

# Libra

a grammar for generating structural topologies

user manual for 1.0.0

July 2022



# Table of contents

<b>Table of contents.....</b>	<b>iii</b>
<b>Introduction .....</b>	<b>1</b>
<b>1 Transformation process.....</b>	<b>4</b>
1.1 Overview.....	4
1.1.1 Initiate model.....	4
1.1.2 Select force(s) and topology .....	5
1.1.3 Place new node .....	5
1.1.4 Set indeterminacies.....	5
1.1.5 Add interim forces .....	5
1.1.6 Update model.....	5
1.2 Workflow of transformations.....	6
1.2.1 Input parameters .....	7
1.2.2 Permanent parameters .....	7
1.2.3 Transient parameters .....	7
<b>2 Design input decisions.....</b>	<b>8</b>
2.1 The transformation impacts .....	8
2.2 The interim forces to eliminate.....	8
2.3 The transformation geometry.....	9
2.4 The transformation structural behavior.....	9
<b>3 Design/Domain constraints.....</b>	<b>11</b>
3.1 Entropy rate domain.....	11
3.2 Constructability domain.....	11
<b>4 Libra in Grasshopper .....</b>	<b>13</b>
4.1 Overview.....	13
4.2 Main components.....	14
4.2.1 Construct design domain .....	14
4.2.2 Construct model.....	14
4.2.3 Construct force(s) selection rule.....	15
4.2.4 Construct node placement rule .....	17

4.2.5	Construct force indeterminacies rule .....	18
4.2.6	Construct policy.....	19
4.2.7	Apply transformation.....	19
4.3	Supplementary components .....	20
4.3.1	Domain category .....	20
4.3.1.1	Construct domain.....	20
4.3.1.2	Deconstruct domain .....	20
4.3.1.3	Visualize domain .....	21
4.3.2	ForceSet category .....	21
4.3.2.1	Construct Force Set.....	21
4.3.2.2	Deconstruct Force Set.....	21
4.3.2.3	Force Set Domains.....	22
4.3.2.4	Visualize Force Set .....	22
4.3.3	Generate category .....	22
4.3.3.1	Construct Force Selection Rule .....	22
4.3.3.2	Construct Node Placement Rule .....	22
4.3.3.3	Construct Force Indeterminacies Rule.....	22
4.3.3.4	Construct Transformation Policy .....	22
4.3.3.5	Apply Transformation Policy .....	22
4.3.3.6	Construct Transformation Policy (fast) .....	22
4.3.4	Model category .....	23
4.3.4.1	Construct Model .....	24
4.3.4.2	Select Force .....	24
4.3.4.3	Model Geometry.....	24
4.3.4.4	Model History.....	24
4.3.4.5	Model Undo .....	25
4.3.4.6	Model Metrics (spider graph) .....	25
4.3.4.7	Model Metrics (text) .....	26
4.3.4.8	Model Export to Illustrator .....	26
4.3.5	Exploration category .....	27
<b>5</b>	<b>Interactive design space exploration .....</b>	<b>28</b>
5.1	Concept.....	28
5.2	Implementation .....	28
<b>Appendix</b>	<b>Rules .....</b>	<b>30</b>
Appendix 1	Entropy rate .....	30

---

Appendix 2	Force(s) selection rule .....	30
Appendix 3	Node placement rule.....	30
Appendix 4	Force indeterminacies rule .....	30



# Introduction

Structural design describes forms that synthesize structural behaviors—they ‘tell’ how a structure withstands and/or deforms under the application of loads. Structural behaviors are subject to various laws, among which static equilibrium of forces plays a central role. The concept of static equilibrium in structures is one that can be reduced to minimal, abstract equilibrium configurations that consist of vectors, nodes, and bars in compression or tension.

For given loading and boundary conditions, the number of arrangements of tension and compression elements in static equilibrium is theoretically infinite. In practice, only a small, finite subset of equilibrium configurations are explored and/or employed. Design space exploration (DSE) is a creative process that consists of incremental generation of multiple design variants (design options) and is chronologically framed in the early design stage. DSE helps designers to find high quality design alternatives and fights against premature design fixation, which notoriously results in the generation of resembling design variants.

Libra describes an incremental transformative process that allows the transition from an incomplete network (Fig. 1, stage 00) to a complete one (Fig. 1, stage 13). In other words, the incremental elimination of interim forces in the network. At every intermediate step the network is in static equilibrium (interim when interim forces—vectors in cyan—are still present, or global when no interim forces exist).

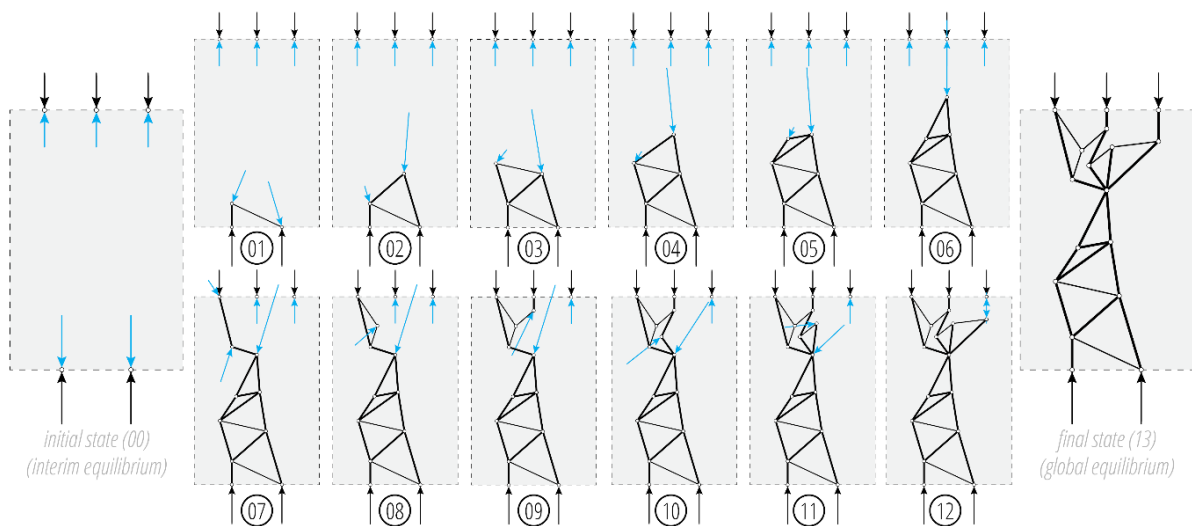
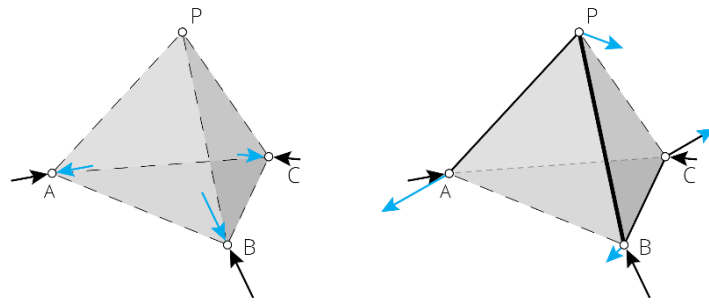


Fig. 1: Incremental transformative process. Notice the transition from an incomplete network to a complete one.

Each step describes a transformation that consists of the introduction of:

- a new **node** (P)
- one, two or three new **bars**
- one, two or three new **interim forces** imposed by the static equilibrium condition

Fig. 2 illustrates a simple example of a transformation step.



*Fig. 2: Before/After transformation in space. Black arrows represent force actions that are applied externally to the (sub)system. Cyan arrows represent interim forces. Bars in compression are thick lines. Bars in tension are thin lines*





# 1 Transformation process

The transition to a complete network implies the successive application of the transformation step. Each transformation step consists of five stages illustrated at Fig. 3. Sets of different inputs control the execution of each stage.

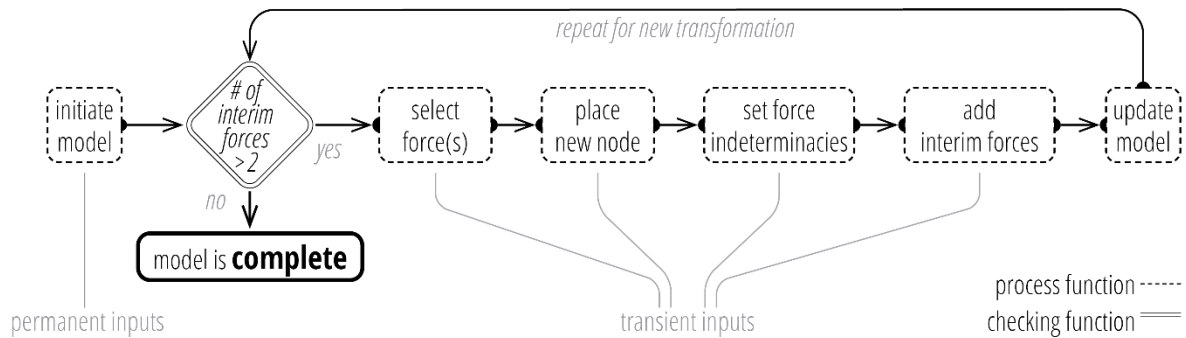


Fig. 3: Schematic workflow of successive policy-based transformations

The first stage (*initialize model*) sets up the process, it is executed once, and it is fed with permanent parameters. The remaining stages (*select force(s) and topology*, *place new node*, *set indeterminacies*, *add interim force(s)* and *update model*) impose the transformation step, they are repeated until the network is complete and they are fed with transient parameters that are subject to change following new design intentions that may emerge during the process.

