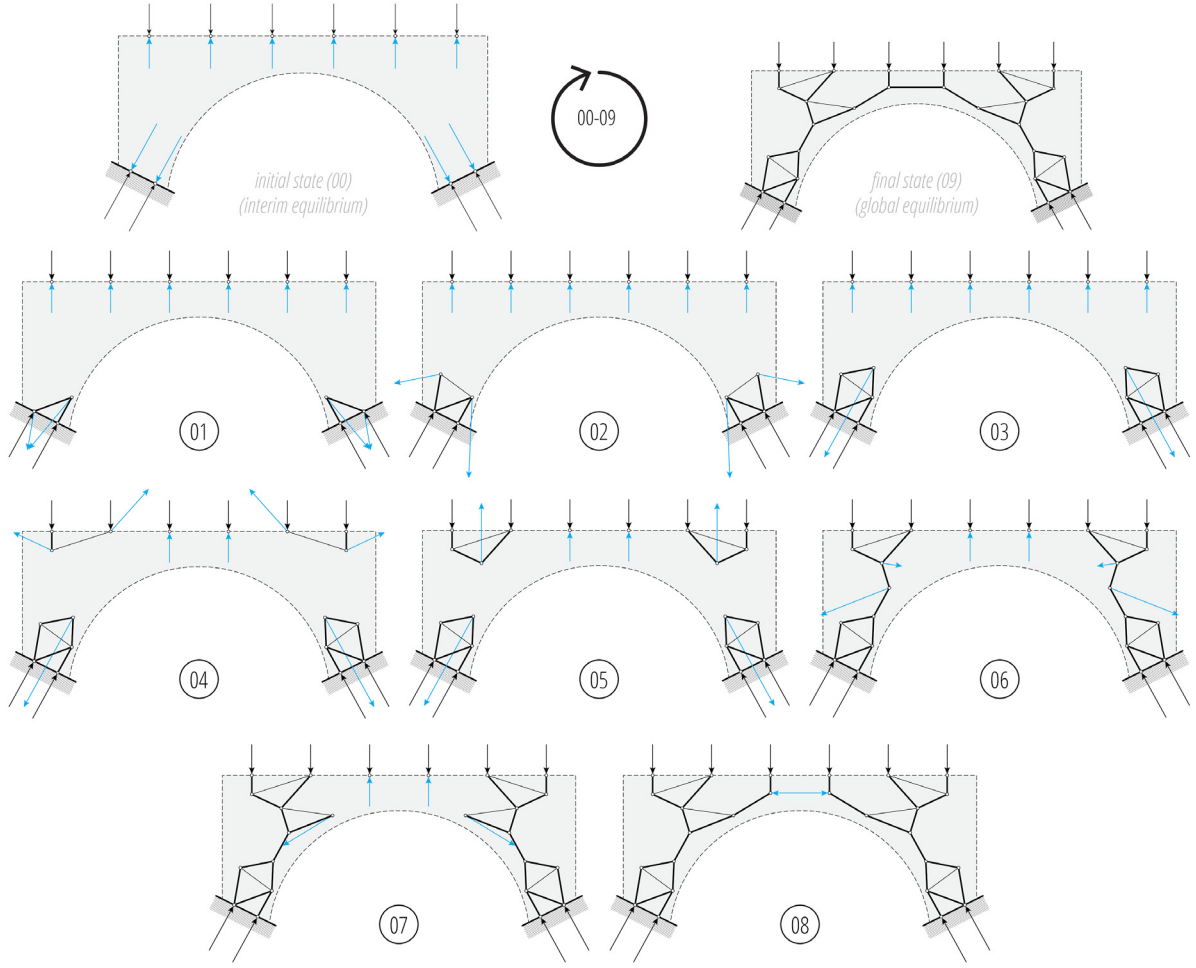

Fig. 3. Incremental transformation steps for the generation of an arch bridge-like network of bars in static equilibrium in the plane. Applied forces in black and interim forces in blue. Thick bars in compression and thin bars in tension. *Design domain* ($\mathcal{D}_d$) in light grey. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

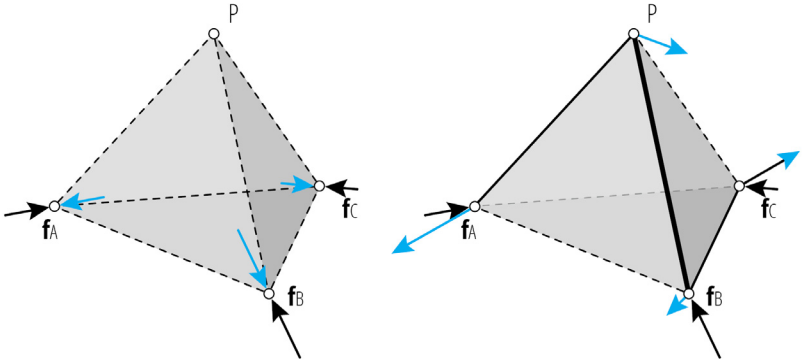

Fig. 4. Before/After transformation in space. Black arrows represent force actions that are applied externally to the system or by other compression or tension bars. Blue arrows represent interim forces. Bars in compression are thick lines. Bars in tension are thin lines. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

By construction, the process excludes design variants that are not in static equilibrium or that are not contained within the design domain, hence only exploring a structurally-relevant design space. Other non-quantitative or subjective requirements – e.g. aesthetics, spatial functionality, cultural values – are expected to be visually assessed by the designer at any desired moment of the generation process. The generation process concludes as soon as the pool of interim forces is empty.

Each transformation step is controlled by a chosen policy whose definition is independent of the current, intermediate state of the network. Hence, the same policy can be applied to any type of network, irrespective of its topology, geometry, or number of remaining interim forces. Each policy is defined by four rules. The first one defines the starting point of the growth—i.e. the choice of interim forces to eliminate. The second one defines the speed of the growth—i.e. the number of interim forces to add or delete.

Both of these rules affect the topology of the growth. The third one defines the geometry of the growth—i.e. the position of the new node P. The fourth one defines the developed forces along the bars—i.e. the type (compression/tension) and magnitude of the axial forces. These rules are valid whatever the current state of any bar network to transform.

The transformation step is further described in Section 4 and its use in the generation process is given in Section 5. Applications follow in Section 7.

4. Transformation step

4.1. Genericity

The entire generative process of bar networks builds on one generic transformation step. Its unicity and syntax follow the intention to allow both universality and speed of application. The chosen strategy lies between two dismissed extremes: either to provide a set of primitive transformations—e.g. divide a force, get the resultant of two forces, split a bar—or to provide a set of high-level transformations—e.g. "add triangulated modules", "support with columns". The former extreme has the benefit of avoiding redundancies among the transformation steps, but lengthens the sequence of needed operations to manage. The latter extreme has the benefit of addressing more directly the designer's formal intention but is prone to become self-constraining and to remain typology-specific, generating predefined forms that lack diversity and/or creativity. In between these two extremes, the proposed transformation step is meant to have a level of abstraction that is low enough to explore the design space, and high enough to influence the generations by means of policy-described geometry and topology features.

4.2. Behaviour

Each chosen transformation step (Fig. 4) operates on a chosen set of three interim forces ($\mathbf{f}_A$, $\mathbf{f}_B$, $\mathbf{f}_C$), and creates a new point P, new bars and new interim forces. The three initial interim forces can be coincident with each other, hence leading to monomial, binomial, or trinomial input forces. These forces are selected from the pool of all interim forces, either explicitly or by means of a higher-level, abstract *force selection rule*. Examples of force selection rules are: "select the forces whose anchor points are the closest", "select the forces with the oldest age in the model", "select the forces whose anchor points are, on average, the closest to existing bars"; "select the forces whose magnitudes are the closest to each other"; or "select the forces whose orientations form the widest angle".

Up to three bars are added. This limitation relates to the fact that up to three interim forces are considered per policy transformation, which seems contradictory to a real design challenge. Nevertheless, three bars is the minimum transformation that a trinomial can undergo. Keeping the number of added bars minimum is a matter of speed, clarity and increased control. In particular, the fewer interim forces are transformed at once, the faster (from a computational point of view) the transformations are, the more gradual (from a visual point of view) the transformations are, and the higher the designer's control over the process is. Furthermore, this limitation retains simplicity and constrains complexity. Simplicity is a key feature that adds value and increases the popularity of any design workflow, multiples the chances of getting established in practice and charms designers and software developers. Low complexity is beneficial for the development and future enhancement of any computational frame-work. Each bar corresponds to one of the 6 edges of a fictitious tetrahedron constructed through four vertices: vertex P and the three existing anchor points A, B, and C. Different bar topological configurations are hence available, as described on Fig. 5. Additional interim forces are applied on P, A, B, and C. Their orientation and magnitude are computed in order to guarantee static equilibrium of the four points after introducing P and the bars. When multiple equilibrium states exist – i.e. when more bar forces or interim forces exist than what can be deduced from equilibrium equations – additional force magnitudes in the bars should be provided by means of a *force indeterminacies rule*.

