EPFL

Thèse n° 9440

Policy-based Exploration of Equilibrium Representations (PEER): A topology grammar for generative conceptual structural design

Présentée le 6 avril 2022

Faculté de l'environnement naturel, architectural et construit
Laboratoire d'exploration structurale
Programme doctoral en génie civil et environnement

pour l'obtention du grade de Docteur ès Sciences

par

Ioannis MIRTSOPOULOS

Acceptée sur proposition du jury

Prof. A. Nussbaumer, président du jury
Prof. C. J. D. Fivet, directeur de thèse
Prof. C. Mueller, rapporteuse
Dr J. Harding, rapporteur
Dr J. Lee, rapporteur

École polytechnique fédérale de Lausanne

2022

Acknowledgements

I am indebted to Corentin, who shared with me his vision about innovative conceptual structural design and trusted in me the full development of a concept that started as an embryo during his postdoctoral research. Today we are both happy to witness its evolution into a complete design workflow through this thesis. Thank you for getting me involved with such an interesting topic. This dissertation would not have been possible without your enthusiasm, engagement, and guidance throughout. Your contribution is not limited to the scientific mentorship I enjoyed but also applies to the technical support. But above all, thank you for all those exchanges that inculcated new ideas and inspiration in me and encouraged me against all kind of challenges that emerged all along.

I would like to thank Prof. Alain Nussbaumer, Prof. Caitlin Mueller, Dr. John Harding, and Dr. Juney Lee, for accepting my invitation to join the jury for my doctoral oral exam. I really appreciate your willingness to review this dissertation and your wish to share your expertise. Especially, I want to thank Dr. John Harding for giving to all the scientific community the open-source gift of *Biomorpher*.

I am thankful to the recently grown large cozy group of Structural Xploration Lab. Specifically those who have been there since the very first day, Jan Brütting, Alex Muresan, and Nicolas Montagne who joined later. I am deeply grateful for all the discussion, advice, support, and humor that could break the silence after long hours of work late in the evening.

I would like to acknowledge my close, regardless of the distance, friends for their understanding, support and long international calls whenever I needed either life advises or to schedule relaxing holidays to rest. Especially, Naskas who has always been available for discussions about programming and many more, and Justine for her patience, love, support, and care.

Finally, I would like to thank my parents for their unconditional love, support and for instilling into me that the most valuable belonging I possess is in my mind. Thank you for making me who I am. Ευχαριστώ πολύ!

Preface/Foreword

From the roman arch to the thin shell, from the pitched roof to the Fink truss, or from the portico to bending-active assembly, the history of structural design is marked by a sequence of discoveries of new structural forms. For the trained eye of the structural designer, these forms are more than a geometric arrangement of matter. They 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. Thanks to the early developments by Stevin, Varignon, Rankine, Culmann and the likes, to the practical applications by Maillart, Gaudi, or Conzett, and to the recent developments by the graphic statics research community, the concept of static equilibrium in structures is one that can be reduced to a minimal, abstract model of vectors, nodes, and bars in compression or tension. In theory, the number of feasible arrangements of bars is infinite. Yet in practice, only a comparatively tiny, finite subset of arrangements is employed. Aren't we missing opportunities? Recent algorithmic developments, from procedural automation to optimization or machine learning, have the potential to allow designers to tame the infinite space of potential structural arrangements, to explore relevant solutions beyond those that result from optimization procedures, and eventually to discover new ones or ones that are more appropriate to given spatial, cultural, technical or economic contexts. In particular, could an intelligent machine actively collaborate with humans during the early stages of the structural design process? Ioannis' thesis successfully sets a new direction for developing such a fruitful human-machine dialog.

Corentin Fivet
Fribourg, December 2021

Abstract

Design exploration is a creative process that consists of the incremental generation of design candidates. Supported by digital means or not, the process handles the ill-structured nature of design and allows creativity to flourish through diversity of design candidates. This research proposes a design framework for the generation of variant bar network topologies in static equilibrium which facilitates the conceptual structural design exploration. Named PEER (Policy-based Exploration of Equilibrium Representations) the framework:

- incrementally grows and transforms networks of bars within specified geometric domains;
- maintains their static equilibrium at every intermediate transformation step;
- is built around a parametric policy – a course of actions - controlled by a choice of four explicit or abstract rules;
- is not constrained to precedent or recursing topologies and/or geometries;

Structural design space exploration is achieved through the generation of schematic, early-stage static equilibrium representations which are indicative, but not deemed optimized, force flows. As such, they are valuable as first design inspirations, prior to comprehensive structural analyses and form refinement. The transformation policy operates on given loads, is free from maximum valency limitations and unbound to specific topologies and geometries. On the contrary, the network's topology is not known a priori, but it is defined during the decision-making design process and constitutes the main output of the exploratory process.

PEER transforms interim networks of bars, whilst the network maintains static equilibrium at every transformation step. Precisely, each transformation results in the introduction of a new node, some bar elements in tension and/or compression and a few interim internal forces. The number of interim forces decreases while the number of bars increases and the entire process ends when no interim force exists anymore, which is always achievable due to the retention of static equilibrium throughout.

Contrary to other approaches and thanks to the incremental growth of topologies, PEER opens the generative design black box. While the process can be fully automated, it also lets the designer interrupt, redirect, or backtrack to previous transformations, at every intermediate step of the transformative process. Additional control is granted to the designer through the combinatorial choice of rules.

The genetic algorithm's stochastic nature matches well with the concept of exploration and the provision of multiple design alternatives. The design space exploration capability of the presented workflow is therefore further augmented by coupling it with interactive genetic algorithms, operating for the course of a single, or multiple, policy-based transformation(s). Via the interactive user interface, the designer selects the mutation and crossover parents based on aesthetic or performance criteria – though optimization is out of the research scope – and steers the exploration process according to personal preferences.

Policy-based incremental design and interactive genetic algorithms can provide designers with infinite alternative design candidates. Consequently, they efficiently boost design space exploration, and their combination ultimately provides a new design workflow for conceptual structural design. Its capacity to unveil numerous, unprecedented, maybe unexpected, but statically valid, structural forms is illustrated through planar and spatial application studies.

KEYWORDS

Design space exploration, structural design, static equilibrium, bar model, topology, conceptual design, generative grammars, evolutionary design, policy-based design.

Résumé

L'exploration conceptuelle est un processus créatif consistant en la génération incrémentale d'options conceptuelles. Assisté ou non de moyens numériques, le processus manipule la nature non-structurée de la conception, permettant à la créativité de s'épanouir via la diversité des options conceptuelles. Cette recherche présente un environnement pour l'exploration conceptuelle des structures. Nommé PEER – Policy-based Exploration of Equilibrium Representations –, l'environnement:

- fait croître et transforme, par incrémentation, des réseaux de barres dans des régions géométriques spécifiques;
- maintient leur équilibre statique à chaque étape intermédiaire de la transformation;
- est basé sur un 'contrat' paramétrique contrôlé par un choix de trois règles explicites ou abstraites ;
- n'est pas contraint par des topologies et/ou géométries antérieures ou récurrents.

L'exploration du champs de conception des structures est obtenu par la génération de représentations préliminaires, schématiques, d'équilibre statique qui sont des flux de forces donnés à titre indicatif et considérés comme non optimisés. En tant que telles, elles sont utiles comme premières inspirations de conception, avant toute analyse approfondie de la structure et son affinage formel. Le contrat de transformation, intervenant sur n'importe quelle charge donnée, est libre de toute limitation de valence maximale et n'est pas liée à des topologies et géométries spécifiques. Au contraire, la topologie du réseau n'est a priori pas connue, mais définie au cours de la prise de décision propre au processus de conception et constitue le principal résultat du processus exploratoire.

PEER transforme des « réseaux de barres » temporaires, tout en les maintenant, à chaque étape de transformation, en équilibre statique. Plus précisément, chaque transformation entraîne l'introduction d'un nouveau nœud, d'éléments de barre en tension et/ou en compression et de forces internes temporaires. Le nombre de forces temporaires diminue alors que le nombre de barres augmente et le processus se termine lorsqu'il n'y a plus de force temporaire, ce qui est toujours possible grâce au maintien constant de l'équilibre statique.

Du point de vue du concepteur/de la conceptrice, contrairement aux autres approches, celle-ci rend la 'black box' du processus de conception générative plus transparent. Alors que le processus peut être automatisé dans son entièreté, il accorde aussi au concepteur/à la conceptrice la possibilité d'interrompre, rediriger, ou revenir sur les transformations précédentes, à chaque étape intermédiaire du processus transformatif. Un contrôle supplémentaire est accordé au concepteur/à la conceptrice par le choix combiné de règles qui filtre de façon pratique le vaste espace de solutions des options conceptuelles.

La nature stochastique de l'algorithme génétique correspond parfaitement au principe d'exploration conceptuelle et à la mise à disposition de multiples alternatives conceptuelles. La capacité d'exploration de l'espace conceptuel de l'environnement présenté est ainsi d'autant accrue en le couplant avec des algorithmes génétiques interactifs, fonctionnant au cours d'une ou plusieurs transformations par contrat. Grâce à l'interface utilisateur/utilisatrice interactive, le concepteur/la conceptrice sélectionne les parents de l'opération de mutation et de croisement selon des critères d'esthétique ou de performance et dirige ainsi le processus d'exploration en fonction de préférences personnelles.

La conception incrémentale par contrat, les algorithmes génétiques interactifs et la visualisation de tous les phénotypes générés mettent à disposition du concepteur/de la conceptrice une quasi-infinité d'options conceptuelles. Par ailleurs, ils apportent une valeur ajoutée, exploitent efficacement l'exploration de l'espace de conception et finalement fournissent un processus de conception pour la phase

préliminaire de conception structurale. La capacité de ce processus à dévoiler un grand nombre de formes structurales inédites, peut-être inattendues, mais statiquement valides, est illustré tout du long par des application dans le plan et dans l'espace.

MOTS-CLÉS

Exploration de l'espace de conception, conception des structures, équilibre statique, modèle de barres, design conceptuel, grammaires génératives, conception évolutive, conception par contrat.

Table of contents

Notation

D_a	auxiliary domain
D_c	constructability domain
D_d	design domain
D_e	entropy rate domain
D_f	feasibility domain
E	entropy rate of a network
f_i	external force applied at node i
n_i	force magnitude (stress) in bar element i
R_F	force selection rule
R_I	force indeterminacies rule
R_N	node placement rule
t_i	interim force applied at node i
x_i	cartesian x-coordinate of node i
y_i	cartesian y-coordinate of node i
z_i	cartesian z-coordinate of node i

1 Introduction

This part introduces the research topic and the motivations behind this dissertation. The part also describes the relevance of the research within the context of structural design space exploration and provides an outline of the dissertation structure.

1.1 Research statement

1.1.1 Introduction

Often design concepts follow the emptiness of a blank page and the scrutiny of loosely defined criteria. Jotting primitive ideas down contextualizes the designer's inspiration and vaguely forms a design concept, usually via a sketchy depiction. Later, this design concept will be further developed until it evolves into a design candidate, namely a design concept that complies with the set requirements and is likely to be materialized into the real world, as a building, a structure, a painting, a car etc. The transition from a blank page to a design candidate raises lots of questions to be answered along the way, but often the answers are hardly evident. Design space exploration (DSE) is a creative process that facilitates and supports this transition. Through this process designers incrementally generate various design candidates and thus they build knowledge about the design challenge they are asked to handle and the answers they are expected to provide. Although design space exploration is performed within the bounds that various auxiliary constraints define, these constraints can become drivers of design, as Kilian says [1].