## 1.1 Overview

### 1.1.1 Initiate model

All transformations are circumscribed within the *design domain* defined at the beginning of the process. These boundaries are constructed beforehand and represent the designer's initial constraints. At the same stage, the *external forces* (applied loads and support reactions) are described and construct the *model* of the process. Their definition is linked to the usage of the final structure—e.g., number of people per square meter etc.—and, potentially, other geographic aspects—e.g., snow/wind loading expressed as force per meter/area. The support reactions also derive from the construction field and the designer's intentions.

The model construction is followed by the introduction of interim forces that bring it in interim static equilibrium. Starting from that moment and all along the transformative process, if the model contains more than two interim forces (it is a disconnected network), the successive transformations continue, as the transition to a complete network requires more transformation steps. Else, if exactly two interim forces are left in the model, their vectors lie on the same line of action, they have the same length and they are opposite. Thus, a bar (in compression or tension based on the interim forces) replaces them and the network gets complete. The last conclusion is made because of the rotational and translational equilibrium condition that is always satisfied by the (initial) model.

The next sections each correspond to a recursive stage in the transformation process.

### 1.1.2 Select force(s) and topology

The first stage in the loop of repeated stages consists of selecting the interim forces to eliminate and the topological configuration of the topological configuration of the introduced bars (Fig. 6). The selection is operated either *explicitly* or *implicitly* (by means of a **force selection rule**) and imposes the starting point of the bars network growth.

In both cases, up to three interim forces are selected from the pool of interim forces. The choice of interim forces creates a *monomial*, a *binomial* or a *trinomial* and respectively allows less or more transformation types. An explicit definition of the intended topological configuration (Fig. 6) of the bars to create must be provided by the designer.

### 1.1.3 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 *implicitly* (by means of a **node placement rule**) and imposes the geometry of the growth.

### 1.1.4 Set indeterminacies

The third stage in the loop consists of defining the developed forces along the newly introduced bars. In a similar fashion to the previous stages, they are defined *explicitly* or *implicitly* (by means of a **force indeterminacies rule**) and impose the structural behavior of the growth. This parameter is only necessary when the interim forces to be introduced cannot be calculated because the system is indeterminate.

### 1.1.5 Add interim forces

The fourth stage in the loop consists of guaranteeing static equilibrium in the model by introducing interim forces. Explicit parameters—i.e., the *entropy rate*—controls their total number and regulates the speed of growth. Their direction and magnitude are also calculated at this stage by solving equilibrium equations.

### 1.1.6 Update model

The last stage in the loop does not expect any input parameters from the designer. Its role is synthetic and consists of updating the model with the new elements that were created by the policy.

## 1.2 Workflow of transformations

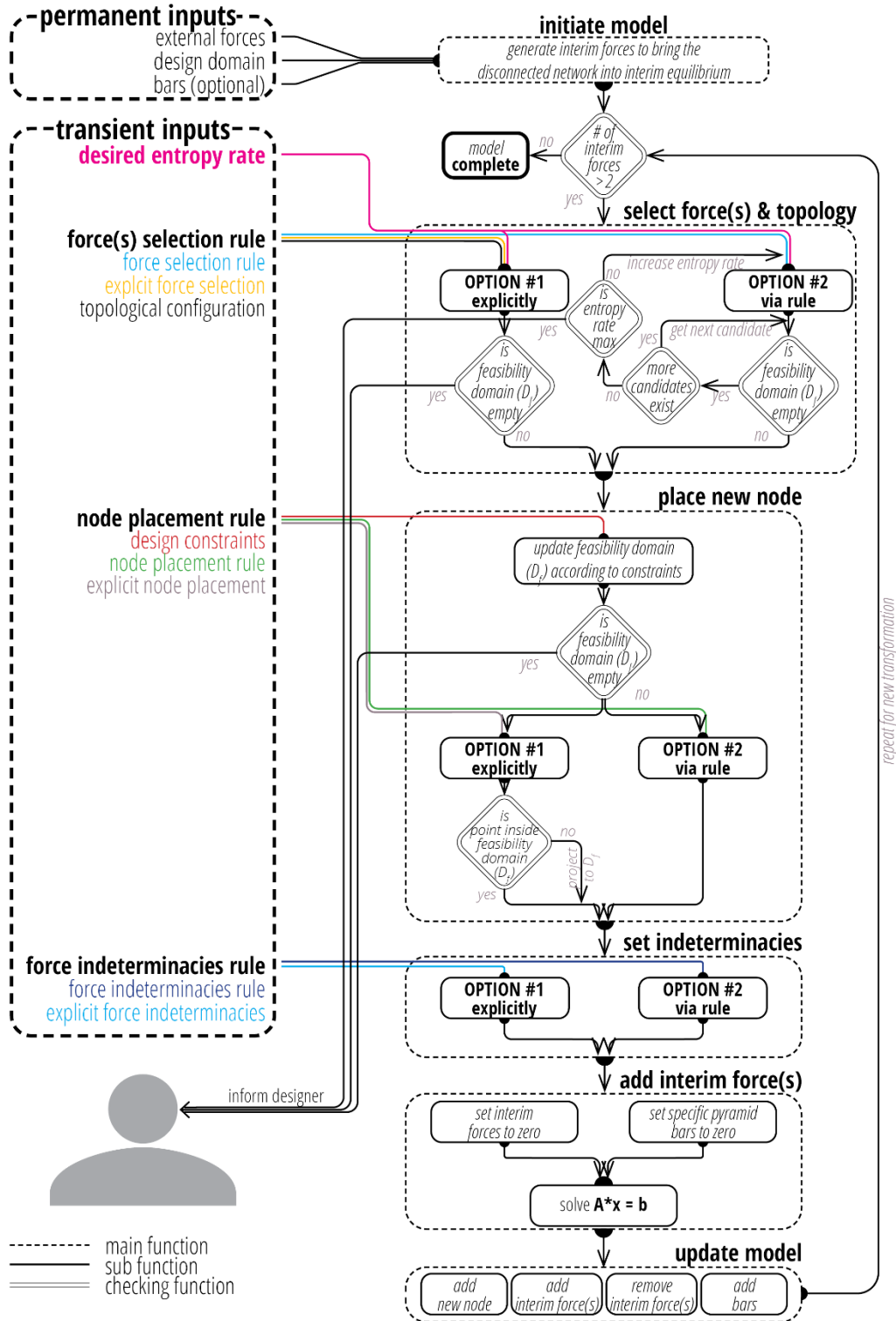


Fig. 4: Algorithmic workflow of successive policy-based transformations

Fig. 4 illustrates all the intermediate stages until the completion of a transformation step and visualizes in groups the types of parameters and the stage of their intervention. The first three recursive stages (*select force(s) and topology*, *place new node* and *set indeterminacies*) process the provided input which has been provided either *implicitly* or *explicitly* (by means of a rule). At every stage that receives explicit input, checks are carried out to ensure compatibility between the input values and the current state of the model. When the checks fail, the designer is informed with a respective message urging him/her to update the incompatible parameters. If input is provided via rules, less checks are carried. When the

checks fail, the algorithm automatically updates the incompatible parameters to ensure the continuity of the transformative process in a seamless way. If, exceptionally, no alternative, but still compatible, input can be found, the designer is informed, and the process terminates.

### 1.2.1 Input parameters

The transformative process exploits different types of input parameters, according to their functionality within the algorithmic workflow. *Permanent* and *transient* input parameters are considered. The execution frequency of each stage of the process controls the definition frequency of each associated parameter.

### 1.2.2 Permanent parameters

The list of permanent input parameters includes the *design domain*, the *external forces* and optionally a set of disconnected *bars* as part of an interim network. They are defined at the very beginning of the process and construct the model of the process. The model construction ensures the static equilibrium of the network. To achieve this condition, a set of simple operations are made. In absence of bars adjacent to the external forces anchor points, interim forces (opposite and equal to the applied forces or support reactions) are pre-computed. For disconnected networks that contain a few bars, case-specific interim forces retain static equilibrium, as a result of previous transformations.

### 1.2.3 Transient parameters

Each transformation is the result of a policy which describes design intentions in a descriptive way. Its definition goes beyond numerical input parameters. Four rules define a policy as shown at Fig. 5. The next sections each correspond to the notion of each rule.