The preferred position of P in space constitutes a free input that is also either explicitly defined by coordinates or controlled

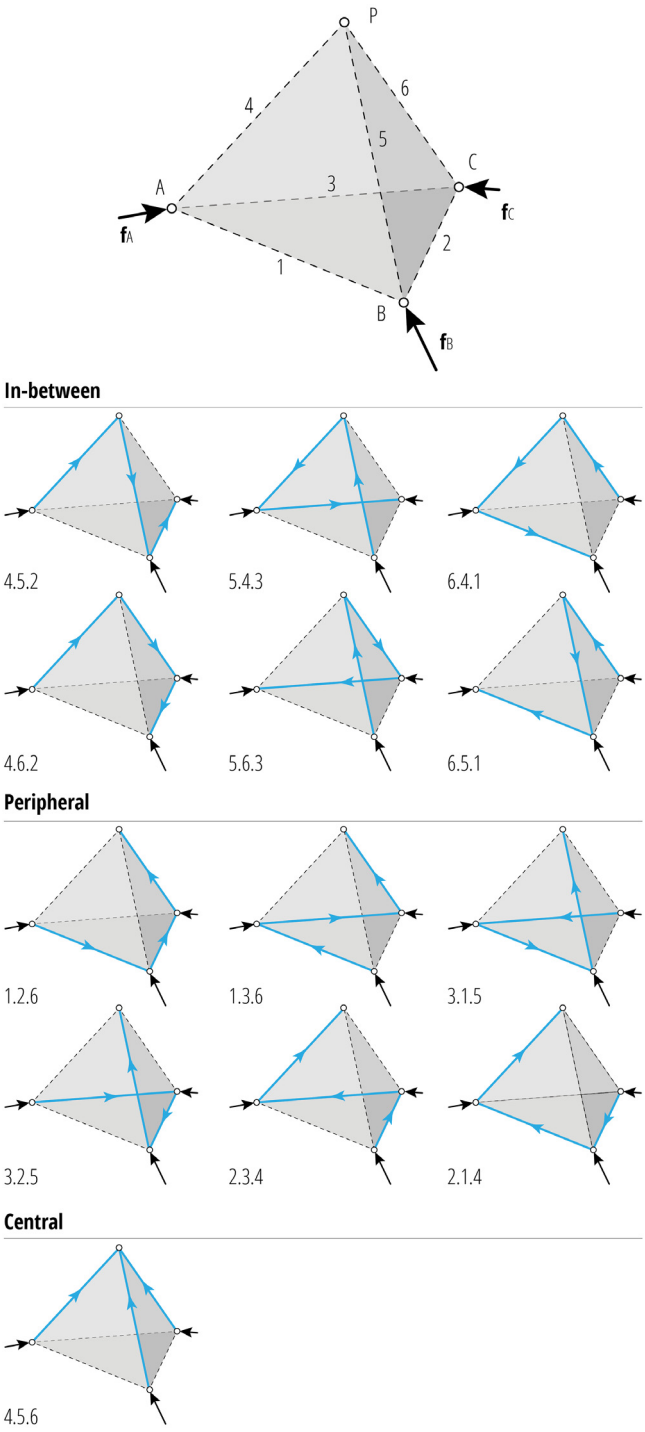

Fig. 5. Possible topological configurations of a trinomial.

by means of a higher-level, abstract *node placement rule*. Examples of such rules are: "choose the position of P that leads to the smallest sum of bar lengths"; "choose the position of P that is the closest to the geometric centre of all nodes in the system"; "choose any random position of P that is not further than a length l from all force anchors"; or "choose any random position of P that lies on a given axis". However, the actual position of P is constrained within a subset of the design domain ($\mathcal{D}_\mathrm{d}$), that is named *entropy rate domain* ($\mathcal{D}_\mathrm{f}$). The $\mathcal{D}_\mathrm{f}$ size depends on an additional input parameter named *entropy rate*. The P placement into the respective to the entropy rate domain ensures static equilibrium of the model.

The *entropy rate* of a transformation step is defined by the difference between the number of interim forces present in the interim network before and after the transformation. A negative, null or positive, *entropy rate* hence leads to the *convergence*, *stagnation* or *divergence* of the generation process, respectively. Since the *entropy rate* has a direct impact on the speed of the generation process, and more importantly on the eventual topological complexity of the system – i.e. on the number of bars and nodes in the completed network – it stands as an explicit input that, along with the set of selected interim forces and the topological configuration of bars, constrains the actual position of P (Fig. 8).

4.3. Inputs and outputs

In summary, each transformation step is defined by a policy that consists of a set of exactly four input choices:

- the choice between a negative (*convergence*), null (*stagnation*), or positive (*divergence*) *entropy rate*;
- a preferred set of 1 to 3 interim forces, defined either explicitly or by means of a *force selection rule*; if the selected forces are provided explicitly, an explicit definition of the intended topological configuration of the bars to create should be provided; otherwise the topological configuration is embedded into the rule;
- the preferred position of a new point P, defined either explicitly or by means of a *node placement rule*;
- if needed, the force magnitudes in some of the bars by means of a *force indeterminacies rule*.

Outputs consist of:

- the effective position of P;
- the effective selection of interim forces to remove;
- up to three newly-created bars connecting P, A, B, and C, and their force magnitudes;
- up to four new interim forces applied on P, A, B, and C, one per anchor point;

It is worth noting that the number of inputs is finite, although the transformation process is not bounded to any finite set of topologies. Parameterizing every transformation down to a finite set of shared parameters, first allows the generation of diverse topologies with one generic transformation step, without the need to handle typological exceptions, second eases its eventual automation.

Moreover, each *force selection rule*, *node placement rule* or *force indeterminacies rule* is chosen among a predefined discrete set of options. They inform the transformation in a qualitative and descriptive way, at a higher level of abstraction. This abstraction efficiently disseminates the decision making in discrete branches and makes the entire process intuitive for the designer. Concretely, the application of a chosen rule is similar to the execution of a function.

5. Generation process

PEER builds on the successive application of transformation steps as illustrated on Fig. 7 which consists of six stages. The first stage sets up the process (*initiate model*) and is only executed once. The remaining stages—i.e. *select force(s) and topology*, *place new node*, *set indeterminacies*, *add interim force(s)*, and *update model*—are repeated until the model is complete—i.e. if no interim force remains in the network.

The first stage depends on permanent inputs only. The last four stages depend on transient inputs that are subject to change following new design intentions that may emerge during the process. The set of transient inputs define the transformation policy.

The next subsections each describe one of the six stages in the process.

5.1. Initiate model

All generated bar networks are expected to be circumscribed within geometric boundaries defined at the beginning of the process as permanent inputs. These volumes and the included voids, if any, form the *design domain* ($\mathcal{D}_\mathrm{d}$) and are generated beforehand based on the designer's initial constraints.

A bar network, completed or in progress, is called *model*. The initial model comprises the *design domain* and initial interim forces. Initial interim forces are opposite to the forces provided by the designer—i.e. to applied loads and support reactions. When given external loads have no support reactions, reactions are added in order to satisfy global rotational and translational equilibrium of the model.

Externally indeterminate configurations of applied loads and support reactions are dealt with as an equivalent determinate case where additional input parameters define the magnitude and orientation of chosen applied or reaction forces—i.e. considering indeterminacies as design inputs (see Section 5.4).

5.2. Select interim force(s) and bar topology

The first stage in the process loop consists of selecting interim forces to eliminate and the topology of the bars to introduce. The selection is operated either explicitly or by means of a chosen *force selection rule*.

If explicit, the selection needs two designer inputs: choice of interim forces from the pool of all interim forces, using their index; and choice of topological configuration – i.e. *in-between*, *peripheral*, and *central*, see Fig. 5 – defining the topology of the newly created bars. The set of selected forces defines a *monomial*, *binomial* or *trinomial*. This set of choices is then checked for feasibility—i.e. whether it complies with the desired *entropy rate* and required connectivities. As shown on Fig. 6, not all combinations of topological configuration, number of interim forces, and *entropy rate* are feasible. Similarly, the presence of voids or non-convex design domains not always allow the desired nodal connectivity. If the set of inputs is deemed non-feasible, a new one is requested from the designer.

If operated through high-level rules, all sets of 1, 2 or 3 forces in the pool of all current interim forces are computed as potential sets of interim forces. The list of feasible sets of interim forces is then sorted according to the given *force selection rule*. The list is then travelled and the first feasible set of interim forces is selected—i.e. the first set that complies with the desired *entropy rate*. If no feasible set of interim forces is found, the *entropy rate* is then increased – e.g. from *convergence* to *stagnation* or from *stagnation* to *divergence* – which increases the number of needed transformation steps and ultimately the number of bars and nodes in the final model. When no set of interim forces is feasible with a diverging entropy rate, the designer gets informed and the generation process is terminated.