For structural design, namely the design of equilibrium-aware architectural forms, static equilibrium is the principal constraint that frames the design space exploration. Despite this strict constraint, an infinite number of non-resembling design candidates still exist in the design space. The selection of appropriate tools to generate candidates and trawl through all of them is crucial. Considering digital means improves overall the task efficiency and computation assists the effortless generation of alternative candidates.

Most conventional computational structural design tools depend on predefined geometries and hence impose lack of diversity and/or creativity. The design approach they operate on (conceptual, analytic etc.) and the design setup and input they require (constraints, level of detail, type of loading etc.) play a major role in their applicability, their suitability and their efficiency to the task of design space exploration. Thinking out of the box and in a creative way, computation has the potential to dodge premature design fixation and provoke design creativity, diversity and plurality. Eventually, delicate manipulation of computation is the key to unveil unprecedented, unconventional and unexpected, but statically valid, structural design candidates.

1.1.2 Thesis statement

This dissertation presents a computational framework for structural design which focuses on the generation of variant topologies that are in static equilibrium. In order to support designers to thoroughly explore the vast design space, the author provides a parametric design workflow that builds on policies

and rules rather than on explicit numerical input values. The computational framework targets at the exploration of diverse conceptual designs of structures, defined as early equilibrium representations.

Design inception

The transition from a blank page, where at most, the design domain and a set of external forces are defined, to a complete design concept that satisfies static equilibrium is a task that designers are daily expected to clear.

This research aims to support early design discussions by providing a computational framework that considers form and forces simultaneously through an alternative workflow. One of the research goals is the discovery of numerous design candidates of a conceptual structure-i.e., that satisfies static equilibrium-that serve architectural purposes as well. At an early design phase, these candidates can steer the round-table discussions. Later they can be further developed, and they can undergo extensive structural analyses along with form refinements. Explicitly, the research aims to:

- challenge the designers' creativity, i.e., by generating alternative design candidates
- facilitate the transition to a complete design concept, i.e., by considering static equilibrium all along and therefore speeding the architect-engineer exchange
- accelerate the generation of alternative design candidates, i.e., by providing a semi-automated generative design process which generates topologies

Fixation, evolution and backtracking

After putting a design concept on paper, laying it aside is not easy, especially for the author. Therefore, it is common to face design fixation when either no fresh ideas are brainstormed or when certain psychologic obsessions [2] restrain creativity to previous design concepts, favorable geometries etc. Escaping from such self-constraining aspects is exceptional. At the same time, design fixation does not allow the consideration of emerging constraints which are an integral part of architectural and structural design.

Emerging constraints require processes that allow for continuous transformation of a design candidate and that operate backwards to retrieve previous design stages. Version control systems turn out to be helpful for other tasks but not for design processes, as it is hard to recognize all the features (qualitative and quantitative) that signify the version one wants to bring back. Also, as a project progresses, there are often needs to update only a local portion of the design and keep the rest intact. In current parametric modeling processes, everything is linked and therefore it is hard to do so.

This research aims to provide a process that considers the constraints emergence and the potentially short lifetime of design concepts before they need to be updated again. Confronting the need to retrieve previous design stages, the research puts emphasis on backtracking mechanisms and handles the establishment of new design branches that share parts of the design concept but evolve differently.

Unconventional geometries and non-predefined topology

The static equilibrium condition that governs structural design is often handled by adapting long-studied and conventional structural geometries and types – e.g., warren howe, pratt trusses - or by adopting basic rules of thumb, such as triangulation. Such design approaches lead to geometries characterized by low complexity, which ensures easy construction, mass production and therefore higher efficiency and lower cost. Nevertheless, they restrain creativity and lead to resembling design candidates. Namely, the built environment gets overflown with the repetition of identical structures that lack unique identity.

This research aims to generate design candidates that are not bound to precedented geometries and do not resort to common topology patterns. Instead, the research explores unconventional design candidates and aims at boosting the creativity of during conceptual structural design. In other words, this research searches for novel structural design candidates that are visually appealing, have their own identity - from an aesthetics point of view - and still satisfy static equilibrium.

Diversity and plurality

On the one hand, design fixation tends to shrink the designers' productivity and creativity. Emerging constraints, on the other hand, make the production of alternative design candidates time demanding. Consequently, the breadth of design candidates is too small and the chances of discovering creative and unprecedented candidates are respectively low.

This research aims to accelerate the generation of structurally informed design candidates. The generation of numerous (theoretically infinite) candidates is also part of the research aims. It is believed that speed and plurality together will be key aspects to bring diversity and creativity in a design process.

Lack of computational framework for generative conceptual structural design

Despite the known challenges that design consists of and the complexity that static equilibrium constraints add, there currently exists no process that allows for structural design space exploration, through the generation of diverse topologies, and fully satisfies all the above-mentioned aspects.

The design approach and the ultimate intentions are both reflected on the dissertation title: *Policy-based Exploration of Equilibrium Representations (PEER): A computational framework for conceptual structural design*. The presented framework aims at addressing all the above-mentioned requirements. The acronym also expresses the wish to develop the human-machine peer collaboration that would allow the machine to temporarily take over the design process before the human takes back the lead, e.g., to impose new design criteria.

Fig. 1: Round table discussions as envisioned

1.1.3 Originality

The dissertation provides a computational framework that supports a novel bottom-up design workflow. Its unicity consists in the way that the designer controls the entire process and the way he/she interacts with it all along. Precisely, the input provision of implicit rules rather than numerical values, the fact that he/she does not rely on a predefined topology and the interactivity that allows for backtracking. The possibility to undo changes that partly influence a design candidate is a great asset for any design workflow, especially a conceptual design one. In particular, the dissertation aims to make the following contributions to the field of conceptual structural design space exploration.

Design policy and rules

The process of design space exploration is operated by a parametric, generic, transformative policy, described by rules, whose repeated application generates numerous structural design candidates. While similar rule-based approaches exist, they are bound to predefined topologies and/or geometries and only operate in two dimensions.

Node-by-node generation of topologies

This research focuses on incremental transformations of equilibrium representations that gradually grow larger. This growth is achieved through the addition of new nodes that imposes each transformation. The incremental evolution opens the black box of generative design contrary to other generative processes or form finding methods. Therefore, it brings more control to the entire process and allows backtracking to previous design states.

Construction and exploration of topologies

The incremental growth also gives the freedom to design the desired topology rather than following a strictly defined topology (e.g. ground structure) or a predefined topology (e.g., through a topologic graph). Each topology, indicated by design decisions, is subsequently constructed as an equilibrium structure. Exploration of the design space counts on diverse topologies as well as on diverse geometries of identical topologies as a result of the input (e.g. placement of new nodes, bar length constraints etc.)

Interactive exploration

Following the opening of the black box of generative structural design, mechanisms to grant additional control to the designer are investigated. Aiming at exploration, human intervention is necessary, whilst highly impactful. Hence, the node-by-node growth is either automated or controlled by the designer to steer the design candidates output from the exploration process towards personal preferences. Interactivity does not only facilitate exploration but also allows for direct retrieval of previous states of the design candidates.

1.1.4 Relevance

The main aspect that makes the dissertation outcome valuable in the realm of conceptual structural design and design exploration is the discovery of unconventional structural forms through exploration. Design space exploration triggers the designer's imagination and thus increases the chances of discovering new structural forms. Mainly, this dissertation provides a framework that is capable of generating a huge amount of design candidates. The number of the design candidates is not possible to be quantified but is extremely larger than the number of design candidates a human being can conceive and later visualize with the help of analogue or computerized means. Through the plurality of design candidates, design diversity is achievable. Hence the discovery of diverse and ultimately unconventional structural forms is feasible.

1.1.5 Organization of the dissertation

This section outlines the structure and organization of this dissertation. The dissertation is divided into four main parts: *Introduction, Policy-based Exploration of Equilibrium Representations, Interactive design space exploration* and *Discussions*. The chapters that they consist of and their respective content is briefly provided below.

Part I - Introduction

Chapter 1: Research statement

This chapter introduces the topic, exemplifies why this research is conducted and underlines its originality and relevance. Linking this research with the state-of-the-art is included in the following chapter.

Chapter 2:Literature review

This chapter includes the critical literature review and thoroughly explains the research gap that this dissertation investigates. The reader is introduced to general concepts and terminology, which though prominent in different fields relate to the development of bespoke design approaches for the design space exploration. Review of ongoing or past relevant research is also provided. Finally, an overview of the foreseen challenges within these fields, inspirations sourcing from them, as well as opportunities that this dissertation responds to, are provided.

Chapter 3: Opportunities

This chapter makes the link between the presented literature review and the developed computational framework and introduces the reader to the next part of the dissertation.

Part II – Conceptualization of Policy-based Exploration of Equilibrium Representations (PEER)

The second part of the thesis answers how the conducted research fills up the existing research gaps. PEER supports a new design workflow. The conceptual inception behind this approach, its structure as a procedural workflow, and the notion of a single transformation step are discussed in the respective chapters.

Chapter 1: Parameterization approach

This chapter presents the theoretical background of the design approach. It starts with the contextualization of the atypical style of parametric design that is presented (policy-based and incremental) and later it explains, through comparisons with other approaches, the author's reasoning to further investigate the specific style.

Chapter 2: Introduced terminology

This chapter introduces several terms that are extensively used by PEER framework and their understanding is crucial. Among other term definitions, the chapter presents a primary comparison between the notions of rules and policies and provides their interpretation within this dissertation scope.

Chapter 3: Intentions and scope

This chapter maps the author's intentions and frames the research scope with regards to the expected outcome of the generative process. The recommended applications of the PEER framework are provided in Part 5 and chapter 5.3.

Chapter 4: Assumptions

In this chapter the author declares the assumptions around the conducted research. The included assumptions relate to the policy definition and the mechanical behavior of the generated networks of bars.

Chapter 5: Meta-methodology

This chapter provides a narrative overview of the methodology throughout the computational development of the PEER framework. Although this description is not framed in a technical way, it underlines the challenges and unexpected tasks that had to be tackled. Hence, it justifies the final path that has been followed.

Chapter 6: Methodology

This chapter outlines a policy-based approach for the transformation of a set of force vectors (sub-network) into a network of bars in compression and/or tension. The chapter focuses on the input and output values and briefly illustrates the intermediate stages that rule a single transformation step. The following part provides detailed description of the methodology through the perspective of its implementation.

Part III – Implementation of Policy-based Exploration of Equilibrium Representations (PEER)

The third part of the thesis discusses at length how the concept of a single transformation step of a few force vectors translates into a design approach that generates structures in static equilibrium.

Chapter 1: Computational implementation

This chapter illustrates the algorithmic workflow and delves into the algorithmic operations that support the transformation of a network of bars. The policy parameters are clearly indicated and the algorithms that define the solution space of allowed transformations are explained. The possibility of adding constraints, as well as their impact on the solution space is discussed too. The chapter mainly focuses on the development of the complete toolkit that translates theory into a real computational tool to manipulate interim networks of bars.

Chapter 2: Illustrated behaviors

This chapter is ancillary to chapters 2.6 and 1.1 and illustrates different behaviors of networks of bars grown as a result of different parameters. Primitive examples of networks of bars are chosen whereas emphasis is put on the growth behavior rather than the geometry outcome.

Chapter 3: Dissemination

This chapter acts as a manual of the developed toolkit, named *Libra*. The list of functionalities as well the setup of the transformative workflow is explained step-by-step.

Chapter 4: Application studies

This chapter includes application studies made with the developed tool. They are presented in chronological order and showcase the increased level of control as new functionalities are gradually added to the computational tool. Symmetric and asymmetric structures are generated within convex and non-convex design domains through the presented policy-based design approach. The chapter also proves the possibility to extend the design approach in three dimensions. An example of a spatial network in static equilibrium is exploited for this reason. Examples of diverse design candidates for the same design brief are eventually provided.