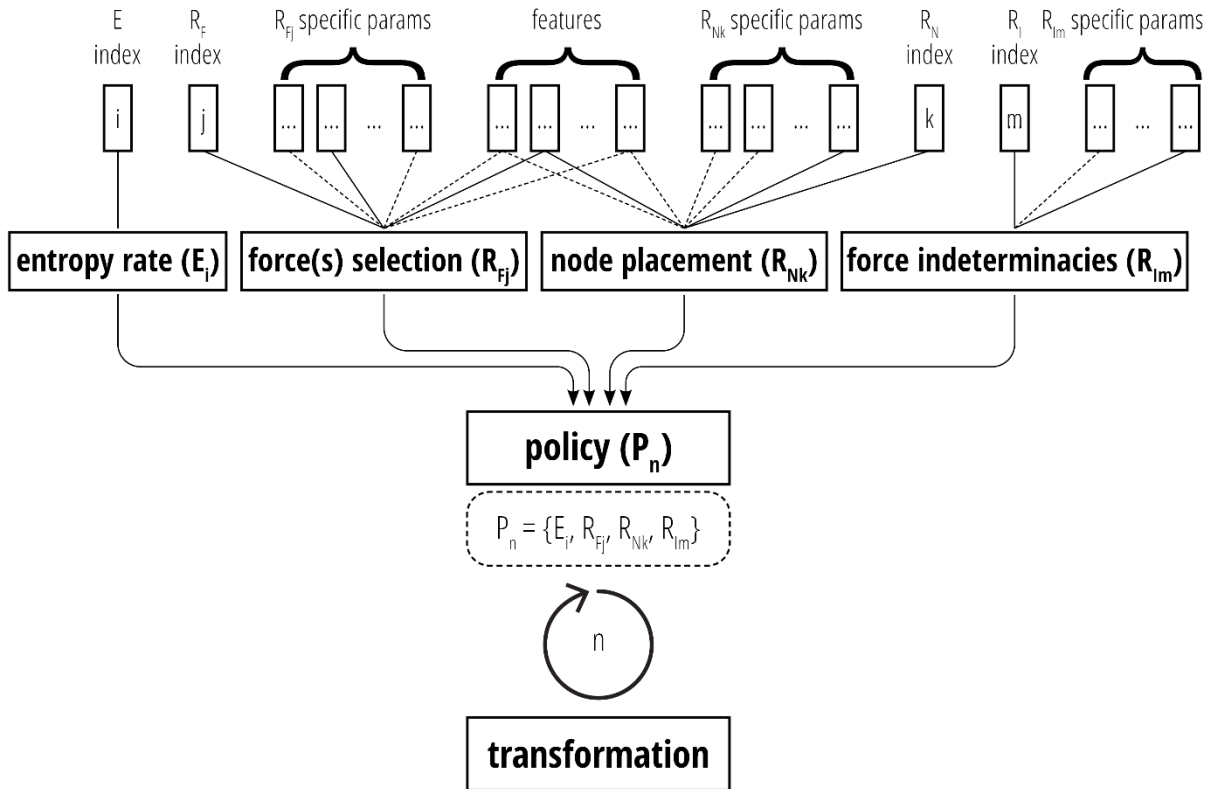


Fig. 5: The set of transient parameters that describe the rules and subsequently the policy which leads to the network transformation

## 2 Design input decisions

Each transformation step is controlled by four design decisions listed below.

### 2.1 The transformation impacts

It refers to the numerical difference of interim forces in the network before and after the transformation, described as the **entropy rate**. As a designer, you express your intention through a ternary decision among:

- *convergence*; decrease the number of interim forces
- *stagnation*; stagnate the number of interim forces
- *divergence*; increase the number of interim forces

**Hint:** This decision directly affects the speed of the transition to a complete network.

### 2.2 The interim forces to eliminate

It refers to the selection of the interim forces as the starting point of each transformation and the topology of the new sub-network, described as **force(s) selection rule**. According to Fig. 6, the possible topologies are:

- in-between
- peripheral
- central

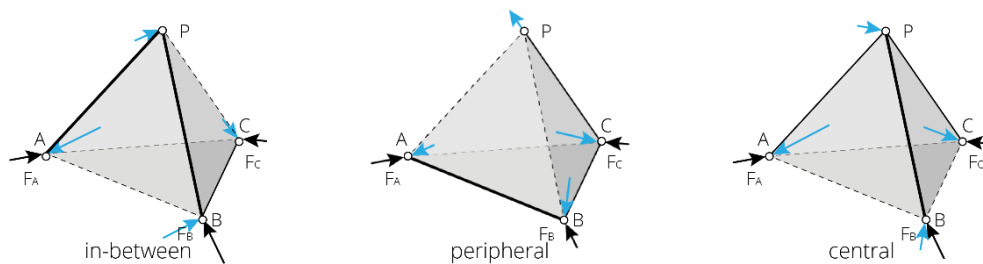


Fig. 6: The possible connectivity topologies. Black arrows represent force actions that are applied externally to the (sub)system. Cyan arrows represent interim forces. Bars in compression are thick lines. Bars in tension are thin lines.

The available **force(s) selection** options are:

- *monomial*; only one force vector is selected
- *binomial*; two force vectors are selected
- *trinomial*; three force vectors are selected

As a designer, you make the selection *implicitly* or *explicitly*.

- *explicitly*; manual selection of the interim forces and the topology
- *implicitly*; automated selection of forces and topology through a designer-chosen rule (there is a long list of self-explanatory rules - Appendix 2; additional parameters might be required)

## 2.3 The transformation geometry

It refers to the position of the new node P, described as **node placement rule**. As a designer, you do the placement *implicitly* or *explicitly*.

- *explicitly*; manual placement of the node
- *implicitly*; automated placement through a designer-chosen rule (there is a list of rules - Appendix 3; additional parameters might be required)

## 2.4 The transformation structural behavior

It refers to the type (compression/tension) and magnitude of the axial forces developed along the introduced bars, described as **force indeterminacies rule**. As a designer, you make the selection *implicitly* or *explicitly*.

- *explicitly*; manual provision of axial forces (N1, N2, N3) for all three new bars
- *implicitly*; automated definition of axial forces (there is a list of self-explanatory rules - Appendix 4; additional parameters might be required)

Fig. 7 illustrates all possible transformations in compliance with the design decisions made. The provided force diagram confirms the static equilibrium condition after the transformation.

**Hint:** Consider drafting the force diagram of each transformation while designing! It will give you a better understanding of the transformation final geometry

	<i>initial state</i>	CONVERGENCE	STAGNATION	DIVERGENCE
<b>MONOMIAL</b>		<b>X</b>		
<b>BINOMIAL</b>		<b>IN-BETWEEN</b>		
		<b>PERIPH / CENTR</b>		
		<b>IN-BETWEEN</b>		
		<b>PERIPHERAL</b>		
<b>TRINOMIAL</b>		<b>CENTRAL</b>		
		<b>PERIPHERAL</b>		

Fig. 7: Transformation step for all combinations of entropy rates, numbers of interim forces and topological configurations. Black arrows represent force actions that are applied externally to the (sub)system; cyan arrows represent interim forces, all circumscribed within a primitive design domain (grey). Bars in compression are thick lines. Bars in tension are thin lines.



## 3 Design/Domain constraints

### 3.1 Entropy rate domain

Libra provides a design workflow that ensures static equilibrium of the generated network. This condition is satisfied by constraining the placement of the new node within a specific geometric domain, named **entropy rate domain** (Fig. 8). In other words, the entropy rate domain describes all possible locations that node P can take, while the network retains static equilibrium. The size of this domain varies, from a single point in space to a volume. Do not worry, as a designer, you do not have to compute this domain. If you manually request the placement of node P at a location outside of the entropy rate domain, the algorithm will consider the closest projected location on the entropy rate domain, ensuring static equilibrium.

### 3.2 Constructability domain

Between node P and the existing nodes of the network, new bars are introduced. Their construction implies that the bars are not interrupted by voids or non-convexities of the design domain. This condition is satisfied when certain visibilities between nodes are satisfied. The geometric domain that describes all possible locations that node P can have, while ensuring that the necessary bars can be constructed uninterrupted is named **constructability domain** (Fig. 8). Do not worry, as a designer, you do not have to compute this domain. The algorithm will check that. Be aware though that the geometric domain where node P can be safely introduced during each transformation is the intersection of the two domains and is named **feasibility domain**.

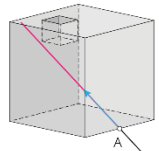
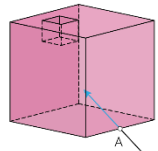
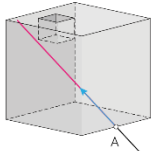
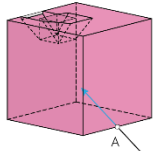
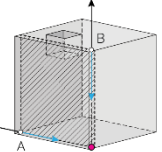
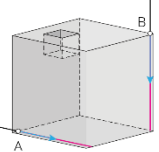
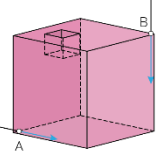
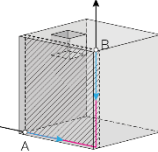
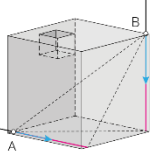
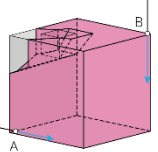
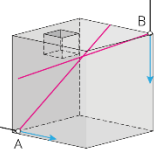
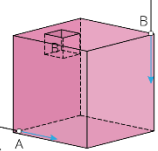
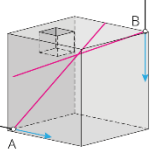
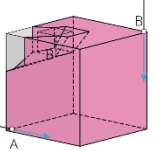
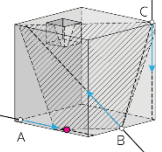
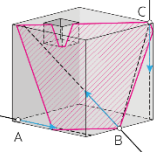
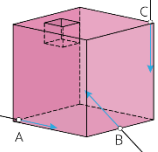
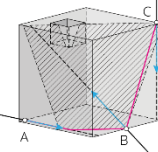
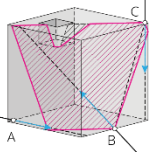
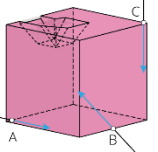
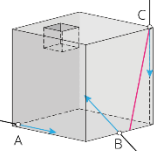
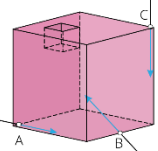
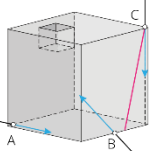
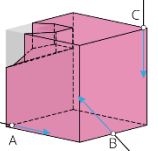
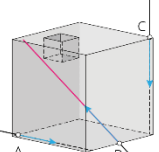
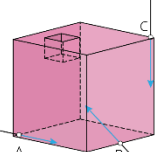
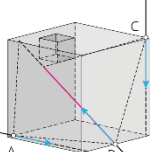
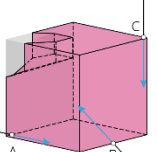
		ENTROPY RATE DOMAIN			CONSTRUCTABILITY DOMAIN		
		CONVERGENCE	STAGNATION	DIVERGENCE	CONVERGENCE	STAGNATION	DIVERGENCE
MONOMIAL		X			X		
			iii.	ix.		xvii.	xxiii.
BINOMIAL	IN-BETWEEN						
		i.	iv.	x.	xv.	xviii.	xxiv.
BINOMIAL	PERIPH / CENTR	X			X		
			v.	xi.		xix.	xxv.
TRINOMIAL	IN-BETWEEN						
		ii.	vi.	xii.	xvi.	xx.	xxvi.
	PERIPHERAL	X			X		
			vii.	xiii.		xxi.	xxvii.
TRINOMIAL	CENTRAL	X			X		
			viii.	xiv.		xxii.	xxviii.