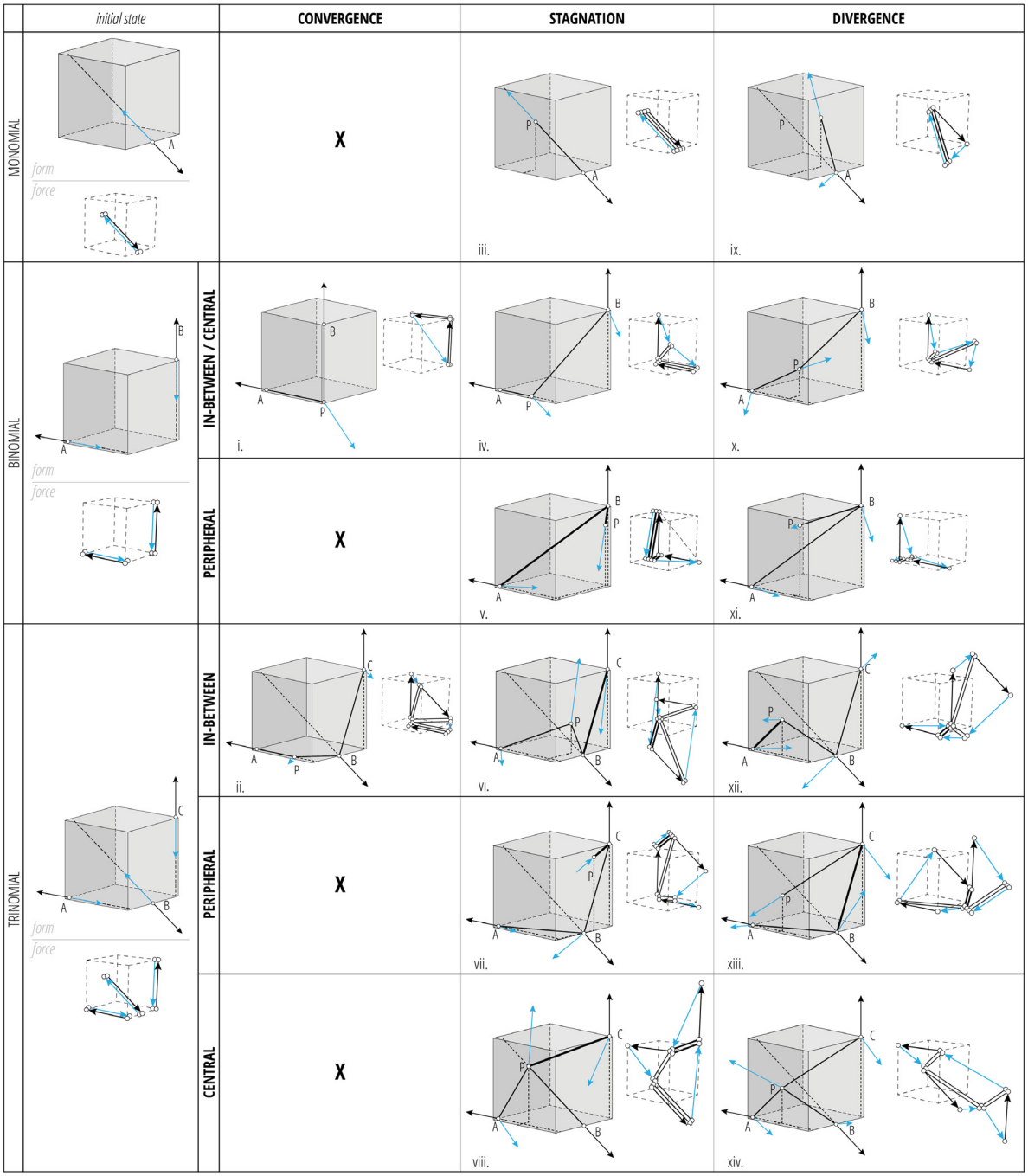

Fig. 6. Transformation syntax in space; black arrows are applied loads or support reactions, blue arrows are interim forces; thick black bars are in compression (struts); thin black bars are in tension (ties). The force diagrams demonstrate static equilibrium. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

5.3. Place new node

The second stage in the loop consists of defining the position of the new node P in the design domain. Again, its location is defined either explicitly or through a chosen *node placement rule*. In both cases, P must lie within a feasibility domain ($\mathcal{D}_\mathrm{f}$). Section 6 provides the mechanisms to compute $\mathcal{D}_\mathrm{f}$ for each entropy rate, topological configuration and number of interim forces. For example, if *convergence* is desired, $\mathcal{D}_\mathrm{f}$ is reduced to a single position. But when *stagnation* and *divergence* are allowed, the feasible domain for P consists of more than one position (see Section 6.1). If a proposed position is not within $\mathcal{D}_\mathrm{f}$, its closest projection to $\mathcal{D}_\mathrm{f}$ is computed and chosen for P.

5.4. Set indeterminacies

Whenever the effective entropy rate is 0—i.e. *stagnation*—or 1—i.e. *divergence*—, and despite knowing the exact location of the new node, the set of unknown forces in the new sub-system of bars and interim forces is greater than what can be deduced by solving static equilibrium equations. In other words, a number of additional design freedoms, expressed as force magnitudes in new bars, is available to the designer. Following a similar fashion as to the selection of interim forces (see Section 5.2) and the node placement (see Section 5.3), *force indeterminacies* can either be provided explicitly or through a higher-level *force indeterminacies rule*.

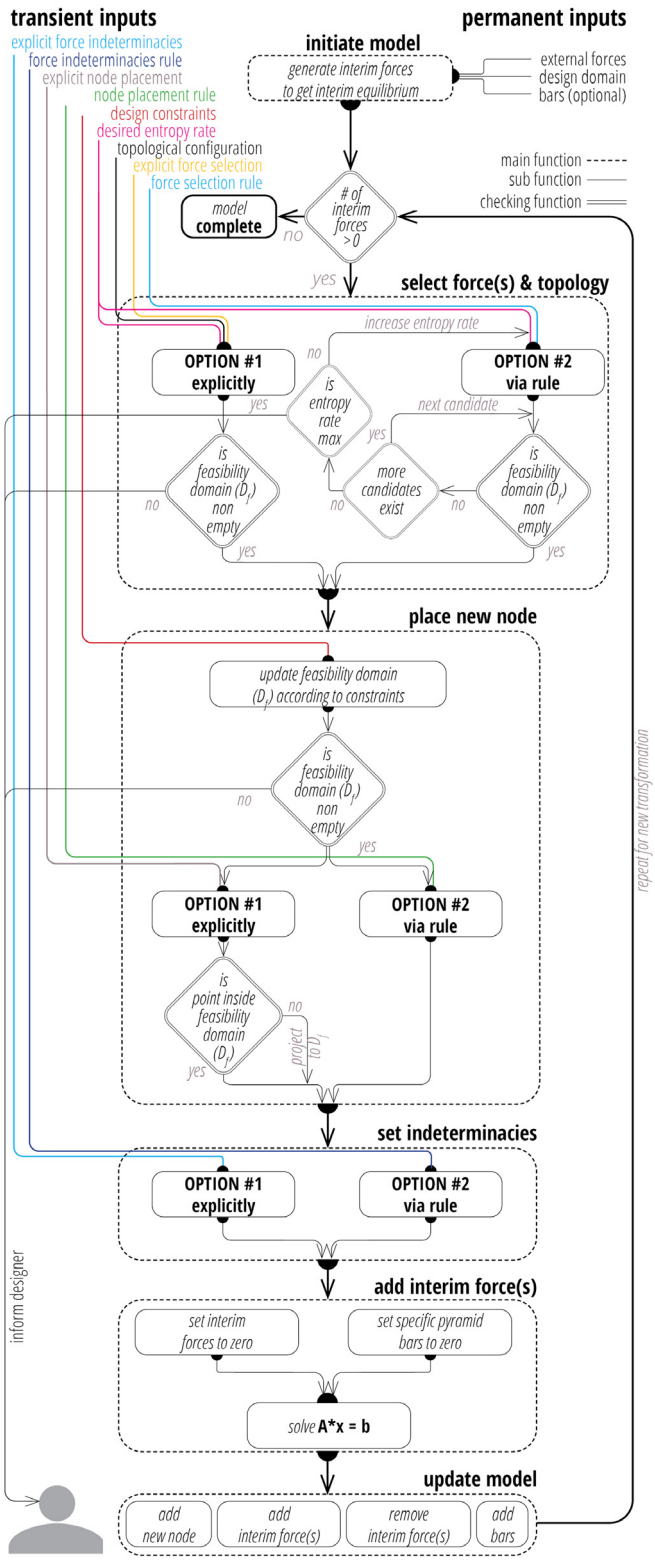

Fig. 7. Algorithmic workflow for applying successive policy-based transformations.

5.5. Add interim forces

This next stage consists in finding the value of the remaining unknown interim forces and bar force magnitudes, while guaranteeing static equilibrium. The initial model is recursively updated and grows larger node-by-node by transforming sub-networks of maximum three nodes and by adding an additional fourth node. All four nodes and potential new bars form a tetrahedron sub-network. Because static equilibrium is checked for every new single node, computing translational equilibrium only is sufficient and ensures satisfaction of rotational equilibrium. Considering that translational equilibrium of a point in space is described by three equations, one for each axis x, y and z, the translational equilibrium of the sub-network being transformed is ruled by 12 equations. The system to solve is of the form $\mathbf{A} \times \mathbf{x} = \mathbf{b}$ where:

- $\mathbf{A}$ is a 12×18 matrix describing the topology and geometry of the tetrahedron;
- $\mathbf{x}$ is a 18×1 vector containing the x, y and z magnitudes of forces in the bars and of interim forces; some of them are pre-defined through the *force indeterminacies rule*, while the rest are the solutions of the system;
- $\mathbf{b}$ is a 12×1 vector containing the x, y and z magnitudes of the external forces.

More precisely:

$$\left[\mathbf{I} \;\middle|\; \begin{matrix} \Delta_{BA} & \mathbf{0} & \Delta_{CA} & \Delta_{PA} & \mathbf{0} & \mathbf{0} \\ \Delta_{AB} & \Delta_{CB} & \mathbf{0} & \mathbf{0} & \Delta_{PB} & \mathbf{0} \\ \mathbf{0} & \Delta_{BC} & \Delta_{AC} & \mathbf{0} & \mathbf{0} & \Delta_{PC} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \Delta_{AP} & \Delta_{BP} & \Delta_{CP} \end{matrix}\right] \begin{bmatrix} \mathbf{t}_A \\ \mathbf{t}_B \\ \mathbf{t}_C \\ \mathbf{t}_P \\ n_1 \\ n_2 \\ n_3 \\ n_4 \\ n_5 \\ n_6 \end{bmatrix} = \begin{bmatrix} \mathbf{f}_A \\ \mathbf{f}_B \\ \mathbf{f}_C \\ \mathbf{0} \end{bmatrix}$$

where $\mathbf{I}$ is the 12×12 identity matrix, $\mathbf{0}$ is the 3×1 null vector, n_k ($k = 1, 2, \ldots, 6$) are the force magnitudes in the six bars of the tetrahedron, as shown on Fig. 4, and for any point $i = \begin{bmatrix} x_i & y_i & z_i \end{bmatrix}^\mathrm{T}$ or $j = \begin{bmatrix} x_j & y_j & z_j \end{bmatrix}^\mathrm{T}$ equal to A, B, C, or P—i.e. the four vertices of the tetrahedron on Fig. 4:

- Δ_{ij} is the $\begin{bmatrix} \frac{x_i - x_j}{l_k} & \frac{y_i - y_j}{l_k} & \frac{z_i - z_j}{l_k} \end{bmatrix}^\mathrm{T}$ difference vector where points i and j are both ends of bar k and l_k is the length of bar k;
- $\mathbf{t}_i$ is the $\begin{bmatrix} t_{i,x} & t_{i,y} & t_{i,z} \end{bmatrix}^\mathrm{T}$ interim force applied at point i;
- $\mathbf{f}_i$ is the $\begin{bmatrix} f_{i,x} & f_{i,y} & f_{i,z} \end{bmatrix}^\mathrm{T}$ external force applied at point i;

Regardless of the network size, the process is identical. The equilibrium equations only consider the sub-network and thus computation time for each transformation is unrelated to the number of earlier or later steps of the transformative process.