Part IV – Interactive design space exploration

The fourth part of the thesis focuses on the fusion of PEER with interactive genetic algorithms (IGA). Particularly, it focuses on the mass-generation of design candidates through evolutionary algorithms and the human involvement in the process. The way large amounts of design candidates contribute to the broad exploration of the design space, as well as reasoning why interactivity facilitates the process are discussed. The part first explains the added value that evolutionary computing brings to the process of design space exploration and justifies the selection of the existing framework of *Biomorpher* [3] that complies with the PEER requirements as those defined by the author. After, it describes how *Biomorpher* operates. The part ends with the integration of IGA into the PEER framework and its application to design briefs.

Chapter 1: Concept

This chapter explains why search algorithms are beneficial for the PEER framework and lists the requirements that the existing framework must satisfy in order to be fused with PEER.

Chapter 2: Framework selection

The chapter provides a comparison among the existing frameworks that operate on search algorithms within the parametric environment of Rhinoceros Grasshopper and argues about the final selection of *Biomorpher* as a framework to integrate into the PEER framework.

Chapter 3: Methodology

This chapter presents the workflow and the functionalities of *Biomorpher* and highlights the interactive control granted to the designer. The sequence of decisions, the setup for optimized solutions and backtracking to previous generations are presented too.

Chapter 4: Implementation

This chapter explains the features that support the interactive design space exploration. The chapter also explains the necessary changes that had to be made for the fusion between *Biomorpher* and *Libra*. Finally, the complete workflow is demonstrated.

Chapter 5: Application studies

This chapter highlights through application studies, the capacity of the policy-based design approach to thoroughly explore the design space after its fusion with interactive genetic algorithms. The full range of design candidates, as well as the step-by-step selection of intermediate semi-complete networks until the final design are provided.

Part V - Discussions

The last part of this dissertation outlines the contributions in the field of conceptual structural design and design space exploration. The part also describes the short- and long-term additions that are expected to bring robustness to the design toolkit. The type of applications that is capable, or not, of designing and those that will be possible in the near future are also included. The part ends with thoughts around the human and machine synergy and its potential for the specific workflow and the field of grammar-inspired design.

Chapter 1: Contributions

This chapters highlights all contributions that this dissertation brings to the field of design space exploration and hence indicates how PEER framework differs from the state-of-the-art alternatives.

Chapter 2: Future work

This chapter lists the future work that the author recognizes as necessary in order to unlock the full potential of the PEER framework. Some of the improvements relate to additional functionalities and others its implementation. The former will grant more control to the designer. The latter will decrease the computing time of many operations and will make the process of exploration faster and more responsive.

Chapter 3: Applications and opportunities; capabilities and incapabilities

This chapter describes the type of applications that PEER is ready to operate on and those that do not fit with its concept of generating equilibrium representations.

Chapter 4: Conclusions

This chapter includes the author's last thoughts and visions for the future with regards to automated generation of design candidates for the conceptual structural design space exploration.

1.2 Literature review

This section starts with the provision of the necessary vocabulary that is extensively used in this dissertation. After, it presents a literature review that explains the wicked nature of design and gradually justifies how design exploration helps designers to tackle design's immense challenges. Later, the section describes how design exploration and conceptual design are related in the context of structural design. The section continues with a brief explanation of shape grammars and grammar rules and how they were exploited for design exploration. In the end, it describes the approaches and the fields that are believed to be beneficial for the specific objectives described before, in section 1.1.2.

1.2.1 General terminology

The author's interpretation of specific terms in the context of this dissertation is provided below.

Conceptual (early) design stage

This corresponds to the inception stage of a design candidate and initiates the transition from a blank page to a primitive design candidate. Traditionally, it is materialized through sketches, hand drawings, and recently through computerized means. In both cases it is characterized by abstraction, whilst outlines the constraints and requirements as expressed by clients or other sources. This primitive material usually represents the core of the round-table discussions among multiple actors. Practically, it comprises the "most creative, most fast-paced and most ill-defined stage" of any project [4]. Although the list of definitions of conceptual design stage is not comprehensive, what explains the weight and the importance given to this stage is demonstrated in Fig. 2. This stage frames the moment during the project when there is most design freedom. At the same time, impactful decisions taken during this stage steer the entire project and influence the overall cost [5]. Considering that freedom is at its peak and cost is of design changes is still low, the conceptual design stage is highly recommended for exploration. Fig. 2 explicitly visualizes the chronological scope of the dissertation and the moment in the chronology of design that the use of PEER framework is suggested.

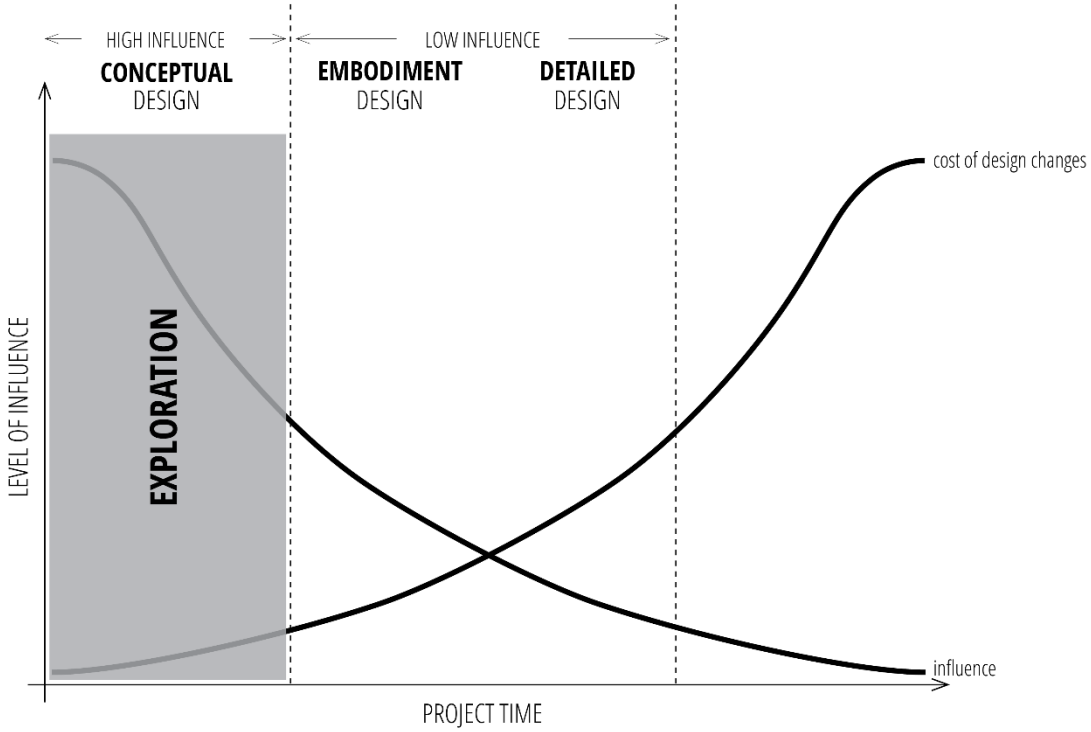


Fig. 2: Level of influence on project costs. Image adjusted from [5,6]

Topology

The birth of topology dates back to 1735, when Leonahard Euler questioned the possibility of defining an itinerary that crosses each of Königsberg's seven bridges to the island in the city center exactly once. He visualized this problem with the help of a diagram, where each riverbank of departure and arrival is denoted by a node and the bridges among them are denoted as scaleless segments. This network helped him to confirm the problem as impossible to solve [7]. The diagrammatic or graphic representation of interrelationships among nodes (of any entity) has been widely used in architecture. Typical examples visualize spatial relationships of rooms within buildings [8]. What is interesting and contained in the scope of this dissertation is that topological relationships are represented by scaleless graphs. Focus is given on the described connectivity, but no value is enclosed on the nodes specific cartesian location.

Computerized

A task is considered computerized if its partial or full completion implies the involvement of computers–e.g., Computer Aided Design (CAD). The involvement of computers is not crucial for the accomplishment of the specific task, as no computational logic is required. Rather, the task can be completed with analogue means and labor. In architectural engineering, typical examples include the drawings drafting and representation procedures or tasks of assessment and optimization. Computerized approaches often accelerate the time required for a task, especially those that involve repetitive design routines–e.g., array functions in CAD–and occasionally provide the possibility of backtracking–e.g., undo a function.

Computational

A task is considered *computable* if a finite sequence of instructions exists such that when followed results in the completion of a task [9]. This is the definition the operation of personal computers (PC) is based on. The reason why computers have become ubiquitous in architecture nowadays lies in their capacity of controlling and feeding information in loops, finally converting a computable function into a design or morphogenetic operation. The technological advances have allowed the integration of

numerous computational methods into many design and analysis approaches which are ruled by computational logic.

1.2.2 Design, a wicked “problem”

Design is a wicked “problem” that makes it ill-structured [10]. So, deterministic methods are not applicable to handle it. Additionally, it is characterized by emerging requirements/constraints and contradictory design pathways. These particularities affect the *inception* – “how designers escape from a blank page challenge?” – and the *evaluation* of a design – “is the design good enough?”

Designers cannot always tame design complexities through mathematical models, fact that makes design more of a *challenge* than a *problem*. As “one cannot first understand (design), then solve (it)" [11], designers must first gain knowledge about it. Designers compensate for their lack of knowledge through creative processes, one of them being *design exploration*, which frames the incremental generation of design candidates. The solution space where valid design candidates emerge from is named *design space* and its size is vast. During the process of *design space exploration*, designers often build on their own experience and browse just a tiny fraction of all prospective design candidates. Tendencies for premature design fixation typically result in the generation of resembling designs showcasing a lack of diversity and/or creativity [12]. The hypothesis is that extensive design space exploration unveils unprecedented structural geometries, fights design fixation, provokes creativity and facilitates impactful decision making. Chronologically speaking, design space exploration is favorable to be practiced at the early- (conceptual) stage as visualized by Paulson and others [5,6,13].

The wicked nature, the absence of quantitative criteria and the lack of global optimality make it hard to evaluate a design. Namely, it is often difficult to rationalize the underlying intentions behind specific design decisions. Thus, design, as an outcome, is often taken “as is”.

1.2.3 Design exploration

One could question “Why preferring *this* design over *that*? Have other alternatives been considered for the same assignment?” This is a valid question that argues the invested effort for in-depth exploration of variant design candidates. Questioning could continue. “If so, were they better or worse?”. Better both in terms of aesthetics, which is a subjective argument, and/or in terms of performance, which is an objective one. Answering these two questions is not an easy task for any designer.

1.2.3.1 Search algorithms; solve and optimize

Computer scientists have long studied the complexity of similar problems and came up with many efficient algorithms which can solve them and efficiently answer the previous questions. They have developed heuristic methods and techniques that embed experiences and rules of thumb to assist the algorithms to find the way towards better solutions after several iterations. The provided human knowledge, not necessarily expressed in a mathematical way, implies that the solution is known in advance, at least roughly, and thus the algorithms return predictable results. It is evident that heuristic algorithms are not appropriate for design exploration, unless specific forms are sought after (e.g., minimal, funicular etc.). Metaheuristic algorithms on the other hand, have been developed to handle problems that require searching through a vast solution space. The lack of prior knowledge and the large size of the solution space of unstructured (or ill-structured) problems, like design, make such algorithms a perfect fit for design exploration.