Fig. 8: Typologies of entropy rate and constructability domains. Black arrows represent force actions that are applied externally to the (sub)system; cyan arrows represent interim forces, all circumscribed within a primitive design domain (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 constructability domain.

## 4 Libra in Grasshopper

### 4.1 Overview

A primitive setup of the transformative design process with Libra is demonstrated below (Fig. 9). Each of the circled component groups accomplishes a certain task. Groups 1 and 2, provide the transient input parameters and groups 3-6 describe the transformation policy. Group 7 applies the transformation in compliance with the constructed policy. When the transition to a complete network uses multiple policies, groups 3-6 are repeated accordingly, and the transformed model (group 7) is provided as input for the upcoming transformations.

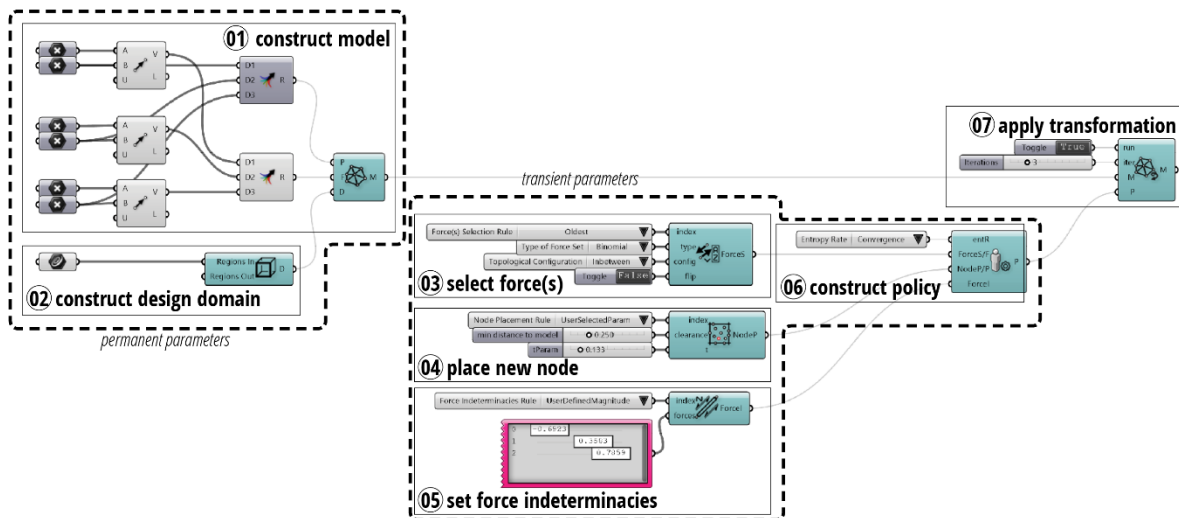


Fig. 9: Grasshopper canvas overview of a complete transformation setup that transforms the interim network according to a policy defined via rules.

The complete arsenal of Grasshopper components that Libra brings is grouped in different categories (Fig. 10).

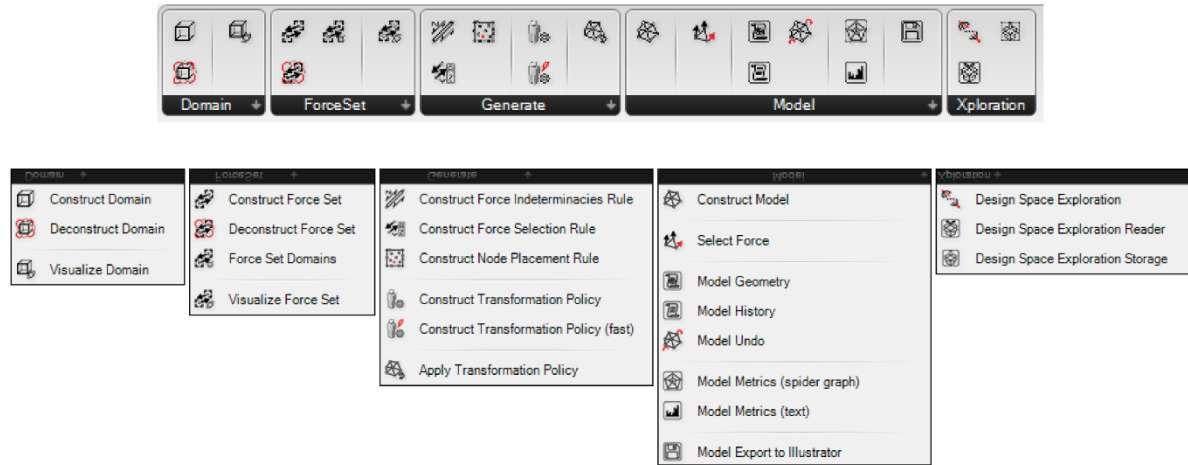


Fig. 10: Complete Libra toolbar

## 4.2 Main components

### 4.2.1 Construct design domain

At the very beginning, the design domain ( $D_d$ ) is defined. Typically, flat closed regions and bounded volumetric representations (in the near future) are expected as input values. All accepted geometries that could describe a geometric domain, continuous or discontinuous, convex or non-convex, are listed as input parameters at the component. Voids are also considered although they are optional.



Fig. 11: Construct design domain

INPUT	DESCRIPTION
<b>regions in</b>	collection of planar closed regions included in $D_d$ ( <i>Curve</i> )
<b>regions out</b>	collection of planar closed regions excluded from $D_d$ ( <i>Curve</i> )
OUTPUT	DESCRIPTION
<b>D</b>	domain to grow the network of bars into ( <i>Domain</i> )

### 4.2.2 Construct model

The second component collects all the permanent input parameters and constructs the initial model ( $M$ ). The applied loads are given as vector inputs along with their anchor points and the design domain is taken from the output of the first component described above. This is the point where the model transformation starts from and therefore these input parameters are not provided elsewhere in the workflow.

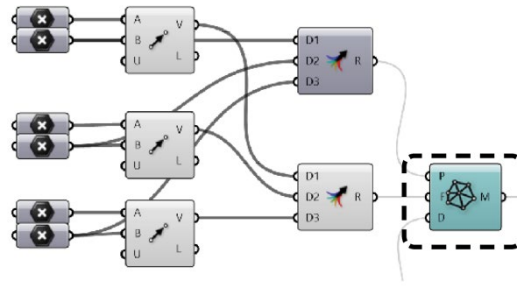


Fig. 12: Construct model

INPUT	DESCRIPTION
<b>P</b>	collection of anchor points for the external force vectors ( <i>Point3d</i> )
<b>F</b>	collection of external force vectors ( <i>Vector3d</i> )
<b>D</b>	domain to grow the network of bars into ( <i>Domain</i> )
OUTPUT	DESCRIPTION
<b>M</b>	interim model to transform incrementally ( <i>Model</i> )

The next step after the construction of an interim network is the definition of the policy out of a set of choices. The provision of policy parameters does not follow any particular order (Fig. 9). Hence, the following sections (4.2.3 - 4.2.5) describe steps operated in a random order.

### 4.2.3 Construct force(s) selection rule

The selection of forces is possible either *explicitly* or *implicitly* (by means of rules). Fig. 13 demonstrates three scenarios that construct different force selection rules with the same Grasshopper component. The designer selects the desired rule within the respective pool of rules (Appendix 2) via its index number. The list of input parameter slots is automatically updated to accommodate the rule specific parameters as well as other features that are necessary to fully describe the chosen force selection rule. None of the input parameters in the updated list of input slots is optional.

The first example (Fig. 13i), demonstrates the standard configuration of the *Construct force selection rule* component. Most of the rules do not require additional parameters for their definition. Therefore, the rule index is enough. Through the fourth input parameter, the designer optionally flips the order of forces. These input parameter slots are static and available regardless of the chosen rule. The flipping functionality applies to binomials only and has direct impact on the topological configuration – i.e., when a peripheral/central binomial stagnates (Fig. 7v and Fig. 7xi) – and ultimately the feasibility domain. In the second example (Fig. 13ii), a random selection of an in-between binomial is requested. The additional input slot requests a seed number that controls the selection randomness. The last example (Fig. 13iii), demonstrates the construction of a custom force selection rule, based on the proximity of the force anchor points to a designer-defined point.