Of all the possible bars in the base tetrahedron (Fig. 5), only a few actually exist once the *entropy rate* and the *topological configuration* are set (see Fig. 4). Moreover, the force magnitude in some new bars is given through the *force indeterminacies rule* (see Section 5.4). Hence, the matrix description can be simplified. Precisely, for each edge k of the tetrahedron where no bar should be created, depending on the selected topological configuration, n_k is set to zero. For each edge k whose inner force magnitude is set during stage 4 (see Section 5.4), n_k is set to the given force magnitude. And for each node i where no interim force is introduced, $\mathbf{t}_i$ is set to the 3×1 null vector. These actions significantly reduce the problem size and make its solving direct. The unknown new interim forces $\mathbf{t}_i$ and bar force magnitudes n_k are obtained after inversion of matrix $\mathbf{A}$.

5.6. Update model

The last stage consists in updating the model—i.e. removing the selected interim forces, adding the potential new ones, adding

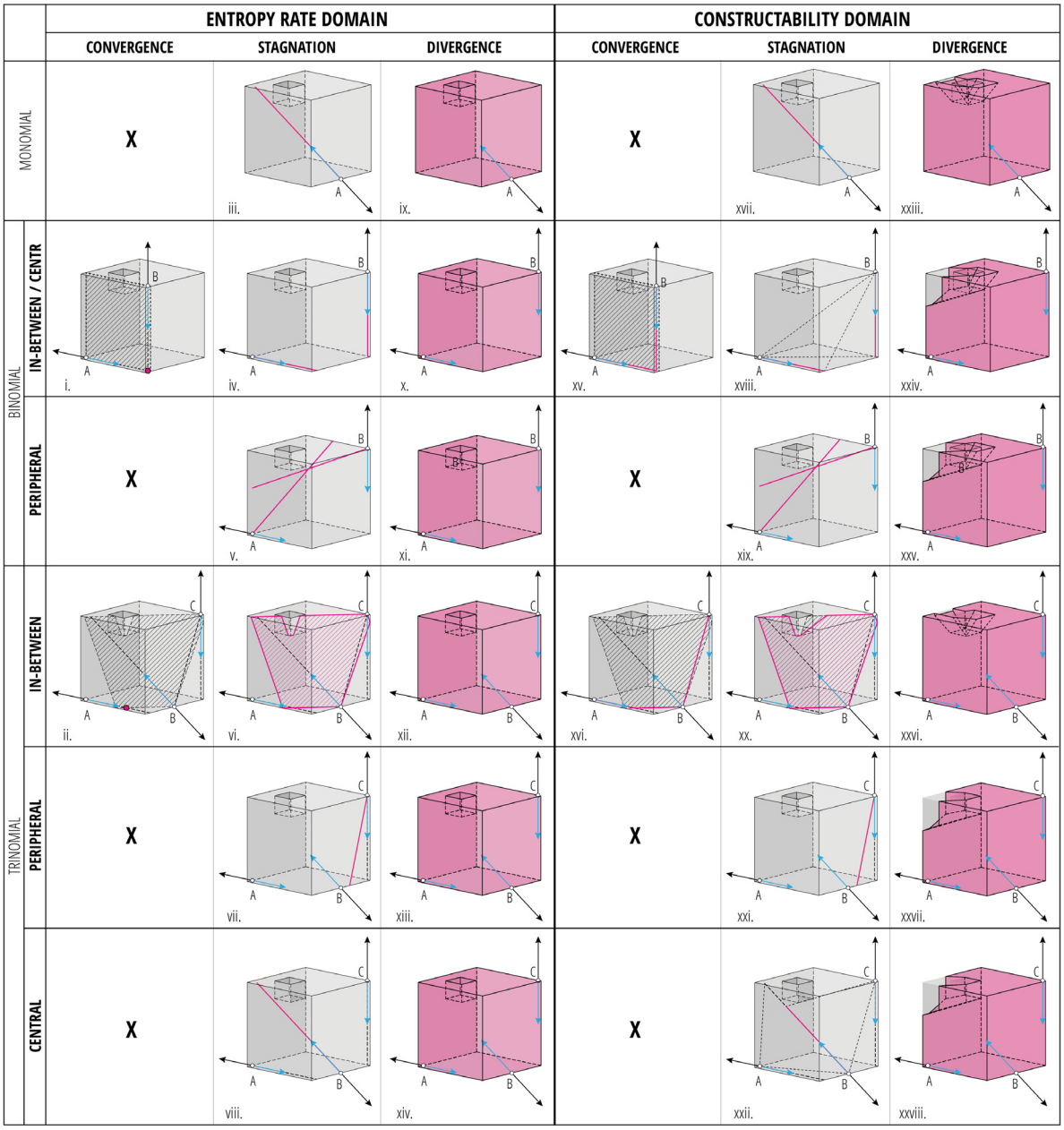

Fig. 8. Typologies of entropy rate and constructability domains. Black arrows are applied loads or reactions, blue arrows are interim forces, all circumscribed within a primitive design domain (light grey). Magenta regions on the left are the intersections of the design domain (a non-convex solid) with the entropy rate domain. Magenta regions on the right are the intersections of the design domain with the entropy rate domain and the constructability domain. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

the new node P, and the new bars. Once the model is updated, a check on the number of remaining interim forces in the model is made. If that number is greater than zero, a new transformation step is performed, either with the same policy as previously, or with new inputs from the designer.

6. Feasibility of the transformation step

Because point P and the bars created after a transformation step must all lie within an input *design domain* or must fulfil auxiliary constraints, not every choice of entropy rate, interim forces to eliminate, and placement of point P is feasible. For instance, a negative entropy rate (*convergence*) on two interim forces whose lines of action meet in a point outside the design domain is not feasible. Neither is the creation of a bar that crosses a void in the design domain. Nor is the creation of a bar that has a length longer than a custom threshold—if auxiliary constraints are set.

These various requirements are here all ensured by requesting each new point P to lie within a *feasibility domain* ($\mathcal{D}_\mathrm{f}$). In other words, the $\mathcal{D}_\mathrm{f}$ is the region that contains all feasible positions of P. An empty $\mathcal{D}_\mathrm{f}$ means that the choice of entropy rate and/or interim forces must be reevaluated. The $\mathcal{D}_\mathrm{f}$ is defined as the boolean intersection of the *design domain* ($\mathcal{D}_\mathrm{d}$), an *entropy rate domain* ($\mathcal{D}_\mathrm{e}$), a *constructability domain* ($\mathcal{D}_\mathrm{c}$), and a *auxiliary domain* ($\mathcal{D}_\mathrm{a}$). Each domain and its respective boolean operations are precisely calculated at least once at stages two (Section 5.2) and three (Section 5.3) of each transformation step. Their calculation is assisted by the integrated geometry library McNeel Rhinoceros 3D offers. Out of the four input choices described in Section 4.3, the *entropy*

rate, the *force selection rule* and the *node placement rule* directly affect the extents of $\mathcal{D}_f$, whereas the *force indeterminacies rule* has no impact on it.

6.1. Entropy rate domain - $\mathcal{D}_e$

The *entropy rate domain* is the set of positions for P that ensure the feasibility of a chosen entropy rate applied on selected interim forces. Its construction therefore depends on the entropy rate, the number of interim forces and the topological configuration of the new bars, see Fig. 8 left.

A negative entropy rate (*convergence*) is only feasible when point P forms an *in-between* configuration with the bars (Fig. 5), in which case the *entropy rate domain* is a single position. When two interim forces are selected (*binomial*, Fig. 8i), the domain is the point of intersection of the lines of action of the two forces. When three forces are selected (*trinomial*, Fig. 8ii), the domain is the intersection point between the line of action of $\vec{\mathbf{f}}_A$ and the plane common to $\vec{\mathbf{f}}_B$ and $\vec{BC}$ — i.e. the segment joining anchor points B and C, see Fig. 5 for nomenclature of vectors.

A null entropy rate (*stagnation*) leads to the *entropy rate domain* being a set of lines or a plane. When a monomial is provided (Fig. 8iii), the domain is the line of action of the force. When a binomial is provided with an *in-between* configuration (Fig. 8iv), the domain is the union of the lines of action of both interim forces. When a binomial is provided with a *central* or *peripheral* configuration (Fig. 8v), the domain is the union of the lines of action of $\vec{AB} + \vec{\mathbf{f}}_A$ and of $\vec{AB} - \vec{\mathbf{f}}_B$. When a trinomial is provided with a *in-between* configuration (Fig. 8vi), the domain is a plane defined by $\vec{\mathbf{f}}_B$ and $\vec{BC}$. When a trinomial is provided with a *peripheral* configuration (Fig. 8vii), the domain is the line of action of $\vec{BC} - \vec{\mathbf{f}}_C$. When a trinomial is provided with a *central* configuration (Fig. 8viii), the domain is equal to the line of action of $\vec{\mathbf{f}}_B$.