In general, metaheuristics algorithms exploit search mechanisms to discover feasible solutions that satisfy certain requirements. Search is primarily based on randomness, which increases the exploratory capacity of the algorithms, but the presence, if any, of quantitative objectives steers the same

mechanisms to the incremental discovery of better performing solutions. The process operates within a predefined search space, explicitly bounded, where numerous, almost infinite, alternative solutions are circumscribed. After a few iterations, a single feasible candidate is returned. Its value depends on the size of the:

- search space
- problem itself

Thus, the size of the above features severely impacts the computing speed.

Combining constraints with objectives redefines such problems as optimization problems. In the case of design, constraints stand for facts that cannot be overcome (e.g., boundaries of design domain). Their definition does not relate to specific objectives, but their introduction is likely to increase the complexity of a problem. For architectural design, indicative quantitative objectives – i.e., those can be included in the definition of an optimization problem - include the structural, acoustic, thermal performance minimization/maximization.

In optimization, the set of all fitness values [14] calculated for all feasible solutions with response to the quantitative objectives defines the so-called fitness landscape. Each search space is associated with a respective fitness landscape that represents worse, identical, or better solutions. Optimization, through deterministic or stochastic means, stands for the discovery of the outperforming solutions lying on the fitness landscape. Their location on the fitness landscape in comparison with other solutions makes them optimum; locally or globally. The efficiency to discover global or local optimality is subject to the choice of the appropriate algorithm, in response to the problem type, and the available time. Concretely, deterministic algorithms are suitable for structured problems, they rely on mathematics and thus guarantee global optimum solutions at the expense of long computations. Heuristics accelerate them. On the other hand, metaheuristic (stochastic), algorithms are suitable for unstructured problems, they rely on randomness and thus cannot guarantee global optimum solutions, but they are generally faster. This comparison best fits with single objective optimization but respective analogies are also observed for multi-objective optimization.

The suitability of metaheuristic (stochastic) algorithms for exploration and optimization, if necessary, of design candidates was highlighted before. The same argument has been repetitively supported by many researchers [15]. Many of the metaheuristic algorithms (Genetic Algorithms, Particle Swarm Optimization, Ant Colony Optimization) are biology-inspired and simulate life's main problem of evolution by natural selection, per Darwin who linked the evolving process of species to their adaptation to their environment. The following section will provide a brief introduction to evolutionary computing and will delve into the believed-to-be-dominant stochastic approach of Genetic Algorithms (GA). Their advantages and disadvantages in the field of design will be discussed prior to a section dedicated to interaction and its integration with evolutionary computing.

1.2.3.2 Evolutionary computing and genetic algorithms

Evolutionary computing represents a set of stochastic search algorithms and optimization algorithms. First applications of evolutionary computing were evident in the mid-1950s, when George Box [16] introduced parametric variability and natural selection to process plant operations. His work focused on building the end product but also to acquire information on how to improve the process. This reference comes earlier than the famous work of John Holland and his book [17] that established the theory of genetic algorithms, the renowned and prominent representatives of evolutionary computing. A few years later, David's Goldberg publication of *Genetic Algorithms in search, optimization and machine learning*

[18], provided several studies that confirmed the applicability of evolutionary computation for search, exploration and optimization.

Genetic algorithms are *"[…] search algorithms based on the mechanics of natural selection and natural genetics"*. Potential solutions (*individuals*) are defined by a finite list (*chromosome*) of discrete values (*genes*). A new generation of individuals is bred at each iteration of the algorithm. In the most basic implementation, each new generation of *children* is created after combining (*crossover*) or altering (*mutation*) the chromosomes of *parents*. Those parents have been selected either randomly or for their fitness (as measured by a given function) according to preset probabilities. Random selections broaden the scope of the search (*exploration*) and elitist selections narrow it down to the fittest solutions (*exploitation*). The balanced combination of both strategies is expected to lead to the desired solution(s).

A simplified flowchart explaining the process follows:

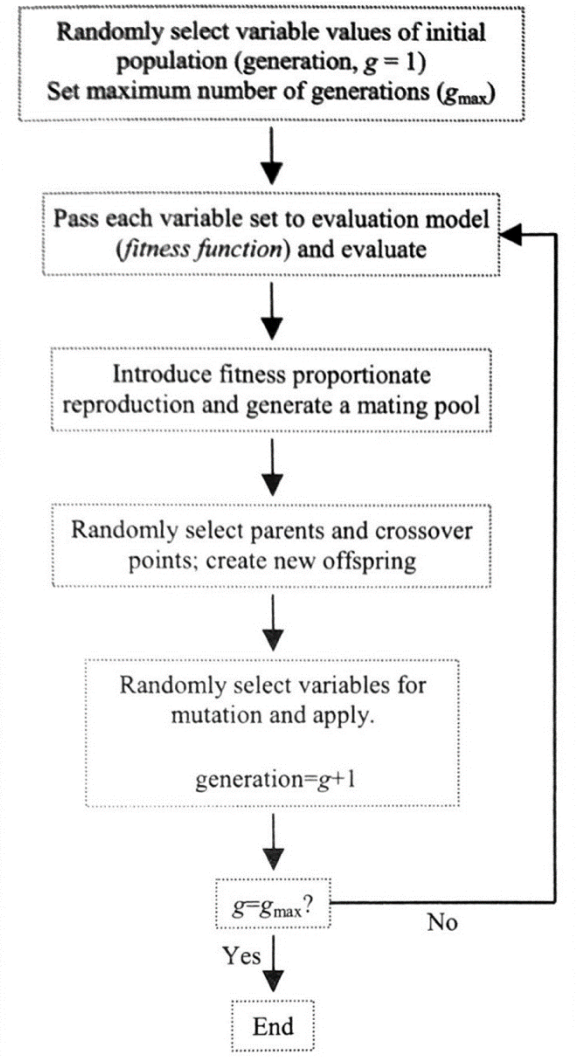


Fig. 3: The simplest GA procedure [6]

Besides genetic algorithms, other renowned implementations of evolutionary computing include: Genetic Programming [19], Evolution Strategies [20], Evolutionary Programming [21] etc., and were first introduced between 1960 and early 1960s. At that time, metaheuristic algorithms were often mentioned as artificial intelligence methods since the methods involved mimicking human problems with solving behavior from learning lessons [22]. To avoid this misinterpretation, consider that most of them are

biology-inspired, they source from Darwinian theories and they operate in a stochastic way that implies randomness rather than intelligence.

Advantages of evolutionary computing

According to Parmee [6] evolutionary computing does not provide an overall panacea for efficient search, exploration and optimization as opposed to most deterministic approaches. Operating based on biological evolution, evolutionary computing is more of a source of inspiration, rather than a phenomenon to be accurately modelled [23]. But, for early stages of design it provides relatively generic optimization procedures that allow for less rigorous formalization of the problem.

In his book *Evolutionary and adaptive computing in engineering design*, Parmee outlines a convincing list of attributes that have established stochastic search techniques relevant to engineering design processes. Those that overlap with the beliefs of the dissertation author have been included below as a conclusive list of arguments followed by co-relations with the current research project:

- require little, if any, a priori knowledge of the search environment; they operate blindly within the predefined search domain, which, in the scope of this research, overlaps with the design domain
- have excellent exploratory capabilities and initially randomly populate the design space with trial solutions; this facilitates the automated, fast generation of variant and valid networks through the random manipulation of the input parameters
- can handle high dimensionality; the current list of variable parameters for the generation of a network is low but the potential development of a very specific policy, as opposed to the current generic one, will retain its compatibility with stochastic algorithms
- are robust across a wide range of problem class; the policy-based design approach could not be compatible with deterministic algorithms because: a. it is design-oriented and thus addresses an unstructured problem and b. policy approaches have an abstraction that can hardly be quantified, as opposed to numerical approaches. For example, in the latter case, the position of a network node can only be described through its coordinates (spatially) or the incidence matrix (topologically), which in both cases is explicit;
- can avoid local optima; in exploration, optimality is not a priority. Nevertheless, the capacity to "climb out" of a local minimum reduces resembling and equally performing solutions.

The above list highlights the exploratory power and versatility of evolutionary computing, promises diversity but ignores the results validity and reliability. In another book of his, David Goldberg [24, p.18-19] discusses the genetic algorithms capacity to provide solutions to complex systems, seemingly more challenging than design:

"Imagine you are waiting in Chicago O'Hare airport for your airplane back home after a hectic GEC theory workshop. The gate announcement is made, and you board the aircraft. […] The whine of the GA-designed gas turbines grows louder, turning into the familiar dull roar, and you feel the aircraft gathering speed as it rolls down the runway. Suddenly, at the moment of incipient takeoff, a thought crosses the threshold from your unconscious to your conscious mind, and you realize the following chilling fact: no one has ever proven mathematically that an airplane can fly!"

He so argues that an airplane is not ineffective or unsafe because a formal mathematical proof of flight does not exist. Goldberg first used GA for structural optimization by solving a 10-bar truss structure optimization problem [25]. Since then, GA and numerous variations of them have been implemented to efficiently solve several engineering problems. Examples of GA are found in computational morphogenesis [26], architectural design [27,28], structural design [13,29] and many other fields. Apparently,

[15] share similar thoughts and has already acknowledged the assets of evolutionary computing techniques, when exploited in optimization tools for parametric modeling.

Disadvantages of evolutionary computing

Whatever nature does, does it better [24]. And GA as natural systems have proven their robustness and efficiency through countless applications. For examples found in nature, evolution is comprehensive and transparent. Those coming from the field of engineering do not have the same transparency when handled by evolutionary processes. Recalling the return of a single solution as nearly optimum (globally or locally) and the blind sequential generation and selection of variant populations features evolutionary computing as a black box hermetically closed to the user. The single algorithmic outcome is thoughtlessly accepted and directly integrated into workflows while tens or hundreds of alternative candidates that were generated for the same requirements and constraints are never considered. For population-based algorithms, the solutions lifetime in most of the cases is short, even limited to a single generation, if their performance with response to the objective function has not been high or competitive enough. The concept of survival of the fittest does not offer the chance for performance improvement to the full body of genotypes. Crossover and mutation rates blindly kill these alternatives at the expense of exploitation. Consequently, their generation and existence as potential solutions have never been recorded or performance evaluated.

In design, the introvert mechanisms of evolutionary computing overshadow the designers' contribution towards the final result. Typically, extroversion, expressed as interactivity, is limited to the moment of formulating the problem, namely while defining the input and output parameters. The rest of the process is a black box, unless the designer interacts with the process.

1.2.3.3 Interactive evolution

Zoologist Dawkins opened up this black box by a solution that he presented in the book *The blink watchmaker: Why the evidence of evolution reveals a universe without design* [30]. In the homonymous documentary he demonstrated how custom human decisions redefine the directions of the originally random changes in the evolution of biomorphs. The concise implementation that he presented has become the prototype for many interactive evolutionary systems since then.

In design, it is very rare to make decisions that rely on a single criterion. There are qualitative objectives (i.e., aesthetics) and perhaps quantitative ones that are often contradictory. In such cases decisions are made as a trade-off between visual result and performance [6]. In traditional stochastic approaches, no possibility to visually assess the design candidates is offered, as the process evaluates quantitative objectives in an automated way, independently of the designer. Interactive evolution can be thought as an alternative evolutionary technique whose objective function is replaced by a human user [31].