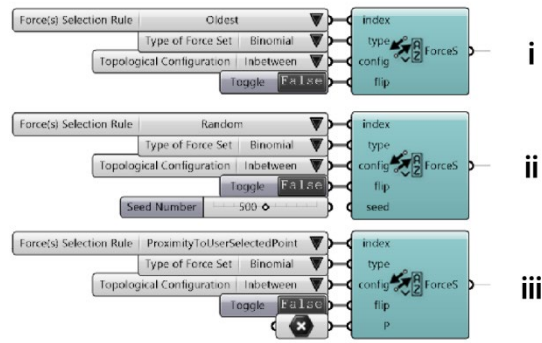


Fig. 13: Select force(s) and bar topology implicitly (via rules)

INPUT		DESCRIPTION
i	index	$j$ index of the force(s) selection rule ( <i>integer</i> )
	flip	boolean operator that flips the forces order in binomials ( <i>boolean</i> )
	type	type of interim forces ( <i>integer</i> / 0: monomial, 1: binomial, 2: trinomial)
	config	topological configuration ( <i>integer</i> / 0: in-between, 1: peripheral, 2: central)
ii	seed	seed number for randomness ( <i>integer</i> )
iii	P	manually defined point of reference ( <i>Point3d</i> )
OUTPUT		DESCRIPTION
forceS		force(s) selection rule, expressed as an object ( <i>ForceSelectionObj</i> )

To *explicitly* select the forces, the designer uses other Grasshopper components. The desired interim forces are manually retrieved from the interim model with the *Select Force* component (Fig. 14i). After, he/she provides them as input to the *Construct Force Set* component (Fig. 14ii), along with the desired entropy rate, the desired topological configuration, and the model at the state where the interim forces were selected from.

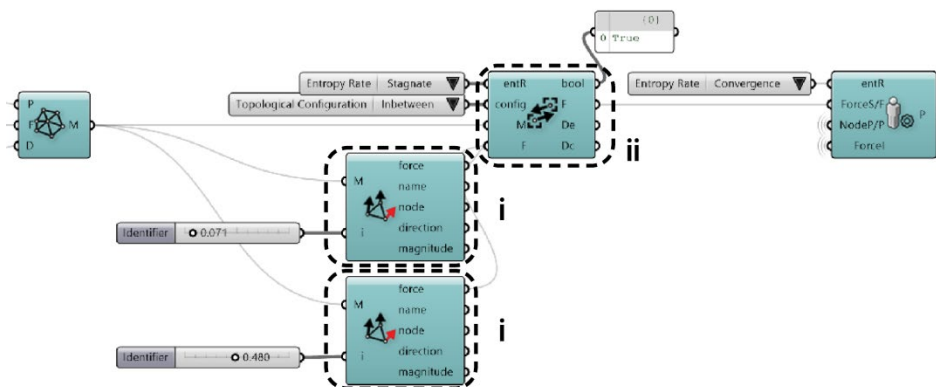


Fig. 14: Select force(s) and bar topology explicitly

The *Select Force* component selects any kind of forces from the provided model and returns its properties. The force indices are remapped to a new domain [0,1] and the selection is done through browsing between the domain bounds.

INPUT		DESCRIPTION
M		interim model to select interim forces from ( <i>Model</i> )
i		identifier of the force: integer id, string name starting with “F_”, browsing double between 0 and 1, or the object itself ( <i>integer, string, double, Force</i> )

OUTPUT	DESCRIPTION
<b>force</b>	retrieved force ( <i>Force</i> )
<b>name</b>	force name ( <i>string</i> )
<b>node</b>	anchor point / node ( <i>Node</i> )
<b>direction</b>	direction vector of the force ( <i>Vector3d</i> )
<b>magnitude</b>	force magnitude ( <i>double</i> / “+” if force is pushing, “-” otherwise)

The *Construct Force Set* component constructs a selection of forces and computes the entropy rate domain and the constructability domain. The designer is responsible to confirm that the feasibility of both domains is approved. If feasibility is not ensured no mechanism seeks after an alternative set of forces.

INPUT	DESCRIPTION
<b>entR</b>	desired entropy rate ( <i>integer</i> / -1: convergence, 0: stagnation, 1: divergence)
<b>config</b>	topological configuration ( <i>integer</i> / 0: in-between, 1: peripheral, 2: central)
<b>M</b>	interim model that contains the interim forces ( <i>Model</i> )
<b>F</b>	interim forces ( <i>Force</i> )

OUTPUT	DESCRIPTION
<b>bool</b>	boolean operator indicating the feasibility of achieving the desired entropy rate for the set of choices made ( <i>boolean</i> )
<b>F</b>	selection of forces, expressed as an object ( <i>ForceSet</i> )
<b>D<sub>e</sub></b>	force set entropy rate domain ( <i>Domain</i> )
<b>D<sub>c</sub></b>	force set constructability domain ( <i>Domain</i> )

#### 4.2.4 Construct node placement rule

The placement of a new node is possible either *explicitly* or *implicitly* (by means of rules). Fig. 15 demonstrates three scenarios that construct different node placement rules with the same Grasshopper component. The designer selects the desired rule within the respective pool of rules (Appendix 3), via its index number. The list of input parameters slots is automatically updated to accommodate the rule specific parameters as well as other features that are necessary to fully describe the chosen node selection rule. None of the input parameters in the updated list of input slots is optional.

The first example (Fig. 15i), demonstrates the standard configuration of the *Construct node placement rule* component, where the node location is defined randomly inside the feasibility domain and no bar length bounds are considered. Most of the rules rely on randomness and therefore besides the index of the rule, the clearance distance and the seed number, no additional parameters are required for their definition. The seed number controls the randomness, and the clearance distance sets a minimum tolerance to prevent the placement of nodes on top of bars or existing nodes.

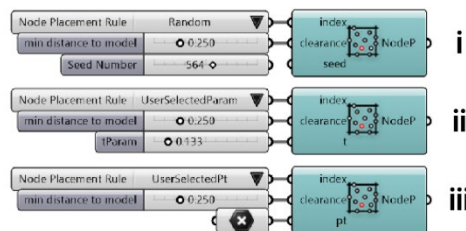


Fig. 15: Place new node via rules (i-iii) and explicitly (iv-v)

To *explicitly* select the forces, the designer can use the same Grasshopper component and selects the type of coordinates to provide; *homogeneous* or *cartesian*. Fig. 15ii demonstrates the former option, whilst Fig. 15iii demonstrates the latter.

INPUT		DESCRIPTION
i	index	$k$ index of the node placement rule ( <i>integer</i> )
	clearance	minimum distance to network ( <i>double</i> )
	seed	seed number for randomness ( <i>integer</i> )
ii	t	homogeneous coordinate that describes the new node location ( <i>double</i> )
ii	pt	cartesian coordinates that describe the new node location ( <i>Point3d</i> )
OUTPUT		DESCRIPTION
nodeP		node placement rule, expressed as an object ( <i>NodePlacementObj</i> )

Alternatively, the location of the new node can be fed directly (Fig. 16) to the *Construct Policy* component (section 4.2.6).

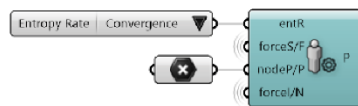


Fig. 16: Place new node explicitly and directly

## 4.2.5 Construct force indeterminacies rule

Setting the force indeterminacies is possible either *explicitly* or *implicitly* (by means of rules). Fig. 17 demonstrates two scenarios that construct different force indeterminacies rules with the same Grasshopper component. The designer selects the desired rule within the respective pool of forces (Appendix 4), via its index number. The list of input parameters slots is automatically updated to accommodate the rule specific parameters as well as other features that are necessary to fully describe the chosen force indeterminacies rule. None of the input parameters in the updated list of input slots is optional.

Currently only two rules exist. Their construction follows. The first example (Fig. 17i), demonstrates how the *Random* force indeterminacies rule is defined. The rule construction returns three random force magnitudes within a symmetric domain with designer-defined bounds. The seed number and the absolute value of maximum stresses are provided. The second example (Fig. 17ii), demonstrates how the *User defined magnitude* force indeterminacies rule is defined. The rule construction returns three explicitly defined force magnitudes that are not bounded to any domain.

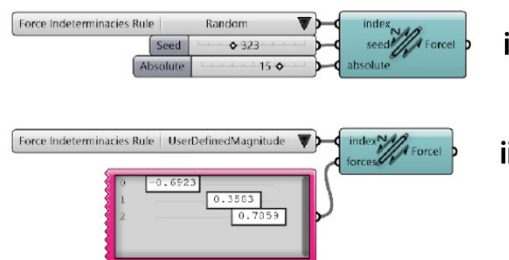


Fig. 17: Set force indeterminacies via rules



INPUT		DESCRIPTION
i	index	$m$ index of the force indeterminacies rule ( <i>integer</i> )
	seed	seed number for randomness ( <i>integer</i> )
	absolute	absolute magnitude of axial developed stress ( <i>double</i> )
ii	forces	magnitude of axial developed stress ( <i>GenePool</i> )
OUTPUT		DESCRIPTION
forceI		force indeterminacies rule, expressed as an object ( <i>ForceIndeterObj</i> )

To explicitly set the force indeterminacies, the designer can be feed directly the *Construct Policy* component (section 4.2.6) with a list of custom force magnitudes (Fig. 18).

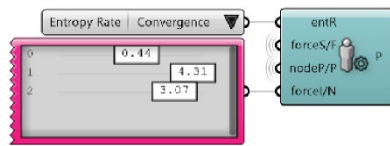


Fig. 18: Set force indeterminacies explicitly

## 4.2.6 Construct policy

The *force selection*, the *node placement* and the setting of *force indeterminacies* are required for the policy definition. The fourth aspect of the transformation policy – the *entropy rate* - is explicitly defined at the *Construct Policy* component. Similarly to the entropy rate, when the first three aspects are defined explicitly, they can be provided directly to the *Construct Policy* component (Fig. 14, Fig. 16 and Fig. 18). Otherwise, the provision of the respective rules is expected (Fig. 9).