A positive entropy rate (*divergence*) always leads to the *entropy rate domain* being the full design domain (Fig. 8ix to xiv).

6.2. Constructability domain - $\mathcal{D}_c$

The *constructability domain* ensures that every single newly created bar stands within the *design domain*—i.e. does not intersect its boundaries. This domain is of particular need when the *design domain* is non-convex, with or without voids. In other words, the *constructability domain* is the set of positions that are visible by the points—i.e. A, B, or C—that connect to P with a new bar. The *constructability domain* therefore also depends on the entropy rate, the number of interim forces and the topological configuration of the new bars, see Fig. 8 right.

6.3. Auxiliary domain - $\mathcal{D}_a$

The *auxiliary domain* ensures supplementary temporary or permanent constraints defined by the designer. For instance, it may constrain the length of the bars within a predefined range $[l_{min}; l_{max}]$, where l_{min} may relate to construction costs since longer bars decrease the amount of nodal connections, and where l_{max} may relate to buckling limits [34]. Geometrically speaking, the *auxiliary domain* in this example is the intersection of two hollowed spheres, with inner and outer radii equal to l_{min} and l_{max} respectively (Fig. 9). It therefore only depends on the choice of interim forces.

The resulting $\mathcal{D}_f$ is computable as soon as an entropy rate and a set of interim forces are selected. Its emptiness informs that either another entropy rate or another set of interim forces must be selected, although neither the *node placement rule*, nor the static equilibrium equations are processed already. Moreover, a visualization of $\mathcal{D}_f$ informs the designer about the available positions for point P, if chosen explicitly.

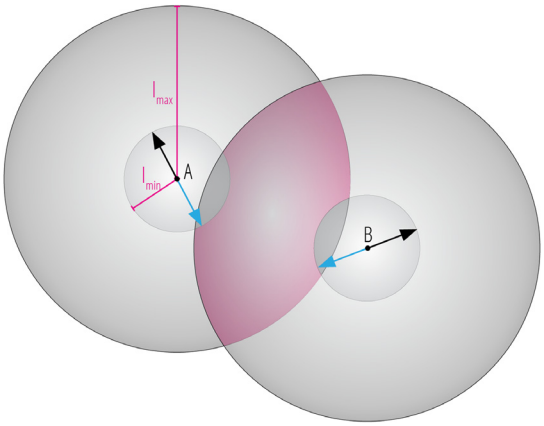

Fig. 9. Example of a domain constraining bar lengths—the magenta region is a planar slice into the spatial auxiliary domain.

7. Application studies

Shape grammar implementations have the potential to both automate parts of the design process and allow exploration of design alternatives [35]. The same is true for the PEER framework. PEER has been implemented into a parametric tool named *Libra* [36], a plugin operating in Rhinoceros' 3D Grasshopper platform. Libra has been used in the following case studies which show how variations in the input parameters allow the generation of diverse, unprecedented conceptual structural design variants in static equilibrium. Four types of bar networks are presented, three in 2D—a pylon, a cantilever, and an arch-like bar network—and one in 3D—a bridge-like bar network. For those studies that more than one design solutions are illustrated, a spider graph that integrates useful metrics is provided. In particular, starting from the top and continuing clock-wise the measured feature include: maximum (axial) force magnitude, number of intersecting bars, maximum bar length, number of nodes, minimum (axial) force magnitude, number of bars, minimum bar length and static action. The inclusion of such a graph facilitates the comparison among design variants based on geometric and performance related metrics.

7.1. Choice of rules

The four case studies make use of a limited set of rules: 11 force selection rules (implicit and explicit), two node placement rules (implicit and explicit), two force indeterminacies rules (implicit and explicit), and three entropy rates—see Table 1. As discussed in Section 5.2, the force selection rules contain information on the number of interim forces to select, how to select them from the pool of all interim forces, and what topological configuration to choose when connecting the anchor points with new bars. In the 2D case studies, the *node placement rule* and the *force indeterminacies rule* describe an explicit choice, while in the 3D case study they describe a random (implicit) choice.

7.2. Step-by-step generation in the plane

The first case study is one of a symmetric pylon, Fig. 10. The design domain presents cavities and acute angles that make it non-convex. Applied policies consist of a variation of force selection rules 1, 2, 3, 4 and 5, and node placement as well as force indeterminacies that are always used-defined, Table A.3. The desired entropy rate is never chosen to be diverging and has always been applied without being adjusted—i.e. the applied entropy rate

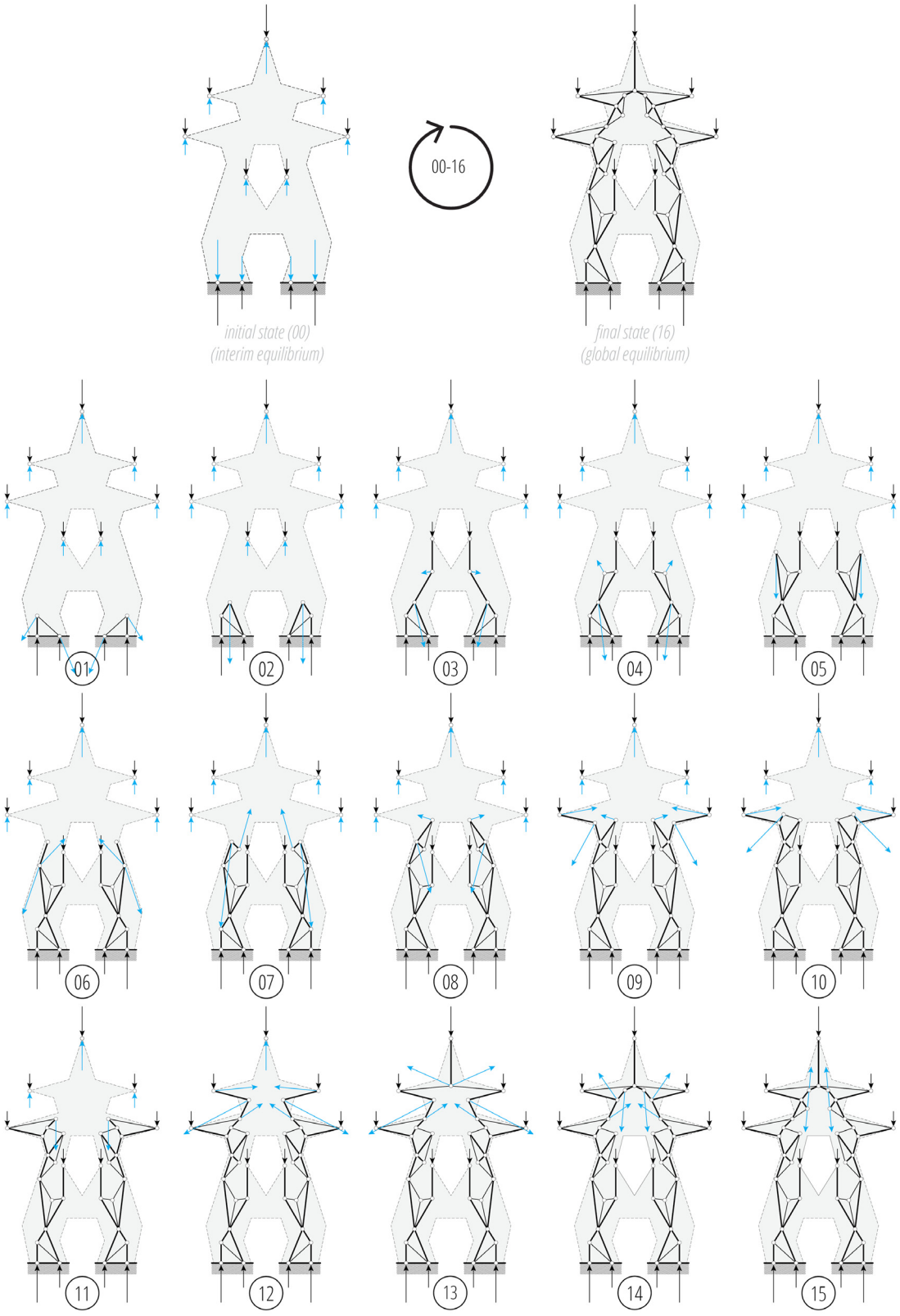

Fig. 10. Symmetric pylon-like bar network.

is always identical to the desired one. For the symmetric result, after completing the left part, the network is mirrored on the other side of the symmetry plane.

The definition of each policy on a manual (non-automated) step-by-step basis allows the generation of a controlled topology without any bar intersections. A different sequence of policies composed of different rules or randomness seeds could have resulted in an intersecting network. This is a concerned that predominates in planar examples. In 3D studies, intersecting bars are expected to be very rare.

Table 1
Set of rules used in the application studies.

Force selection rules	
0	Custom set of forces/binomial/in-between config.
1	Oldest/binomial/in-between config.
2	Newest/monomial/in-between config.
3	Newest/binomial/in-between config.
4	Oldest-newest/binomial/in-between config.
5	Closest/binomial/in-between config.
6	Furthest/binomial/in-between config.
7	Furthest-to-domain's-centroid/binomial/in-between config.
8	Minimum magnitude/binomial/in-between config.
9	Proximity to user-defined point/monomial/in-between config.
10	Proximity to user-defined point/binomial/in-between config.
11	Proximity to user-defined point/trinomial/in-between config.
12	Closest-to-domain's-centroid/trinomial/in-between config.
Node placement rules	
a	Random with seed number
b	User defined t-param
Force indeterminacies rules	
A	Random with seed number
B	User defined force indeterminacies
Entropy rates	
−1	Convergence
0	Stagnation
1	Divergence

Local narrowness of the design domain greatly constrains the geometry and topology of the flow of forces and subsequently the introduction of the necessary bars. However, as the rule application acts blindly the process continues unaffected until completion, which happens after 16 steps. The 16th step replaces the opposite and equal forces with a bar in tension.