Advantages of interactive evolution

Stochastic algorithms heavily rely on randomness. The concept of interactive evolution is to involve the human user in the selection-variation loop of the evolutionary algorithms [32] and provide visibility over the ongoing evolution. Rather than waiting pathetically for the final outcome, interactivity upgrades the role of the user and makes him/her active in the search process. The user always has the choice to take or leave the intermediate results but is also obliged to decide how the process will evolve. The ultimate benefit of interactivity is that the output does not source solely from the calculation of the input parameters, namely the structure of the underlying algorithm, but from a forging process. In other words, a consequence of modifications happening during the algorithm execution. Specifically:

- Human intervention into the search and selection process allows faster convergence onto the most promising regions of the fitness landscape. Thus, the human knowledge contributes as some sort of heuristics to the process.
- Human intervention detaches the evolutionary from purely performance-driven results. The modifications are imposed by the user's subjective decision-making and steer the outcome accordingly.

Occasionally, interactive evolution is supported by storing past generations in databases. In those cases, the designer is free to retrieve past solutions and further explore the solution space start from a new base (parent). In structural design, this approach has been extensively used by ParaGen, developed by Peter von Buelow, as one of the first interactive evolutionary platforms to explore variant structural design candidates [29].

The flexibility offered by interactive evolution goes beyond the selection-variation processes. Users are free to modify the input parameters as well as the optimization objectives interactively. Such features highlight interactive evolution as a perfect match for unstructured problems dominated by emerging requirements/constraints and contradictory objectives.

Disadvantages of interactive evolution

The selection of desired solution candidates is a tedious process that users are asked to do. Due to psychological constraints, humans can only select from a small set of choices [32]. In other words, the user gets psychologically tired, and the interactive evolution cannot be continued after many generations [31]. Evolutionary generations include tens or hundreds of candidates which cannot be visualized in one screen. Humans can memorize only a fraction of them and are able to make comparisons before selected the preferred candidate. The combination of the two increases the operation time and the overall fatigue. Potential reduction of the generated candidates would result in comparatively narrower solution space exploration and thus it is not considered as an option. Clustering resembling results into families of similar solutions represented by a single result, significantly reduces the size of the solution candidates as well as the workload of the designer. Alternatively, computer scientists proceeded with the development of cluster-oriented genetic algorithms (COGA) [33,34]. A more progressive approach suggested by Terano and Ishino is to tune the parameters of the objective function with the help of simple reinforcement learning [35].

1.2.3.4 Interactive evolutionary design

In the fields of architectural and structural design, interactivity has been intended since more than a century. The well documented traditional form-finding methods used by Antonio Gaudi, Heinz Isler, Ove Arup, Frei Otto evidently prove the necessity for interactivity between the model-and the designer [36]. In the digital era of the same fields, the experimental space where models at different scales were traditionally built has been replaced by parametric digital environments, like Rhinoceros Grasshopper and Autodesk Dynamo. The latter have gained ground over the laborious physical models due to the ease of use, the flexibility and speed to edit designs with less effort. The updated arsenal of computational tools has made a lot of designers to speak of interactive design approaches.

A valid question refers to the type of interactivity expected and its chronological identity within the design process. Interactivity corelates with time and duration. Duration in the evolution of a design candidate might refer to the project life, from the conceptual phase to its construction, which lasts years or refer to the evolutionary transformation within a generative process, which lasts (milli-)seconds.

Moreover, interactivity is likely to be offered at different moments:

- at **declaration time**; humans do not intervene with the algorithmic execution;
- during **execution time**; humans actively intervene with the algorithmic execution;
- at **any time**

The moment of interaction and the type of impact it has on the process creates different interpretations of the term. Specifically, interactivity at *declaration time* refers to the moment that input values are provided to the parameters of the algorithm. The possibility to variate these values impacts the final result, but the algorithmic procedures remain unaffected. Interactivity during the *execution time* is an integral part of the interactive evolutionary process and the choices made affect the algorithmic workflow. This implies a higher level of involvement and unveils some sort of dependency between the definition of the external input variables and the execution of internal functionalities. Interactivity at *any time* is the addition of the former two.

Allowing human interaction throughout the process has allowed evolutionary computation to act as exploratory tool, as opposed to merely being an optimizer [37]. According to the abovementioned distinction, interactive evolutionary computation allows designers to enhance their creativity and boost design space exploration. Explicit design choices implicitly target areas of the design space that designers find more interesting and thus accelerate convergence on aesthetically pleasing designs [38].

The notion of convergence has variable applications. It originates from computer science and refers to the proximity of an algorithm to approach an expected result (heuristics). In engineering, it is often found in combination with form-finding methods that are expected to result in forms known in advance. In design space exploration, the term relates with creativity and the notion of convergent, or divergent, thinking [39]. According to [40], interactive genetic algorithms in design can help enhance creativity because they facilitate both convergent and divergent thinking; analysis and refining of an idea towards a single best solution, and development of new ideas that lead to new design candidates respectively.

The capacity of interactive evolutionary design to generate multiple design candidates, along with the interactive human selection that it offers, has featured it as the most common exploration tool used in the realm of digital and parametric design. The implementation of a supplementary user interface (UI), within parametric design platforms, is necessary to support interaction. Multiple researchers (von Buelow, Mueller, Harding among others) have acknowledged the benefits of interactive evolutionary design and have developed the necessary tools, either for educational purposes or for free use by any interested designers. Hence some approaches have been restrained to the realm of a class or an institute. Other researchers have freely distributed their contributions to the field and thus the vast society of computational designers can take advantage of them.

In the scope of this research, the functionality of design space exploration tools has more interest than the applications that they have been used for and therefore the comparison and the review relate to this aspect.

CORE studio from Thornton Tomasetti has developed an open-source tool for exploring design space on the web, named *Design Explorer*[1]. The exploration is orientated towards high-performance, low energy buildings and therefore the tool manipulates geometrical volumes (depth, height, orientation) and other performance related values (cooling, heating, lighting, orientation etc.). The users export

[1] http://core.thorntontomasetti.com/design-explorer/

parametric models in the form of a .csv file and a series of images and Spectacles models. The tool uses evolutionary algorithms developed in-house (the *brute force solver*), or optimization algorithms such as *Galapagos* or *Octopus* [41] that traverse the parametric model in an automated way. At the end, the tool returns a 2D grid visualization of the design space called a *parallel coordinates plot*. The tool is still used by researchers in the field of early design decisions and building performance.

Mueller considered the use of interactive evolutionary design for the computational exploration of the structural design space [42]. As part of her doctoral thesis, Mueller developed a computational tool named *StructureFIT*, and it is still fully functioning in the Digital Structures lab website. With the help of genetic algorithms, designers can explore and optimize alternative structural design candidates while intervening for the selection of parents to crossover and mutate. The topology of the design candidates is part of the designer's input, and the examples are in plane.

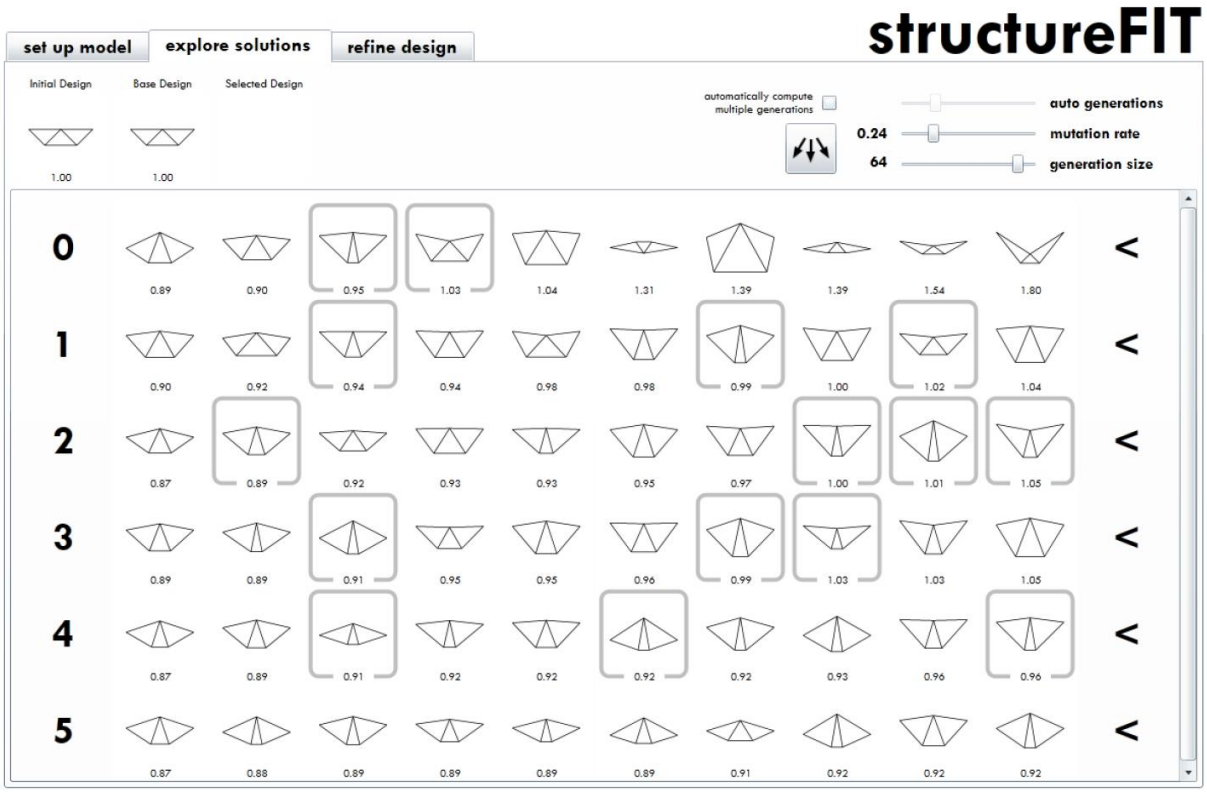


Fig. 4: StructureFIT GUI

Recently, Harding enriched the arsenal of available exploration tools, upon recognizing that multiple simultaneous displays are crucial for effective design space exploration [43]. *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 [3]. Part of this dissertation is built on top of its open-source implementation, its user-friendliness, its power and efficiency to thoroughly explore the design space, following respective modifications. Therefore, extensive presentation of the computational tool is provided in sections 4.2 and 4.3.

ParaGen, a tool that operates on structures whose topology is predefined, has been developed by von Buelow and has been used internally in University of Michigan [29]. The evolutionary exploration mechanism controls the nodal coordinates and imposes the nodes relocation until specific criteria are satisfied. The features that amplify *ParaGen*'s strength is the storage of design solutions to a database where from they can be retrieved at any moment of the exploration process at the designer's request.

Martini [44] exploited a population-based metaheuristic optimization method for multimodal size, shape and topology optimization of structural forces.