INPUT		DESCRIPTION
entR		desired entropy rate ( <i>integer</i> / -1: convergence, 0: stagnation, 1: divergence)
forceS/F		force(s) selection rule, expressed as an object ( <i>ForceSelectionObj</i> ), <b>or</b> selection of forces, expressed as an object ( <i>ForceSet</i> )
nodeP/P		node placement rule, expressed as an object ( <i>NodePlacementObj</i> ), <b>or</b> cartesian coordinates that describe the new node location ( <i>Point3d</i> )
forceI/N		force indeterminacies rule, expressed as an object ( <i>ForceIndeterObj</i> ), <b>or</b> developed stress for bar #1, #2 and #3 ( <i>GenePool</i> )
OUTPUT		DESCRIPTION
P		transformation policy ( <i>Policy</i> )

## 4.2.7 Apply transformation

The last component launches the network transformation(s). The current state of the bars network is provided along with the transformation policy and the number of desired transformation steps. The designer simply launches the transformations with a *Run* toggle.



Fig. 19: Apply transformation Grasshopper component

INPUT	DESCRIPTION
<b>run</b>	boolean operator that launches the network transformation(s) ( <i>boolean</i> )
<b>iter</b>	number of transformation steps ( <i>integer</i> )
<b>M</b>	interim model to transform ( <i>Model</i> )
<b>P</b>	transformation policy ( <i>Policy</i> )
OUTPUT	DESCRIPTION
<b>M</b>	transformed model ( <i>Model</i> )

## 4.3 Supplementary components

### 4.3.1 Domain category

This category of components takes care of the construction, deconstruction and visualization of domains.

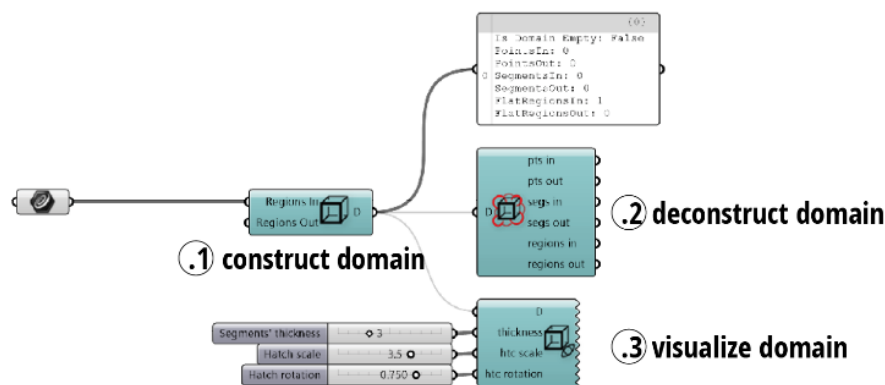


Fig. 20: Domain category Grasshopper components

#### 4.3.1.1 Construct domain

See 4.2.1.

#### 4.3.1.2 Deconstruct domain

It breaks down a domain into the specific geometric representations that define it.

INPUT	DESCRIPTION
<b>D</b>	domain ( <i>Domain</i> )
OUTPUT	DESCRIPTION
<b>pts in</b>	collection of points included in <i>D</i> ( <i>Point3d</i> )
<b>pts out</b>	collection of points excluded from <i>D</i> ( <i>Point3d</i> )
<b>segs in</b>	collection of segments included in <i>D</i> ( <i>Curve</i> )
<b>segs out</b>	collection of segments excluded from <i>D</i> ( <i>Curve</i> )
<b>regions in</b>	collection of planar closed regions included in <i>D</i> ( <i>Curve</i> )
<b>regions out</b>	collection of planar closed regions excluded from <i>D</i> ( <i>Curve</i> )

### 4.3.1.3 Visualize domain

Plot and color the geometric representations that define a domain. Points are highlighted with an index number, segments are colored with magenta and planar closed regions are hatched. For segments, the line thickness is controlled. For hatched regions, the scale and the rotation are also controllable.

INPUT	DESCRIPTION
<b>D</b>	domain to visualize ( <i>Domain</i> )
<b>thickness</b>	segments' / regions' thickness ( <i>integer</i> )
<b>htc scale</b>	hatch scale ( <i>double</i> )
<b>htc rotation</b>	hatch rotation in radians ( <i>double</i> )

## 4.3.2 ForceSet category

This category of components is necessary for the explicit selection of forces to apply the transformations onto. Particularly, these components take care of the construction, deconstruction and visualization of force sets. For detailed investigation of specific force sets, the extraction of the necessary domains is feasible too.

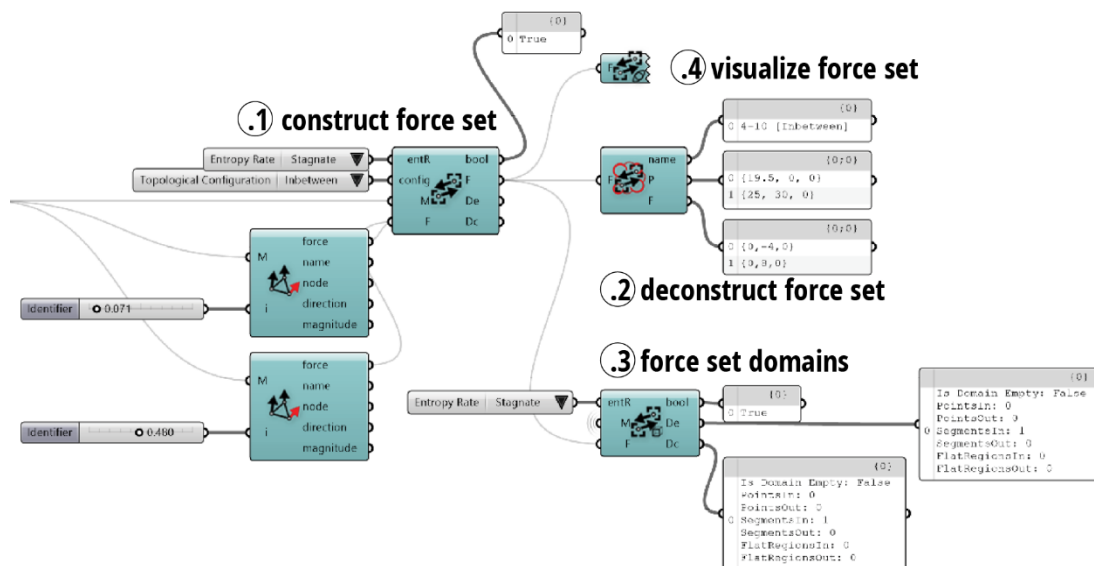


Fig. 21: ForceSet category Grasshopper components

### 4.3.2.1 Construct Force Set

See 154.2.3.

### 4.3.2.2 Deconstruct Force Set

INPUT	DESCRIPTION
<b>F</b>	force set ( <i>ForceSet</i> )

OUTPUT	DESCRIPTION
<b>name</b>	force set name (configuration included) ( <i>string</i> )
<b>P</b>	points involved with the force set ( <i>Point3d</i> )
<b>F</b>	force vectors involved with the force set ( <i>Vector3d</i> )

### 4.3.2.3 Force Set Domains

INPUT	DESCRIPTION
entR	desired entropy rate ( <i>integer / -1: convergence, 0: stagnation, 1: divergence</i> )
M	interim model that contains the force set ( <i>Model</i> )
F	force set ( <i>ForceSet</i> )
OUTPUT	DESCRIPTION
bool	is valid ( <i>boolean / true if the force set is valid</i> )
D <sub>e</sub>	force set entropy rate domain ( <i>Domain</i> )
D <sub>c</sub>	force set constructability domain ( <i>Domain</i> )

### 4.3.2.4 Visualize Force Set

INPUT	DESCRIPTION
F	force set to visualize ( <i>ForceSet</i> )

## 4.3.3 Generate category

This category of components includes the main (essential) components to control and apply the transformations as explained in section 4.2. For fast generations or designers in need of random networks, a fast policy transformation component is provided.

### 4.3.3.1 Construct Force Selection Rule

See 4.2.3.

### 4.3.3.2 Construct Node Placement Rule

See 4.2.4.

### 4.3.3.3 Construct Force Indeterminacies Rule

See 4.2.5.

### 4.3.3.4 Construct Transformation Policy

See 4.2.6.

### 4.3.3.5 Apply Transformation Policy

See 4.2.7.

### 4.3.3.6 Construct Transformation Policy (fast)

A way to construct a transformation policy very fast. The desired entropy rate is explicitly defined. The remaining 3 rules (*force selection*, *node placement* and *force indeterminacies*) are randomly generated with the help of a seed number.