The resulting bar network mixes triangular and non-triangular bar layouts. Though the bar network may be judged as unstable if it were to be built as is, it does not constitute an issue *per se* because the bar network is only a preliminary, conceptual arrangement of abstract force flows—e.g. additional bars may still be added later on, connections may be made rigid, or the bars may eventually represent stress fields in a continuum of material. Static equilibrium is guaranteed. The same remarks apply to all following application studies.

7.3. Step-by-step generation in space

The second case study concerns a spatial network in a design domain similar to a bridge, Fig. 11. Non-planar interim binomial forces cannot be the subject of an entropy rate that leads to convergence. Therefore, the use of trinomials is necessary when working in space. Checking the rule application feasibility is more time consuming in 3D than in 2D, due to the boolean operations applied to construct the feasibility domain ($\mathcal{D}_f$). However, computing time to solve the rule and update the model is not affected by the third dimension.

The choice of rules is made on a manual (non-automated) step-by-step basis, Table A.4. The first design objective is to bring the interim forces from the supports up to the deck level, using force selection rules 11 or 12. After, all interim forces are coplanar and the transition to a connected network is straight forward, using force selection rules 0 or 10. Selecting *binomials* constrains the exploration strictly on that plane.

The developed design workflow gets even more powerful when undesired transformations are undone to be replaced by designer-approved ones. This flexibility is an important feature of any design process, always present during design exploration. Fig. 3 details an example of an incremental transition from an incomplete network with interim forces into a complete bar network in static equilibrium. Transformations are constrained inside a non-convex arch-like design domain. The policy applied throughout the generation process is described on Table A.2.

7.4. Comparison of generations

Fig. 12 displays four cantilevering trusses, all resulting from the same problem consisting of one force applied vertically to two given reaction supports. The inputs choice is intentionally different in every design option in order to achieve diversification. The design domain is convex and eases fast convergence. The processes are described in Tables A.5 to A.7. Additional restrictions on minimum and maximum bar lengths are applied through the *auxiliary domain*. The number of transformations is highly dependent of the desired bar length bounds.

The tree cantilevers are geometric and topological variations of the same family of structures. Key metrics arranged in a spider graph are provided for each design outputs and allow their quantitative comparison. Static action [37] is measured as the sum of the products between length and force magnitude of each bar. Assuming that the material is the same in every bar and presents a similar strength in compression and tension, static action is directly proportional to the amount of material needed to manufacture the bars. As such, it is practical for early performance assessment of conceptual designs.

For every design problem, the number of design results is infinite. The exploratory power of PEER is demonstrated on Fig. 13, where the same design problem has been tackled by multiple designers. The non-convex design domain adds complexity to the generative process and many designer intentions, at transformation level, cannot be completed. Subsequently, selection of different input parameters result in completely different design options. All applied policies are given in Tables A.9 to A.24.

8. Discussion

The method itself is straight-forward and clear regarding the minimum number of iterations required until the network gets connected. The number of interim forces decreases by one when the design space converges once. The designer has direct control ("on demand") over the convergence (or divergence) and freely chooses the number of iterations the process converges, stagnates or diverges for. Consequently, the minimum total number of transformations is known all along the iterative process, provided that the desired entropy rate is always applied.

Similar design methods (eifForm [38], ParaGen [15], StructureFIT [32]) follow top-down approaches and operate on predefined topology which offers more design control but limits the breadth of exploration. Moreover, no structural analysis (e.g. FE solver) is required to check/confirm static equilibrium or additional post-processing (i.e. dynamic relaxation) to impose it. If the current transformation, or following ones, does not satisfy the designer's qualitative criteria, backtracking to previous stages is allowed.

Each transformative policy is a high-level description of a generic tetrahedron of bars and interim forces, added to the existing model. The rules are independent of specific structural typologies and unaware of the designer's intentions, but structural awareness is embedded into their syntax. Contrary to [23] that manipulates strictly triangular shapes, static equilibrium does not require triangulation and thus the design space can be explored thoroughly. This disconnection allows designers to escape from catalogues of structural systems or optimization procedures and frames a new, infinite space of structural topologies, where new structural forms are to be discovered.

With regards to computing performance, the largest portion of computation time belongs to the boolean operations required for

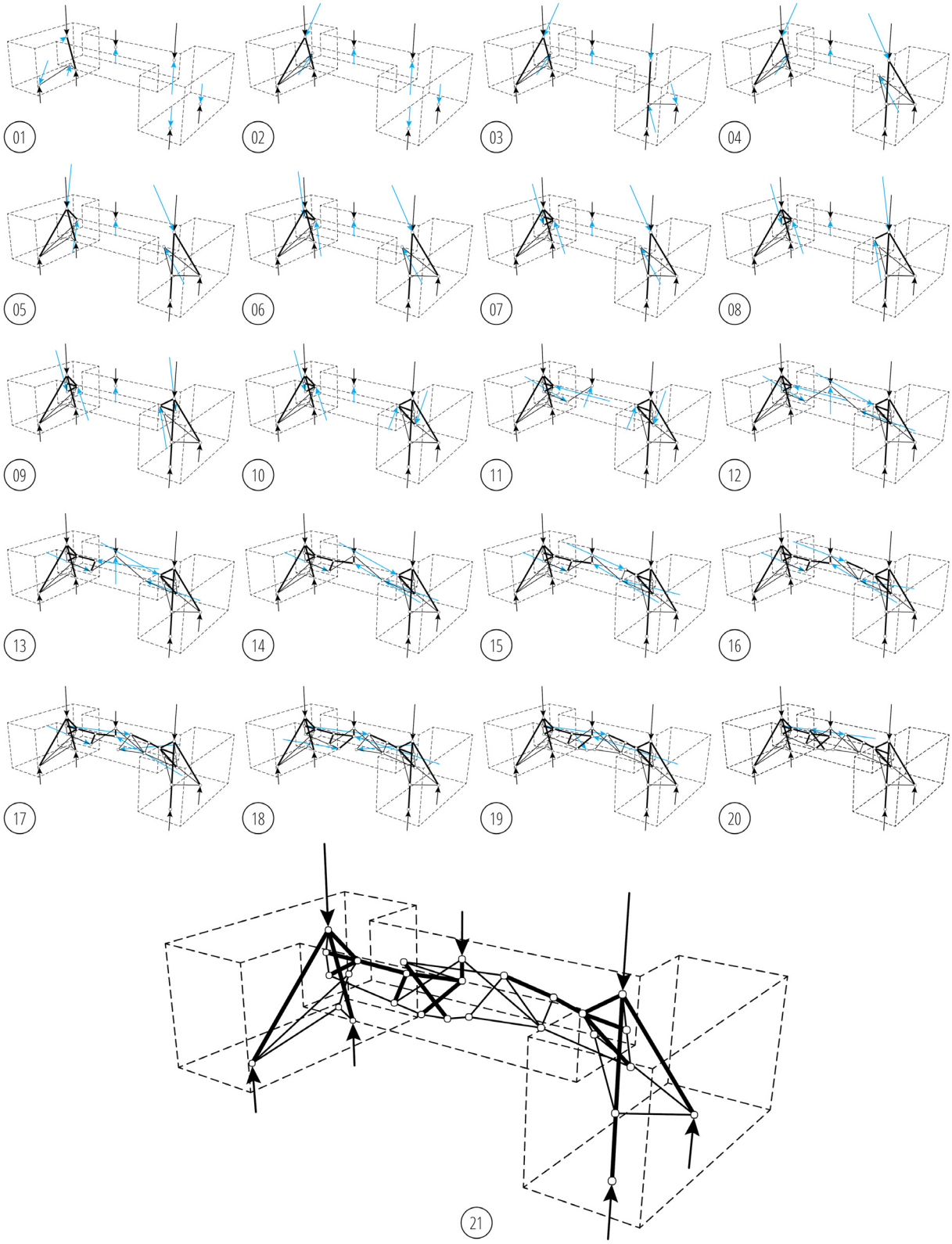

Fig. 11. Spatial bridge-like network of bars in compression and tension.

the calculation of the entropy rate domain, the constructability domain and the feasibility domain. Obviously these calculations are more time consuming among three-dimensional domains. Consequently, the computing time for a fully automated transition from an incomplete network to a complete one ranges between a few milliseconds (for two dimensional convex design domains) to a couple of seconds (for three-dimensional non-convex ones). In particular, the (almost) fully automated example included in this paper (Fig. 13xvi), is generated in 35 ms (steps 2–5). The time to apply a transformation step is a couple of ms,