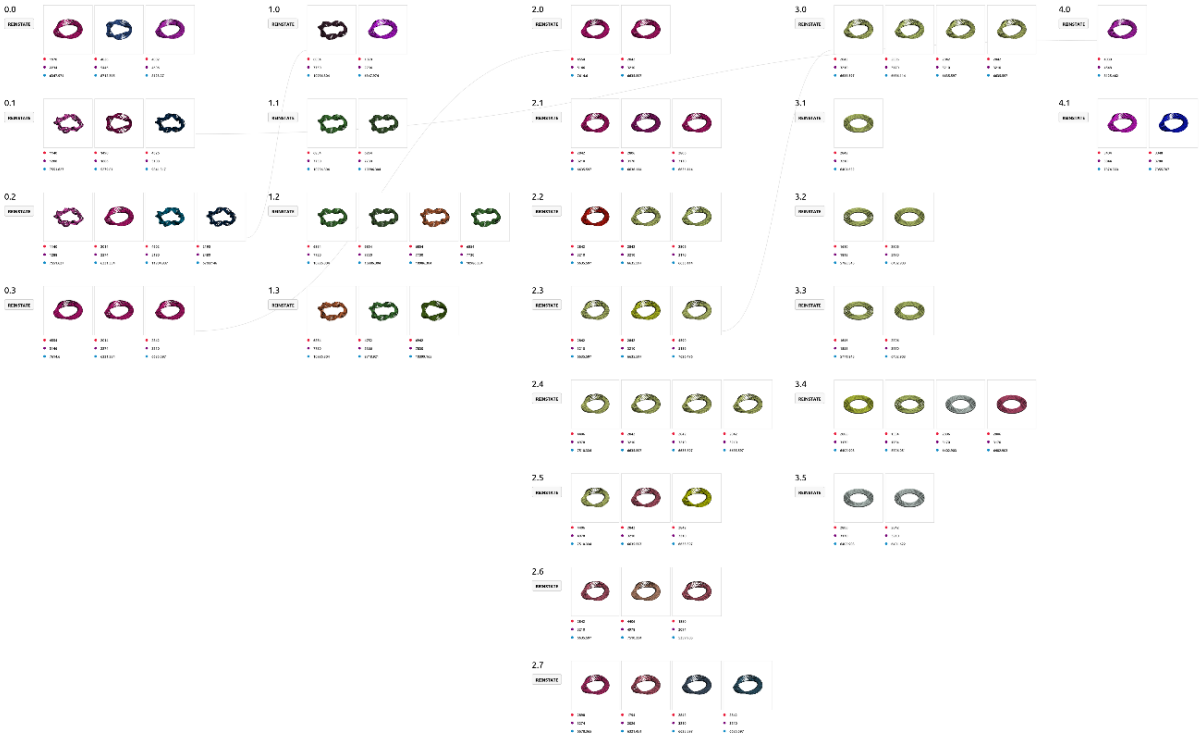


Fig. 5: Biomorpher: Search tree example. Each new search is formed from a previous population [3]

1.2.4 Conceptual structural design

Mueller [13] emphasizes that conceptual design helps designers to find high quality design alternatives through a Design Space Exploration (DSE) process. 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. Conceptual structural design stands for the process leading to schematic, early-stage static equilibrium representations, or, in other words, the outcome of structurally informed forms following the conceptual design stage. Conceptual structural design is a key phase in the design of structures as it recognizes the external forces (applied loads and support reactions) and thus frames the initial discussions regarding the required openings (spans) and the required material. Though schematic representations of static equilibrium are applicable to any material, an a priori selection of materials is recommended. In an abstract way, conceptual structural design is a representation of indicative force flows, to be used as first inspirations, prior to comprehensive structural analyses and form refinement.

Networks of bars is an abstract representation means of conceptual structural design (section 2.2.2). Graphic statics build on networks of bars and therefore it a convenient approach to generate and manipulate conceptual structural design candidates. The reciprocal relationship between the form diagram and the force diagram provides all the necessary information for a structure in static equilibrium. In the 19th century, graphic statics gained large popularity as a means to analyze 2D, or projected 3D structures. In the 21st century, its understanding and extension into three dimensions, together with representational and computational advances, has promoted it into a valuable exploration tool. Maillart used graphic statics in a creative but efficient way to explore the design space through early-stage schematic representations of static equilibrium [45]. Lachauer et al. [46] presented a novel method for computer-aided equilibrium modelling of structures in early design stages based on the force density method. van Mele

et al. [47] developed interactive applications that consider graphic statics and can be used to design efficient truss structures by imposing geometrical constraints to the force diagram. Fivet et al. [48] implemented methods for interactive design exploration of static equilibria using constraint-based graphic statics. As part of his doctoral thesis [49], Fivet also suggested a minimum set of operations to transform strut-and-tie networks with the help of two reciprocal diagrams (form and force). The operations included primitive operations such as *createPointInTheFormDiagram*, *createPointInTheForceDiagram*, *deletePoint*, *movePoint*, *mergePoint* etc. As he described, if algorithms exist to process these basic operations, they can be assembled in sequence in order to define and perform more complex operations. Hartz et al. [50] have enabled form-finding of structures that have both compression and tension forces through graphic statics. Ohlbrock et al. [51] introduced the use of a topological graph, that visualizes nodal relationships, as a manipulative design input. The Combinatorial Equilibrium Modeling (CEM) framework that Ohlbrock has proposed operates on the predefined topological graph and has been extensively used for the design space exploration of footbridges and stadiums. Among other applications where CEM has been used, Ochoa et al. [52] combined CEM with machine learning as a bottom up form-finding method of conceptual structural candidates.

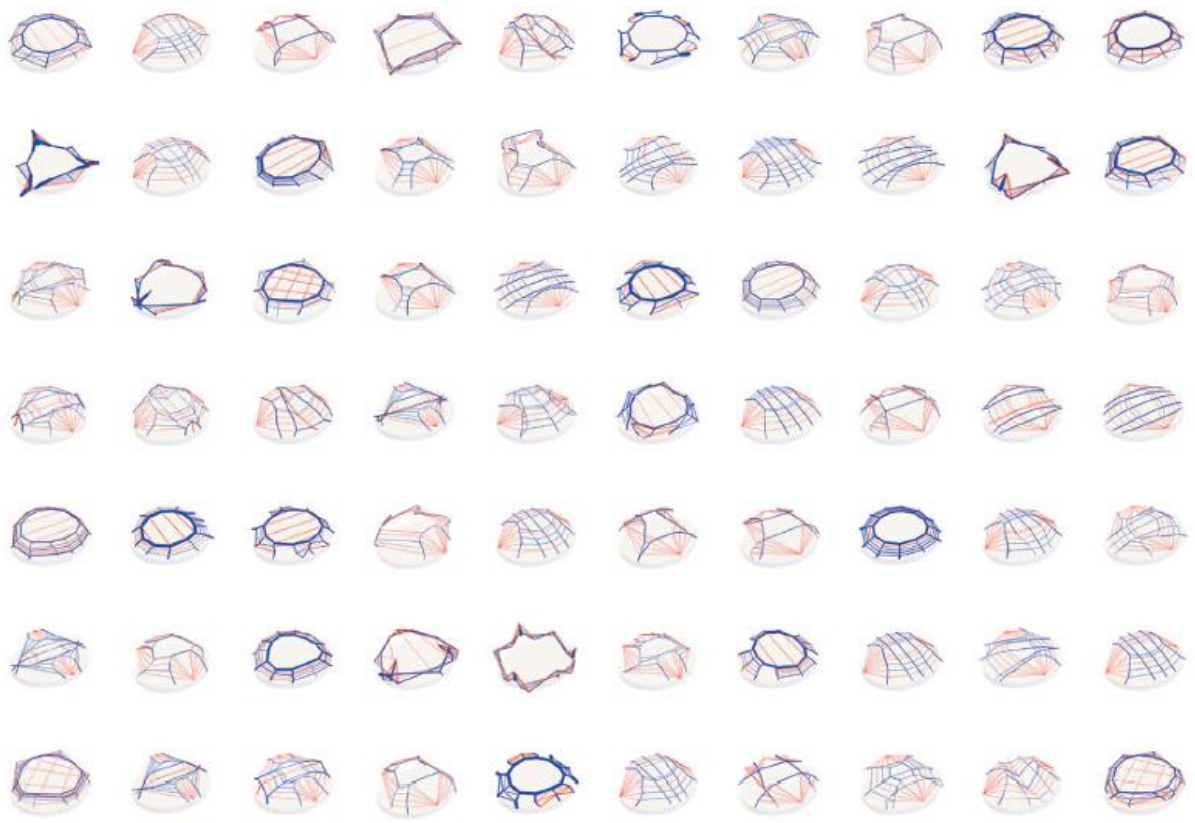


Fig. 6: Combinatorial Equilibrium Modeling combined with Machine Learning for the design of stadium roofs. Random sample of 70 generated design options [52].

Other projects for conceptual structural design that do not involve the use of graphic statics include the work of Mueller [42] and von Buelow [29] as described earlier.

1.2.5 Topology generation

The optimization approach that has associated its naming with the notion of topology is the famous topology optimization (TO). TO is concerned with finding an efficient structural layout and material distribution within a predetermined design space subject to specific loads and boundary conditions [53]. Throughout the years, multiple algorithms have been developed (*Evolutionary Structural Optimization*

or *ESO* [54], *Bi-Directional Evolutionary Structural Optimization* or *BESO* [55], *Solid Isotropic Material Penalization* or *SIMP* [56]). TO is a well establish deterministic approach. Namely for the same loading and boundary conditions it converges to the same geometry. Therefore, it does generate a topology but it does not contribute to the exploration of the design space.

Park et al. [57] suggest that layout optimization can serve as inspiration to designers at the conceptual design stage when given numerical discretization is coarse. Based on this concept, recently, He et al. [58] proposed an optimization framework that uses layout and geometry optimization methods in sequence to identify conceptual designs for building and bridge structures. The authors successfully highlight that the optimum, organic geometries, that their approach generates, are perfectly aligned with the need to design materially efficient forms, that reduce the material consumption, and with the advances in the field of construction, and in particular digital fabrication. Thanks to the integration of their framework in the parametric environment of Grasshopper in McNeel Rhinoceros, the exploration of new structural forms with reduced environmental impacts is expected.

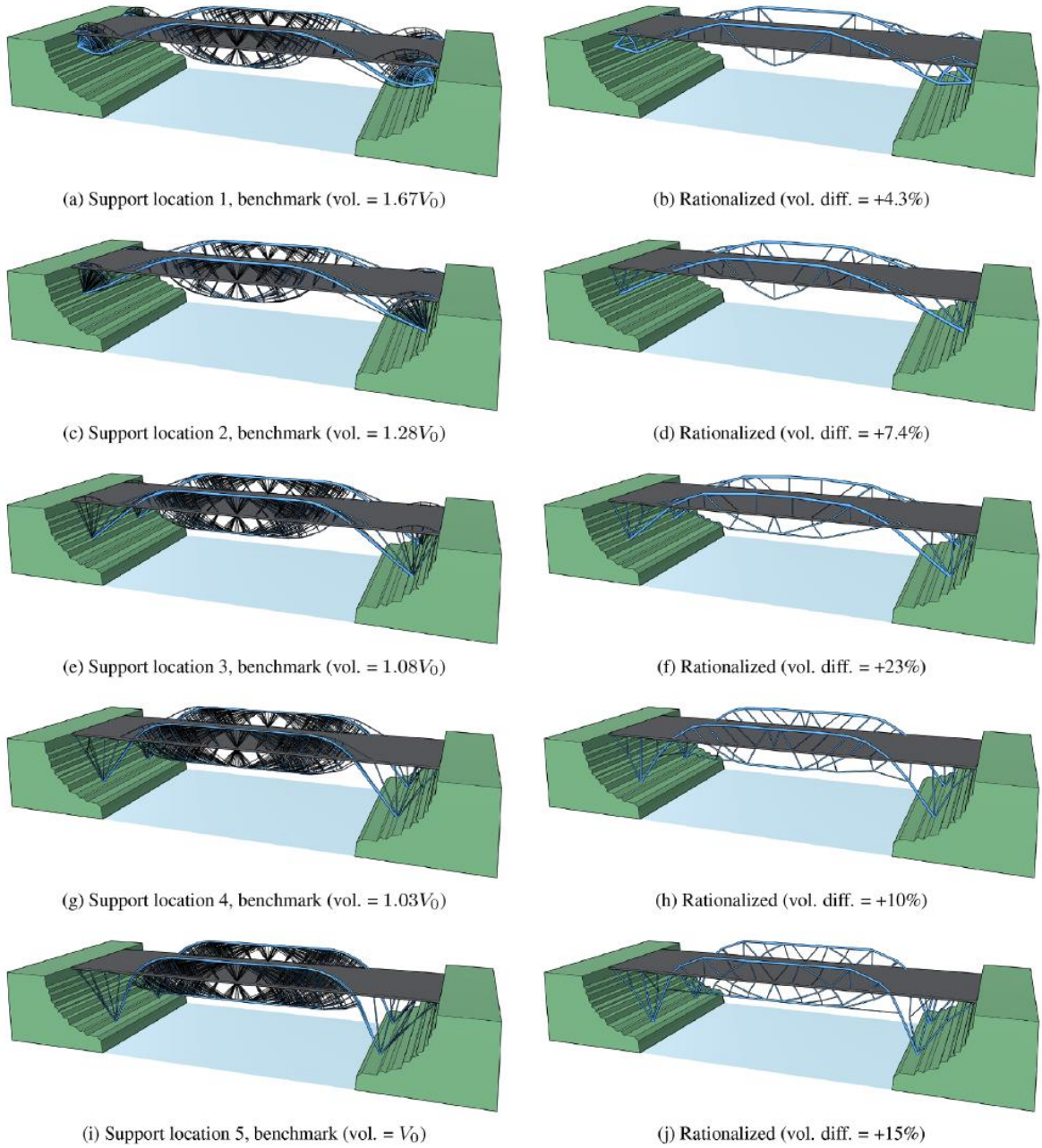


Fig. 7: Simple bridge example: parametric study solutions obtained by varying the roller support locations [58]