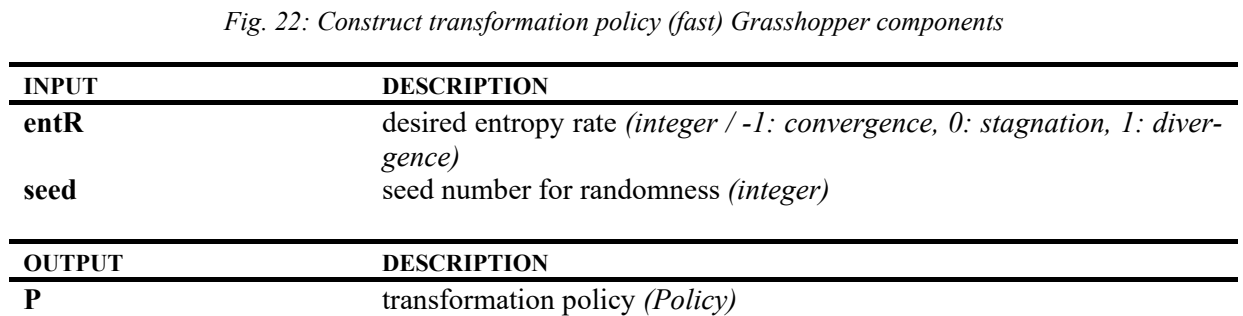
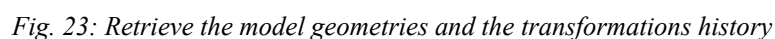


Fig. 22: Construct transformation policy (fast) Grasshopper components

INPUT	DESCRIPTION
entR	desired entropy rate ( <i>integer / -1: convergence, 0: stagnation, 1: divergence</i> )
seed	seed number for randomness ( <i>integer</i> )
OUTPUT	DESCRIPTION
P	transformation policy ( <i>Policy</i> )

#### 4.3.4 Model category

This category of components contains functionalities related to the model. The most important ones retrieve information such as: the geometric elements of the model, the transformation history, metric values etc.



#### 4.3.4.1 Construct Model

See 4.2.2.

#### 4.3.4.2 Select Force

See 154.2.3.

#### 4.3.4.3 Model Geometry

The specific elements of each generated model are previewed in the Rhinoceros import but their selection is not feasible. The following component exports each of the included elements (force, bars, nodes) as Grasshopper objects. Like this the designer has the chance to bake all or part of them.

INPUT	DESCRIPTION
<b>M</b>	transformed model ( <i>Model</i> )
<b>mirror plane</b>	(optional) the plane of symmetry in case that only half of a network is generated ( <i>Plane</i> )
<b>nodal radius</b>	circle radius that represents a node ( <i>double</i> )
<b>forces scale</b>	scaling factor for the forces ( <i>double</i> )
OUTPUT	DESCRIPTION
<b>forces ext</b>	directed line curve ( <i>Curve</i> )
<b>forces int</b>	directed line curve ( <i>Curve</i> )
<b>bars compression</b>	network bars that bear compressive axial forces ( <i>Curve</i> )
<b>bars tension</b>	network bars that bear tensile axial forces ( <i>Curve</i> )
<b>nodes</b>	network nodes as points, unless nodal radius > 0 ( <i>Point3d/Circle</i> )

#### 4.3.4.4 Model History

The component returns all the actual parameters used for the model transformation. The information either refers to a specific transformation step or to the entire transformative process. This information has particular value as it reveals occasions when the desired parameters -i.e., the node *P* location – or expected behaviors – i.e., the entropy rate - were not feasible and the algorithm replaced them with feasible ones. For example, when the desired entropy rate of convergence is not feasible, the algorithm automatically decreases the entropy rate to stagnation, or even to divergence, until the network transformation is feasible. This decrease becomes evident when plotting the applied entropy rate.

INPUT	DESCRIPTION
<b>M</b>	transformed model ( <i>Model</i> )
<b>i</b>	transformation step whose information is requested ( <i>int</i> )
OUTPUT	DESCRIPTION
<b>ForceS</b>	force(s) selection rule at <b>i</b> transformation step ( <i>ForceSelectionObj</i> )
<b>NodeP</b>	node placement rule at <b>i</b> transformation step ( <i>NodePlacementObj</i> )
<b>ForceI</b>	force indeterminacies rule at <b>i</b> transformation step ( <i>ForceIndeterObj</i> )
<b>entR</b>	applied entropy rate at <b>i</b> transformation step ( <i>string</i> )
<b>F</b>	set of interim forces selected at <b>i</b> transformation step ( <i>ForceSet</i> )
<b>P</b>	location of <i>P</i> node at <b>i</b> transformation step ( <i>Point3d</i> )
<b>N</b>	force magnitudes at <b>i</b> transformation step ( <i>double</i> )
<b>V</b>	force vectors at <b>i</b> transformation step ( <i>Vector3d</i> )

#### 4.3.4.5 Model Undo

As part of the design exploration process, it is essential to backtrack to previous states that satisfy criteria that the current state does not. Just specify the number the number of undo steps!

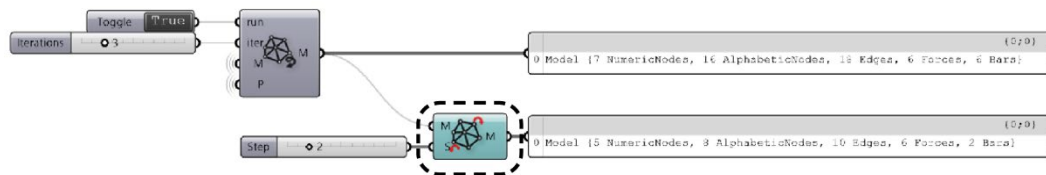


Fig. 24: Model undo to previous state

INPUT	DESCRIPTION
M	transformed model ( <i>Model</i> )
S	number of steps to undo ( <i>integer</i> )
OUTPUT	DESCRIPTION
M	transformed model ( <i>Model</i> )

#### 4.3.4.6 Model Metrics (spider graph)

The component plots a spider graph with eight of the most important metrics of the bar networks. Hence, the comparison between networks is easy. The input parameters are responsible for configuring the visual representation of the spider graph.

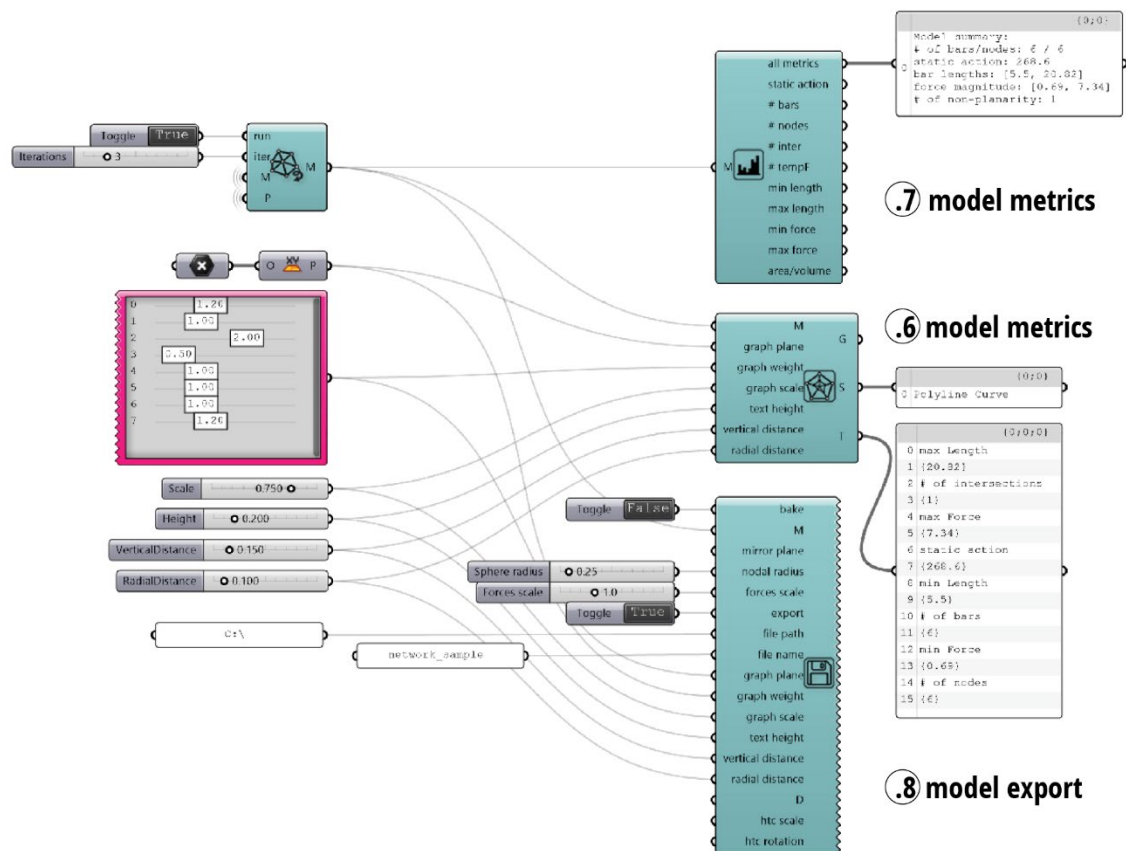


Fig. 25: A collection of Grasshopper components to collect metric values or export the model to Illustrator

INPUT	DESCRIPTION
<b>M</b>	transformed model ( <i>Model</i> )
<b>graph plane</b>	spider graph plane ( <i>Plane</i> )
<b>graph weight</b>	weight for each of the 8 metric values ( <i>GenePool</i> )
<b>graph scale</b>	spider graph scale ( <i>double</i> )
<b>text height</b>	legend text height ( <i>double</i> )
<b>vertical distance</b>	vertical distance between graph and legend titles ( <i>double</i> )
<b>radial distance</b>	spatial distance between graph and legend titles ( <i>double</i> )
OUTPUT	DESCRIPTION
<b>G</b>	spider graph ( <i>Curve</i> )
<b>S</b>	metrics polygon ( <i>Curve</i> )
<b>T</b>	spider graph legend ( <i>Text</i> )

#### 4.3.4.7 Model Metrics (text)

The component calculates and returns an overview of the most important metrics of the bar networks. Each of the output metrics is possible to be used as an optimization.