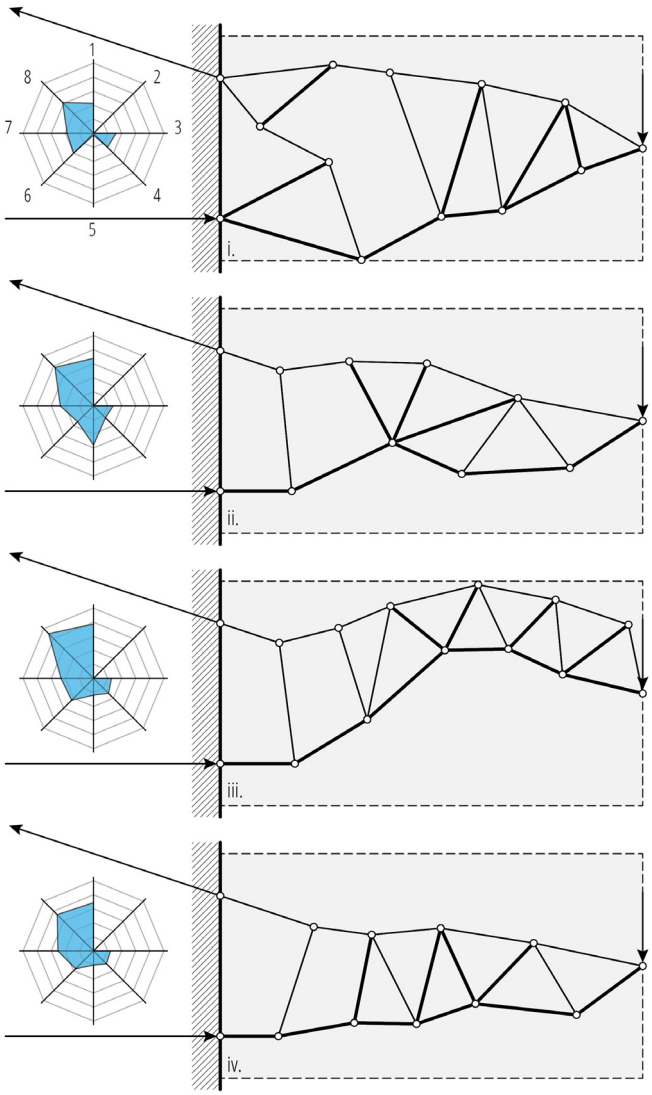

Fig. 12. Cantilevering bar networks and key metrics recorded on a spider graph (1: maximum force magnitude, 2: number of bar intersections, 3: maximum bar length, 4: number of nodes, 5: minimum force magnitude, 6: number of bars, 7: minimum bar length, 8: static action).

because PEER is independent of the number of performed transformations. At every step, the rule is applied onto the selected set of forces which forms a sub-network in interim equilibrium. Subsequently, the algorithmic complexity related to the application of the rule is polynomial, as it only requires solving the same matrix, that always has the same size, regardless of the network size.

Although the policy application is unbiased from numerical variables, the logic brought to the design process from the human side with the notion of rules prevents or imposes specific transformations. Decision making, genuinely human, manipulates qualitative criteria and accelerates the convergence process, but it is tedious and cumbersome. Fully automated policy transformation accelerates the design process. However, the machine alone cannot efficiently apply rules in an aesthetics-approved manner. The lack of critical thinking, usually results in networks which satisfy all the set constraints (static equilibrium, bar length bounds, clearance distance etc.), but do not widely spread across the design domain making them introvert and dense topology wise—i.e. multiple nodes populated close to each other.

The role of the designer during the entire process is highly impactful. The combination of input choices can accelerate/decelerate or even terminate the process when no transformation is feasible. Currently, the designer is expected to think ahead in order to converge the networks fast and efficiently without unnecessary transformations that result in introvert networks as described beforehand. The success or failure of the input choices is well recognized at the end of the process but hardly predictable at the beginning of the process. Bottom-up approaches incorporate the concept of emergence. Though according to the authors emergence is not linked with randomness, it is likely to contribute towards creative and though-out-of-the-box design solutions. Like in chess or Go playing, where the human mind foresees and processes ahead a limited number of movements that will bring the player closer to the victory, this design workflow is significantly constrained if it relies on human logic only. The efficiency of the current PEER framework will be advanced through machine intelligence. The machine, trained as an intelligent co-designer – e.g. through reinforcement learning –, will foresee and prevent undesired design situations, imposed by human-made decisions. It will suggest sequences of design decisions that allow for efficient exploration and sufficiently-controlled design solutions. An approach that builds on human–machine peer collaboration will allow the machine to take over the design evolution for a number of transformations, before the human takes back the lead to impose qualitative criteria—i.e. aesthetics. While this is still work in progress, the authors are confident that the PEER framework constitutes a strong base for development.

Future development considers the introduction of constraints as part of the policy. At a transformation level, their impact will be reflected on the size of the *auxiliary domain* and subsequently the *feasibility domain*. At a global level, the introduction of constraints will safely ensure non intersecting bars, constructable nodes, controlled axial forces, controlled bar lengths etc. The possibility to add/remove bars is also considered. Currently, new bars are easily added by replacing existing bars by the axial developed forces. Having a new set of two interim forces, new policy transformations can be applied leading to an updated network and topology. In order to remove bars, the possibility to fuse neighbouring nodes is envisioned. Nodal coordinates shall be updated based on proximity or minimum bar lengths, resulting in a sophisticated and practical way to remove bars. Dynamic change of the permanent inputs (i.e. design domain, $\mathcal{D}_d$), as those are described in Fig. 7, is expected to contribute towards a deeper understanding of the whole design problem and subsequently provoke creativity and out-of-the-box thinking. The diversity of the generated design candidates is questionable. Therefore, future development will evaluate the capability of the proposed grammar to increase diversity of design solutions. The evaluation requires graph classification of the generated structural topologies, achieved with the help of deep graph convolutional neural networks (DGCNN) [39].

The accomplishment of the short and long term objectives listed above is expected to reveal PEER's full potential for the generation of highly controlled spatial bar networks in static equilibrium. Namely, it will provide designers with a powerful new design toolkit to synthetize systems that provide a balanced answer to complex architectural/structural contexts with reduced environmental impacts and controlled production costs. The relevance of this research accords both with academia and industry. Inside academia, direct applications to the teaching of structures in schools of civil engineering and architecture are foreseen. The incremental generation and extrapolation of the intuitive equilibrium-related constraints that are imposed by each transformative step can be scholarly studied too. Outside of academia,

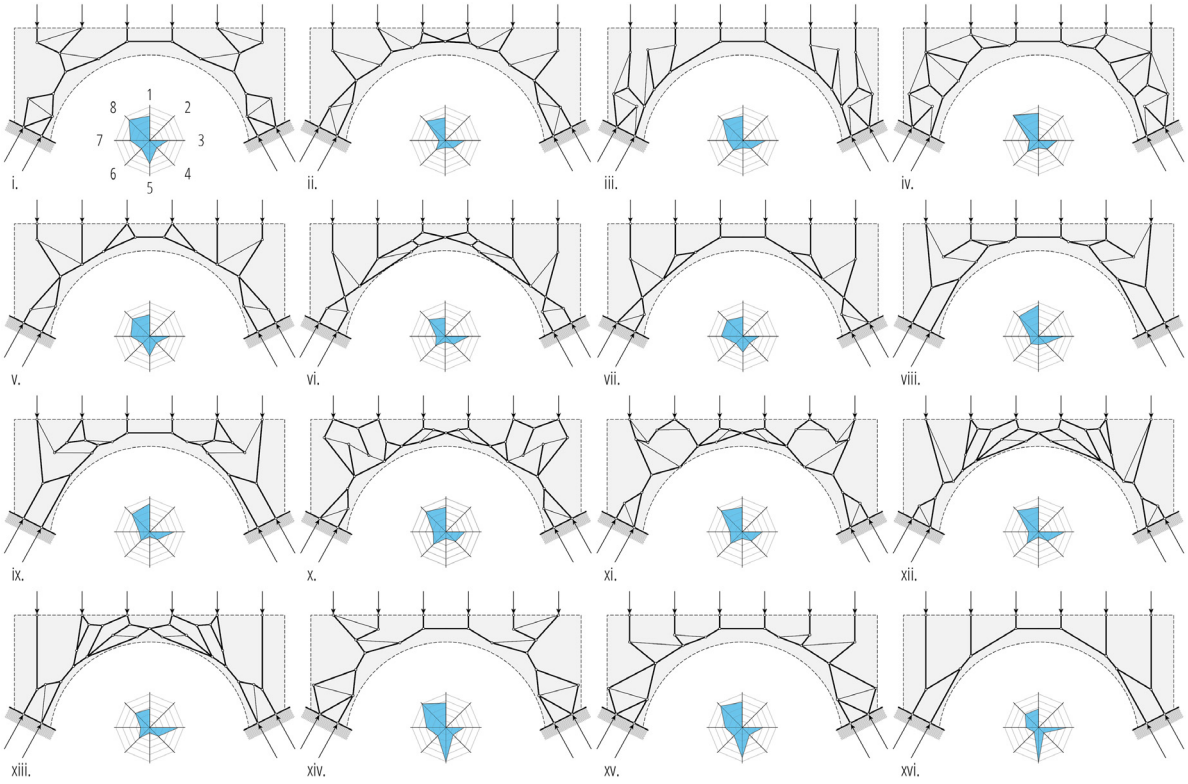

Fig. 13. A collection of diverse arch bridge-like bar networks as a result of different design input parameters. Key metrics recorded on a spider graph (1: maximum force magnitude, 2: number of bar intersections, 3: maximum bar length, 4: number of nodes, 5: minimum force magnitude, 6: number of bars, 7: minimum bar length, 8: static action).

automated and intelligent generation of early-stage structural design is of great importance to architectural and engineering offices and provides the following benefits:

- less exchanges between architects and engineers at the early stage thanks to the embedded equilibrium-awareness;
- fast generation of alternative design variants in response to emergent constraints and requirements;
- novel and bespoke structural design variants at no additional cost;

9. Conclusion

This paper presented PEER, a new computational design framework to explore variant topologies in static equilibrium represented as bar networks within specified geometric domains. The topologies are not necessarily triangulated and do not result from optimization procedures. Exploration follows an incremental policy-based approach which is defined by four high-level rules. These rules collaboratively embed static equilibrium constraints. The framework allows step-wise control over the generation, including backtracking to previous states. This control over the design process, along with the use of policy-based inputs and other rules expressions—i.e. convergence/divergence "on demand" through the *entropy rate* or direct control over the force magnitude in the bars through the *force indeterminacies rule*—makes PEER a new suitable framework for structural design exploration (DSE). Case studies demonstrated how PEER is capable of generating diverse networks of bars in static equilibrium in 2D and 3D.