1.2.6 Shape grammars and grammar rules

According to Noam Chomsky [59, p.1], linguistics is concerned with the properties of successful grammars, namely linguistic structures with no specific reference to particular languages. In fact, linguistic syntax wraps up the principles and processes by which sentences are constructed. Meanwhile, any language is *"a set of (finite or infinite) sentences each finite in length and out of a finite set of elements. [...] Each natural language has a finite number of phonemes (or letters in its alphabet) and each sentence is representable as a finite sequence of these phonemes (or letters). [...] Similarly, the set of 'sentences' of some formalized system of mathematics can be considered a language".*

Stiny and Gips [60], compared shape grammars to linguistics grammars; *"an alphabet of shapes generates n-dimensional shapes"*. Since then, grammar and rules have often been proposed as possible support mechanisms for architectural designers [61]. Sets of design rules assembled in various configurations can generate numerous design variants. Many shape grammars and corresponding implementations have been proposed, but shape grammars are not widely adopted by architectural designers [62]. Nevertheless, until nowadays they continue to attract the attention of many scholars: Duarte [63]; Economou [64]; Chakrabarti [65].

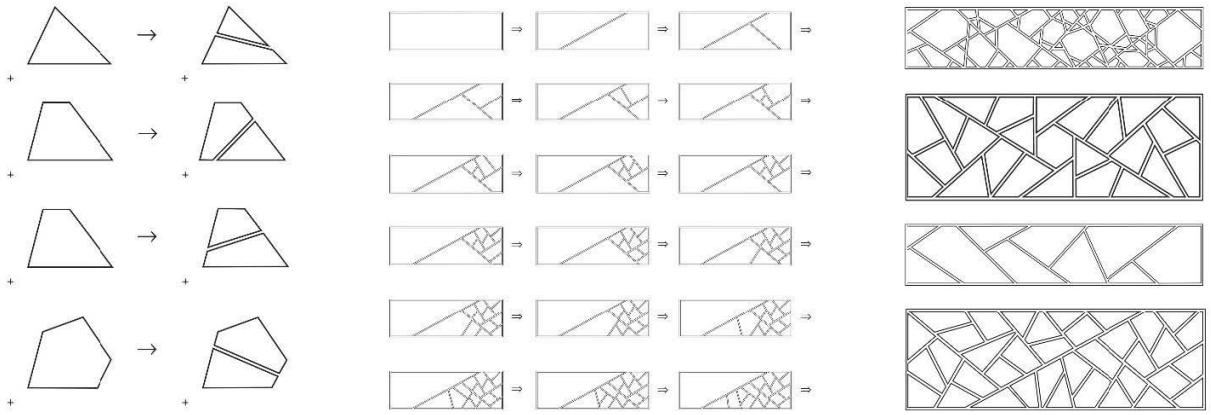


Fig. 8: Lattice pattern grammar [66]

Often, shape grammar applications relate to urban design studies as the combinatorial configurations of certain planar shapes imitates the plan view of urban blocks. Research projects dedicated to urban design include: Beirao et al. [67] who rearranged closed shapes and volumes for the exploration of urban patterns, Stouffs et al. [68] who suggested a rule-based approach to generate building data for urban planning analysis.

Another field of design that has welcomed the use of shape grammars is the floor planning. The most popular example is the study of existing Palladian villas by Stiny and Mitchell [59] who investigated how shape grammars can be formulated to generate new architectural plans. In floor planning, the rearrangement of closed shapes can be controlled through graph theory and allows specific relationship (proximity or connectivity) to specific building room types (e.g., based on use). Among the vast community of similar examples, Shekhawat et al. [8] have implemented a tool for computer-generated dimensioned floorplans based on given adjacencies. Other recent approaches [70,71] pave the way for more intelligent floor plans generations with the use of self-organizing configurations and reinforcement learning respectively.

At a volumetric level, Konig and Eizenberg [72] developed grammars for Frank Lloyd Wright prairie houses. They studied existing examples to formulate specific grammar rules which could generate design candidates that resemble those originally designed by Frank Lloyd Wright himself. Mitchell [73] focused on functional requirements and developed shape grammars that transform a plain cubical

volume into a shelter within a finite number of steps. Much later, Geyer [74] applied grammar rules at a component level for the design of entire buildings.

Shape grammars and grammar rules have attracted the interest of researchers working in the field of structural design. Lee et al. [75] coupled vector-based graphic statics with grammar rules for the generation of two-dimensional trusses. Although their approach allows the automated generation of diverse equilibrium structures, the rules definition is too abstract and not intuitive. Therefore, designers can hardly understand the impact of each rule on the design of trusses. Furthermore, it is unclear what is the recommended sequence of rules to be applied in order to converge to a state without any interim forces. Recently, Cascone et al. [76] proposed a structural grammar for the design of diagrid-like structures.

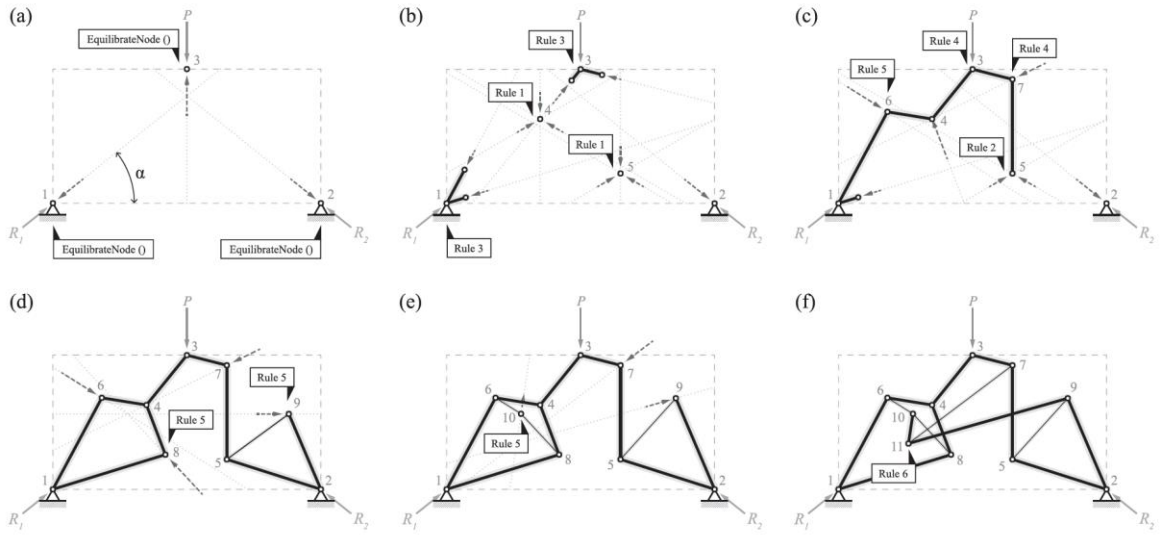


Fig. 9: Automatic random generation sequence through grammar rules [75]

1.2.7 Shape grammars and metaheuristics

More advanced approaches have coupled generative design, which is imposed by grammars, with metaheuristics. The used metaheuristics do not guarantee global optimality of the design solutions but rather support optimally directed design exploration. Shea et al. [77,78] applied shape grammars to the synthesis of trusses. The generated planar structures source from a grammar that ensures static equilibrium through triangulation. In order to achieve optimality, the grammar has been combined with simulated annealing. Another proof of synergy between associative geometry and performance-driven generative design consists of a method called *eifForm* which combines structural grammars, performance evaluation including structural analysis and performance metrics and stochastic optimization via simulated annealing. The method has been applied for the design of variant trusses as part of the same stadium roof [79]. Evolutionary shape grammars have not been tried out purely in structural engineering/design. O'Neil et al. [80] conducted a small survey on the benefits that shape grammars and evolutionary approaches have on creativity in general. The same design brief was given to all testers. Based on the provided feedback, they all managed to design a shelter without necessarily have any technical background, fact which confirms the potential that shape grammars and evolutionary approaches have in the field of design space exploration. Recently, Mueller [42] suggested application-specific grammar rules and manually generated trans-typological sets of two-dimensional structural systems with the help of interactive genetic algorithms. The implementation of the last project integrates multiple aspects. Metaheuristics facilitate plurality and since in this particular case they are interactive, they allow for higher control over the generative process and easy steering to the desired design pathways.

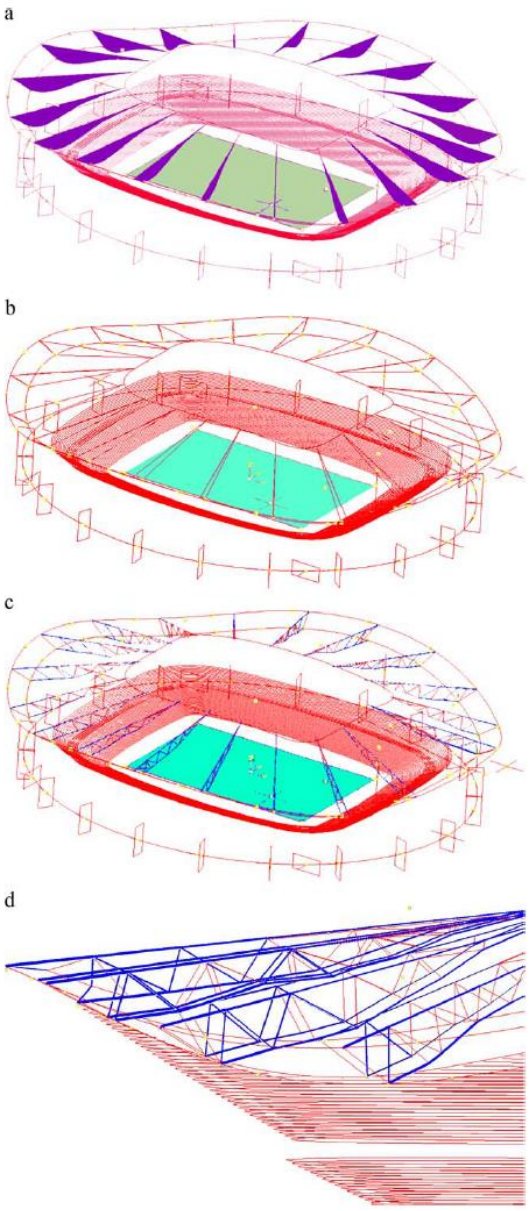


Figure 10: Optimized cantilever trusses generated by eifForm for a stadium roof [79]

1.3 Opportunities

It is evident that the combination of various fields is of utmost importance for the exploration of the design space. Each field contributes in its own way and therefore different combinations of theirs have been considered already. The presented research projects implement concepts that range from computer science to urban design and structural engineering.