INPUT	DESCRIPTION
<b>M</b>	transformed model ( <i>Model</i> )
OUTPUT	DESCRIPTION
<b>all metrics</b>	description of all model metrics ( <i>string</i> )
<b>static action</b>	model static action ( <i>double</i> )
<b># bars</b>	number of network bars ( <i>int</i> )
<b># nodes</b>	number of network nodes ( <i>int</i> )
<b># inter</b>	number of intersecting bars ( <i>int</i> )
<b># tempF</b>	number of interim (temporary) forces present in the network ( <i>int</i> )
<b>min length</b>	shortest bar length in the network ( <i>double</i> )
<b>max length</b>	longest bar length in the network ( <i>double</i> )
<b>min force</b>	lowest axial developed stress in the network ( <i>double</i> )
<b>max force</b>	highest axial developed stress in the network ( <i>double</i> )

#### 4.3.4.8 Model Export to Illustrator

The generated model can be exported to Adobe Illustrator where highly illustrative visualizations can be processed.

INPUT	DESCRIPTION
<b>bake</b>	boolean operator that launches the bake to Rhino function ( <i>bool</i> )
<b>M</b>	model to export ( <i>Model</i> )
<b>mirror plane</b>	(optional) the plane of symmetry in case that only half of a network is generated ( <i>Plane</i> )
<b>nodal radius</b>	circle radius that represents a node ( <i>double</i> )
<b>forces scale</b>	scaling factor for the forces ( <i>double</i> )
<b>export</b>	
<b>file path</b>	
<b>file name</b>	
<b>graph plane</b>	(optional) spider graph plane ( <i>Plane</i> )
<b>graph weight</b>	weight for each of the 8 metric values ( <i>GenePool</i> )



<b>graph scale</b>	spider graph scale ( <i>double</i> )
<b>text height</b>	legend text height ( <i>double</i> )
<b>vertical distance</b>	vertical distance between graph and legend titles ( <i>double</i> )
<b>radial distance</b>	radial distance between graph and legend titles ( <i>double</i> )
<b>D</b>	(optional) domain to visualize ( <i>Domain</i> )
<b>htc scale</b>	hatch scale ( <i>double</i> )
<b>htc rotation</b>	hatch rotation in radians ( <i>double</i> )

### 4.3.5 Exploration category

This category contains components that have been adopted by *Biomorpher* (<https://github.com/johnharding/Biomorpher>) developed by Dr. John Harding and accordingly adapted to the needs of *Libra*. Particularly, *Biomorpher* is a Grasshopper plug-in that allows any parametric definition constructed in the same environment to be explored and optimized using interactive genetic algorithms. More precisely, it integrates any Grasshopper native object into the workflow. *Libra* generates its own objects and thus a few steps have been made to fuse the two frameworks is not straightforward. The modified *Biomorpher* component joins *Libra* under the name *DesignSpaceExplorer* (Fig. 27).

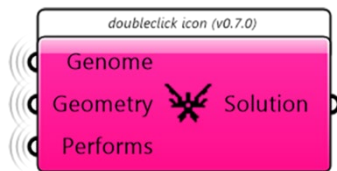


Fig. 26: Native *Biomorpher* component

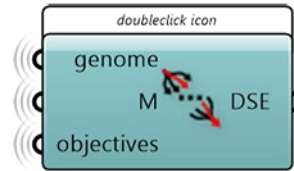


Fig. 27: *DesignSpaceExplorer* *Libra* component

The category contains more files, as those were adopted by *Biomorpher*. For more information how to unlock the magic of the arsenal offered by this fusion, go through the *Biomorpher* User Manual (<https://github.com/StructuralXplorationLab/Libra/tree/main/doc>).

# 5 Interactive design space exploration

## 5.1 Concept

The *Libra* designer sets up a sequence of policies that incrementally transforms an incomplete network of bars until the transition to a complete one. In the last decades, policy-based design processes (shape grammars and grammar rules) have been preferred for the exploration of the design domain. Their selection corresponds to the fact that they operate independently of known geometries. Still, at the end the process they only return a single result; potentially novel and creative but not guaranteed. Search algorithms and evolutionary computing are beneficial for massive exploration of the design space and capable of yielding multiple design candidates in a fast and automated way. Interactive Genetic algorithms (IGA) have been integrated into *Libra* to produce diverse, but still structurally relevant, networks more efficiently—i.e., lower static action, less intersecting bars etc. The stochastic nature of metaheuristic algorithms augments the breadth of design candidates but also returns optimized networks for various metrics.

## 5.2 Implementation

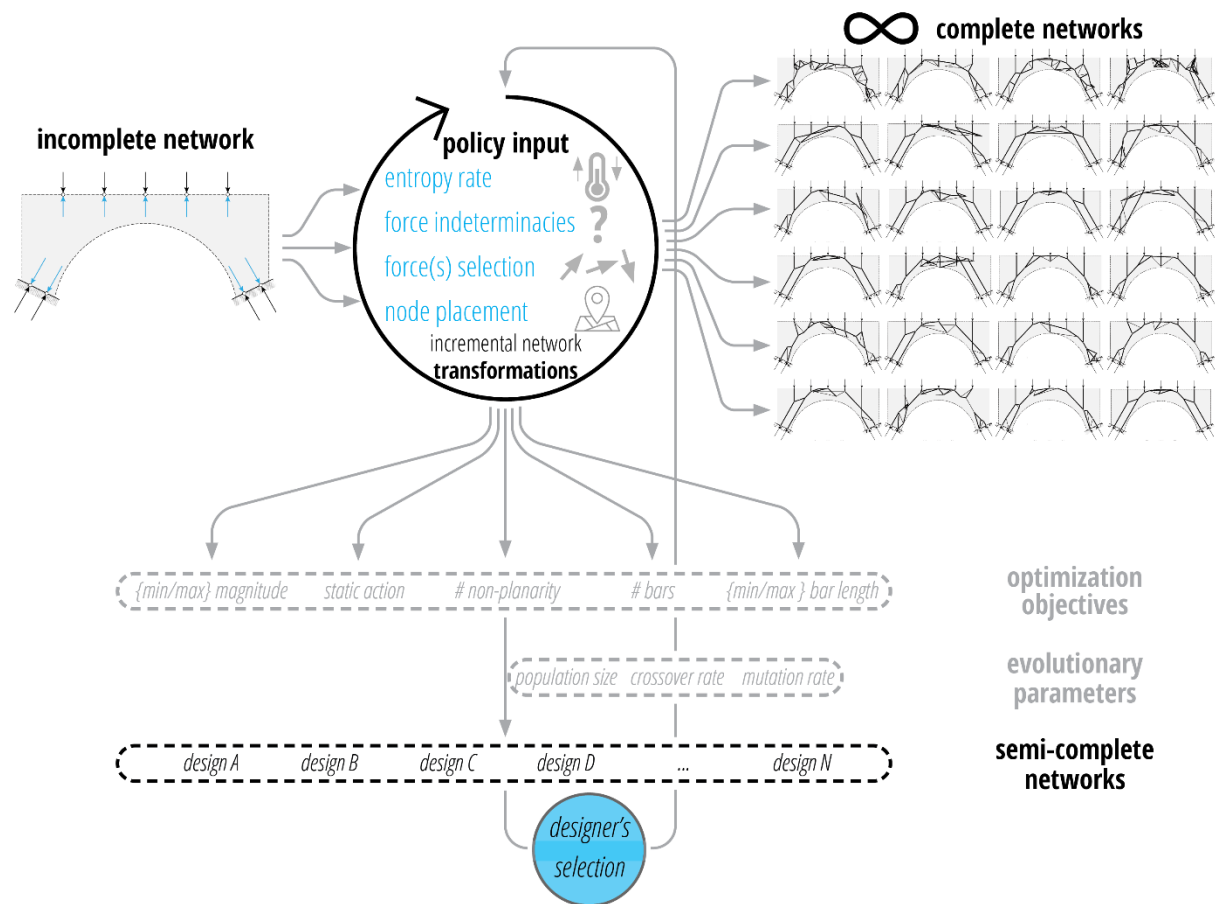


Fig. 28: Integration with interactive genetic algorithms

Fig. 28 provides a complete overview of the evolutionary process. The designer constructs a policy through a selection of four rules and transforms the network of bars for the course of a single, or multiple,

generation of the genetic algorithm. The evolutionary parameters and the optimization objectives, if any, can be updated at any moment. If the exploration aims at improved performance, the evolutionary algorithm tunes the rule parameters, in compliance with the optimization objectives. Each generation of the evolutionary algorithm plots a population of new incomplete bar networks in a specially designed interactive user interface. When the evolution stops, the designer chooses, if he/she wishes, the designs that satisfy subjective or other quantitative criteria.

# Appendix Rules

## Appendix 1 Entropy rate

-1	Convergence
0	Stagnation
1	Divergence

## Appendix 2 Force(s) selection rule

0	Random
1	ProximityToUserSelectedPoint
2	MinMagnitude
3	MaxMagnitude
4	Oldest
5	Newest
6	Closest
7	Furthest
8	ClosestToDomainCentroid
9	FurthestToDomainCentroid
10	ClosestToPtCloud
11	FurthestToPtCloud

## Appendix 3 Node placement rule

0	Random
1	UserSelectedParam
2	UserSelectedPt

## Appendix 4 Force indeterminacies rule

0	Random
1	UserDefinedMagnitude