Above all, the PEER framework brings up a new way of exploring/designing topologies in static equilibrium in a way that analysis software does not allow for. All design outputs are statically-valid structural forms though the policy definition is not bounded to specific typologies. Topology results from the applied policies but is neither provided explicitly/implicitly nor is known a priori. It reflects the designer's input choice but does not resort to common topology patterns—i.e. triangulation. As an early-stage tool it stands neither as an analysis tool nor as a high-end structural engineering tool for the latest-stages of a project. Hence, the generated design variants require further processing in order to consider multiple load cases and combinations of them, resulting in the introduction of bars that have not been considered in advance. Typical examples are bracing bars. Furthermore, as a tool it is not applicable to hydraulic engineering (dam design, dredging engineering, coastal engineering) or geoengineering (soil mechanics, foundation engineering, embarkments etc.). As a design approach, it is not meant to replace long-standing approaches such as continuum topology optimization (BESO, SIMP etc.) or ground structures approaches which are well-known for the generation of structures.

Despite the design freedom PEER provides, the sequence of design inputs greatly impacts the end result, in a way that is often hard to predict. In other words, it is hard for a designer to evaluate the effects of a chosen policy onto the eventual structural form. Therefore, future assistance from machine intelligence is highly recommended for better-controlled solutions and even broader design space exploration. The incremental, policy-based approach followed in the STEM framework is particularly suited for both training such machine and for supporting human–machine synergy during the design process, still aiming at provoking creativity and fighting design fixation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.cad.2023.103518.

References

[1] Rittel HW, Webber MM. Dilemmas in a general theory of planning. Policy Sci 1973;4(2):155–69. http://dx.doi.org/10.1007/BF01405730.

[2] Purcell AT, Gero JS. Design and other types of fixation or is fixation always incompatible with innovation ? Des Stud 1996;17(4):363–83. http://dx.doi.org/10.1016/S0142-694X(96)00023-3.

[3] Paulson Jr BC. Designing to reduce construction costs. J Construct Div 1976;102(4):587–92. http://dx.doi.org/10.1061/JCCEAZ.0000639.

[4] Parmee IC. Evolutionary and adaptive computing in engineering design. Springer Science & Business Media; 2012, http://dx.doi.org/10.1007/978-1-4471-0273-1.

[5] Mueller C, Ochsendorf J. Combining structural performance and designer preferences in evolutionary design space exploration. Autom Constr 2015;52:70–82.

[6] Fivet C, Zastavni D. Robert Maillart's key methods from the Salginatobel bridge design process (1928). J Int Assoc Shell Spat Struct 2012;53(171):39–47.

[7] Huerta S. Structural design in the work of Gaudí. Archit Sci Rev 2006;49(4):324–39. http://dx.doi.org/10.3763/asre.2006.4943.

[8] Zastavni D. The structural design of Maillart's Chiasso Shed (1924): A graphic procedure. Struct Eng Int 2008;18(3):247–52. http://dx.doi.org/10.2749/101686608785096496.

[9] Allen E, Zalewski W. Understanding Famous Structures Through Simple Graphical Analyses. In: 84th ACSA annual meeting - building technology conference. 1996, p. 3–8.

[10] Conzett J, Mostafavi M. Structure as space: engineering and architecture in the works of jürg conzett. AA Publications; 2006.

[11] Stasiuk D. Design modeling terminology. ProvingGround 2013.

[12] Simon HA. The structure of ill structured problems. Artificial Intelligence 1973;4(3–4):181–201. http://dx.doi.org/10.1016/0004-3702(73)90011-8.

[13] von Buelow P. Advantages of evolutionary computation used for exploration in the creative design process. J Integr Des Process Sci 2007;11(3):5–18. http://dx.doi.org/10.5555/1517398.1517400.

[14] Martini K. Harmony search method for multimodal size, shape, and topology optimization of structural frameworks. J Struct Eng 2011;137(11):1332–9. http://dx.doi.org/10.1061/(ASCE)ST.1943-541X.0000378.

[15] Von Buelow P. ParaGen: Performative exploration of generative systems. J Int Assoc Shell Spat Struct 2012;53(4):271–84.

[16] Harding J. Evolving parametric models using genetic programming. In: ECAADe 2016, Vol. 1. 2016, p. 423–32.

[17] Harding J, Brandt-Olsen C. Biomorpher: Interactive evolution for parametric design. Int J Archit Comput 2018;16(2):144–63. http://dx.doi.org/10.1177/1478077118778579.

[18] Ohlbrock PO, D'Acunto P. A computer-aided approach to equilibrium design based on graphic statics and combinatorial variations. Comput Aided Des 2020;121:102802. http://dx.doi.org/10.1016/j.cad.2019.102802.

[19] Harding J, Joyce S, Shepherd P, Williams C. Thinking topologically at early stage parametric design. Adv Archit Geom 2012 2012;67–76. http://dx.doi.org/10.1007/978-3-7091-1251-9_5.

[20] Harding JE, Shepherd P. Meta-parametric design. Des Stud 2017;52:73–95. http://dx.doi.org/10.1016/J.DESTUD.2016.09.005.

[21] Stiny G, Gips J. Shape grammars and the generative specification of painting and sculpture. In: The best computer papers of 1971, Vol. 71. 1971, p. 1460–5.

[22] Stiny G, Mitchell WJ. The palladian grammar. Environ Plan B: Plann Des 1978;5(1):5–18. http://dx.doi.org/10.1068/b050005.

[23] Shea K, Cagan J, Fenves SJ. A shape annealing approach to optimal truss design with dynamic grouping of members. Trans ASME, J Mech Des 1997;119(3):388–94. http://dx.doi.org/10.1115/1.2826360.

[24] Shea K, Cagan J, et al. Languages and semantics of grammatical discrete structures. Ai Edam 1999;13(4):241–51. http://dx.doi.org/10.1017/S0890060499134012.

[25] Geyer P. Multidisciplinary grammars supporting design optimization of buildings. Res Eng Des 2008;18(4):197–216. http://dx.doi.org/10.1007/s00163-007-0038-6.

[26] O'Neill M, Mcdermott J, Swafford JM, Byrne J, Hemberg E, Brabazon A. Evolutionary design using grammatical evolution and shape grammars : designing a shelter. Int J Des Eng 2010;3(1):1–23. http://dx.doi.org/10.1504/IJDE.2010.032820.

[27] Beirão JN, Duarte JP, Stouffs R. Creating specific grammars with generic grammars: Towards flexible urban design. Nexus Netw J 2011;13(1):73–111. http://dx.doi.org/10.1007/s00004-011-0059-3.

[28] Chakrabarti A, Shea K, Stone R, Cagan J, Campbell M, Hernandez NV, Wood KL. Computer-based design synthesis research: an overview. J Comput Inf Sci Eng 2011;11(2). http://dx.doi.org/10.1115/1.3593409.

[29] Byrne J, Fenton M, Hemberg E, McDermott J, O'Neill M, Shotton E, Nally C. Combining structural analysis and multi-objective criteria for evolutionary architectural design. In: Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics). LNCS, vol. 6625, 2011, p. 204–13. http://dx.doi.org/10.1007/978-3-642-20520-0_21.

[30] Zimmermann L, Chen T, Shea K. A 3D, performance-driven generative design framework: automating the link from a 3D spatial grammar interpreter to structural finite element analysis and stochastic optimization. Artif Intell Eng Des Anal Manuf 2018;32(2):189–99. http://dx.doi.org/10.1017/S0890060417000324, Publisher: Cambridge University Press.

[31] Tomei V, Faiella D, Cascone F, Mele E. Structural grammar for design optimization of grid shell structures and diagrid tall buildings. Autom Constr 2022;143. http://dx.doi.org/10.1016/j.autcon.2022.104588.

[32] Mueller CT. Computational exploration of the structural design space (Ph.D. thesis), Massachusetts Institute of Technology; 2014.

[33] Lee J, Mueller C, Fivet C. Automatic generation of diverse equilibrium structures through shape grammars and graphic statics. Int J Space Struct 2016;31(2–4):147–64. http://dx.doi.org/10.1177/0266351116660798.

[34] Mirtsopoulos I, Fivet C. Grammar-based generation of bar networks in static equilibrium with bounded bar lengths. In: Proceedings of the IASS annual symposium 2020/21 and the 7th international conference on spatial structures. Guilford, UK; 2021.

[35] Strobbe T, Pauwels P, Verstraeten R, De Meyer R, Van Campenhout J. Toward a visual approach in the exploration of shape grammars. Artif Intell Eng Des Anal Manuf: AI EDAM 2015;29(4):503. http://dx.doi.org/10.1017/S0890060415000475.

[36] Mirtsopoulos I. Libra: A grammar for generating structural topologies. 2022, http://dx.doi.org/10.5281/zenodo.7180829, https://github.com/StructuralXplorationLab/Libra.

[37] Baker W, Beghini L, Mazurek A, Carrion J, Beghini A. Maxwell's reciprocal diagrams and discrete Michell frames. Struct Multidiscip Optim 2013;48. http://dx.doi.org/10.1007/s00158-013-0910-0.

[38] Shea K, Aish R, Gourtovaia M. Towards integrated performance-driven generative design tools. Autom Constr 2005;14(2):253–64. http://dx.doi.org/10.1016/j.autcon.2004.07.002.

[39] Data61 C. StellarGraph machine learning library. 2018, https://github.com/stellargraph/stellargraph.