This dissertation adopts the same spirit; combines concepts from different fields in order to develop new ways of conceptual structural design space exploration. It gets inspired by shape grammars and grammar rules to develop its own language, gets combined with interactive evolutionary computing and adopts the abstract representation of static equilibrium using networks of bars like graphic statics does. Precisely, shape grammars has spawned the idea of using a generic transformative policy which output variant design candidates for different input parameters. Interactive evolutionary computing has augmented the notion of exploration by granting control of the process to the designer who actively steers

the design process according to personal preferences. Networks of bars in compression and tension, is a minimalistic representation of equilibrium and force diagrams can easily derive from them to confirm the condition of equilibrium.

The next part describes the computational framework that supports a new design workflow that the author proposes. The part starts with the analysis of concepts in parametric design that frame the functionality of the framework and gradually funnels the reader to the proposed vocabulary, the workflow, the implementation, and the application studies.

2 Conceptualization of Policy-based Exploration of Equilibrium Representations (PEER)

This part explains the policy-based exploration of equilibrium representations for the generation of diverse networks of bars in compression and/or tension. It starts with initial thoughts that have formulated the conceptual core, followed by the used terminology, the intentions and the assumptions that frame the presented framework. After, the methodology is provided through primitive examples of equilibrium representations (sub-networks) that exemplify the entire concept.

2.1 Parameterization approach

This section describes the conceptual inception of the PEER framework. Its contextualization is supported by a comparative study that highlights the shortcomings of parametric design.

Considering that design encompasses a series of aspects, its rationalization and discretization into easily conceivable parts (as per the *"divide and conquer"* paradigm) often occurs. Linking design with algorithmic thinking is apparent, especially in the digital era that we live in. Indeed, algorithmic thinking counts on the wide spread of computers, which have hand-held the transition from the analogue era to the digital one – first computerized and later computational. The significance of algorithmic thinking goes beyond the technological advances themselves and its applicability does not imply the use of computers but often suggests it.

In its computer science notion, an algorithm is built around classes, procedures, libraries, constants, variables, and operates under arithmetical, logical, combinatorial, relational syntaxes. According to Terzidis [81], an algorithm serves as a means for articulating the problem, whether solvable or addressable. For all problems that algorithms are expected to handle, the algorithmic implementation requires good understanding of the dynamic variables and the underlying relationship among them. Namely, the identification of all input parameters, whose repeated calculation replicates identical output.

Perhaps the first mentioning of parametric architecture dates to early '70s when Moretti [82] used the design of a stadium as an example to explain how its form derives from nineteen parameters. Recently, in design, be it architectural, structural or mechanical, the constantly increasing project complexity, due to geometry- and/or performance-related requirements, parameterization has allowed, but also pushed, designers to integrate more and more technological means into their daily workflows. Numerous CAD and CAE tools have been developed, often out of need of accelerating the execution of tedious and repetitive design tasks. Novel parametric workflows have flourished, grabbing the attention of the involved professions and eventually gaining tireless supporters.

Thanks to the concept of parameterization and its immense adoption by numerous software, design as a process has been revolutionized. Finally, the construction of diverse models has become fast. This

achievement has made numerical modeling to be deemed highly exploratory and design creativity has been falsely linked to it. Undoubtedly, parameterization has remarkably accelerated the construction of multiple design candidates, but upon critical thinking, it is evident that the level of variation and novelty have gotten constrained by the parameters themselves, their established *interrelationship* and their *definition.*

The following subsections are dedicated to these two features and through comparative studies they underline the parameters' impact on the exploration capacity of parametric design processes.

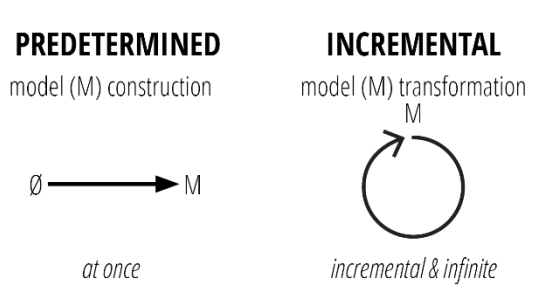


Fig. 11: Parameter interrelationship in design

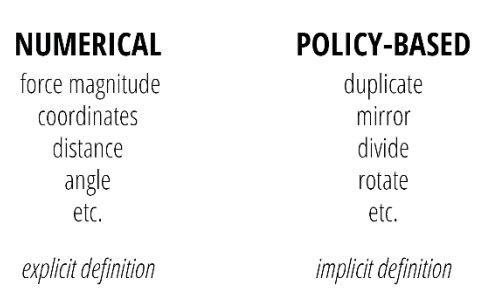


Fig. 12: Parameter definition in design

2.1.1 Predetermined vs. incremental interrelationship

Parametric modeling is possible through textual or graphical programming environments. Such environments are used to build directed acyclic graphs (DAG) [43], which construct parametric models. The alteration of at least one of the finite input parameters in the DAG, fires the construction of a new (digital) model. Like this, parametric modeling facilitates the generation of new design candidates, with less effort and in less time [83]. In principle, each design candidate visualizes the impact of the input parameters on the model as a result of their underlying relationship described by the DAG. Their bond is *predetermined* or *incrementally changing*.

Predetermined design

In predetermined parametric design the digital model is constructed once and instantly after the execution of a set of mathematical or geometric calculations [Fig. 11, left]. The interrelationship among the input parameters is arranged beforehand and the model output is likely to be predictable. In this sense, it takes after the deterministic algorithms; as many times as the calculations are executed, they always output the same model. In reference to architectural forms, be it buildings or structures, predetermined design allows both modelling and analysis to take place in the same integrated environment and at a later stage favors (post-) optimization.

Parametric description of forms and effortless construction of design candidates come at the expense of some cost. The extrapolation of the input parameters that describe a form and the setup of a parametric model through a DAG require considerable amount of laborious time, facts that make parametric design much slower than hand-drawing. Moreover, each single parametric model can generate a limited number of diverse design candidates, as diversity is constrained by the static topological structure of the DAG.

Given that each DAG represents only a family of forms, described by the same inputs which corelate in a fixed way, diverse DAG are required to generate original and diverse forms during the design exploration process. Woodbury [84] has dedicated an entire chapter in his book *Elements of parametric design* to answer what parametric modeling is. Though he never really answered this question, he gave a synonym to the term, *constrained design*. He has not provided any further reasoning to support this terminology, but it is believed that Woodbury's omission relates to the abovementioned weaknesses.

Finally, given that the digital model comprises the execution and completion of numerous calculations, no intermediate stages of the model exist. This shortcoming goes against the concept of high-level and thorough design exploration.

Incremental design

Contrary to the predetermined design approach, the incremental design approach transforms existing models step-by-step [Fig. 11, right]. Whilst it can have parametric definition, incremental design does not operate on the base of a DAG. Still, it is likely to take advantage of existing visual programming methods such as Rhinoceros Grasshopper and Autodesk Dynamo. Incremental design operates in an open and segmented way. While the process can be fully automated, it lets the designer interrupt, redirect, or move a step backward, at every step of the generative process. In other words, designers provide input, interact with the output and influence the design process at every step of the model refinement. Incremental approaches are therefore closer to intuitive modelling processes, e.g., sculpting, painting or architecture, where transformative actions are applied successively, each based on an updated understanding of the problem being searched without a preconceived idea of what the outcome should be.

Concretely, incremental design starts with an intermediate model and transforms it at any moment that a variation is desired, and for as long as it is desired. From start to end, every step of the process provides a self-contained model. The transformation of an intermediate model is equivalent to a controlled mutation which clearly describes the performed operations (e.g., an element's introduction, deletion, scaling, mirroring etc.). The transformation applies onto the entire model or parts of it (sub-model). In the latter case, the new entity gets immediately integrated onto the current model and updates it before the next transformation starts. The transformative process may continue infinitely provided that the required input for the intended transformation is always available. Practically, the transformative process comes to an end as per the designer's request and/or when specific expectations are met. The strength of incremental design over predetermined design is highlighted by the fact that all intermediate design candidates are tentative but still possible to be further explored if new design requirements emerge. This convenience makes incremental design highly appropriate for the exploration of the design space.

2.1.2 Numerical vs. policy-based definition

In computer science, many types of parameters exist: integers, floats, doubles, strings, characters, booleans etc. In parametric design, the definition of a parameter is paramount and relates to: (a) the domain size where a parameter gets values from and (b) the nature of the value itself. Based on its nature, a parameter can be categorized as *numerical* (arithmetic) or *policy-based* (descriptive) [Fig. 12].

Numerical design

The definition of a numerical parameter includes four values: (a) the lower bound, (b) the upper bound, (c) the resolution and (d) the current value. While the last is subject to change freely, as per the designer's desire or optimization routines, it may not always be true. The first three values remain fixed during the design process. Moreover, they are often predefined in an arbitrary way as designers cannot always predict/understand the effect of a set of independent variables on some dependent variable under certain conditions. Consequently, design exploration is merely limited to numerical fine tuning within the bounds and the resolution intervals. Therefore, design space exploration is limited to a fraction of it. Examples of numerical parameters are arithmetic values of node coordinates, force magnitude, distance, angle, curvature etc.

Policy-based design

The definition of a policy parameter is not tangibly related to numbers, but their use is not excluded. Usually, a policy parameter consists of a set of rules which are expressed in a qualitative, descriptive way with high level of abstraction. However, this abstraction does not link to any discrepancy or misunderstanding for any code interpreter. Instead, the abstraction makes their definition independent of bounds and ultimately each value describes a feasible request at all times of the design process. In design, policy parameters are selected from a pool of available policies and therefore only the policy selection indicator complies with some sort of bounds that describe the pool size.

Policy parameters include notions of comparison between *before-after* states (when they explicitly describe their impact) or *currently available* options (when they explicitly select via performance, e.g., the oldest/largest/furthest etc.). Their definition matches with the concept of incremental design, where the model constantly changes in compliance with specific design intentions. Policies, disguised as shape grammars or transformation rules, first showed up in the literature many decades ago when they entered various design fields – e.g., painting [60], architectural design [69], structural design [75], urban design [67], industrial design [85].

2.2 Introduced terminology

This section briefly presents the terms that describe the PEER framework. Among other terms, such as rules and transformations, the definition of a policy is also provided and clarifies the framework naming.

2.2.1 Rules, policies and transformations

In the context of this dissertation, three terms that relate to linguistic syntax as described in section 1.2.6 are introduced and get contextualized: (a) rule, (b) policy and (c) transformation.

A *rule* represents an explicit guideline and has no ambiguity about its outcome. Its definition incorporates low, or high, level of abstraction, but often requires several supplementary parameters that thoroughly describe specific features and ensure identical outcome in case of replications. Rules have the notion of guidelines for various aspects and are grouped in respective pools (groups). Each pool has finite or infinite number of rules based on its nature. In linguistics, a rule is equivalent to the grammar which is supported by the language alphabet. The alphabet itself contains a finite number of letters, i.e., Phoenician alphabet has 22 letters; 24 the Greek one; 26 the Latin one; 32 the Cyrillic etc. Grammars define how the substances of an alphabet are assembled together in a structured way.

In design, rules are guidelines supported by parameters. Although the number of parameters is finite and such that the design intentions are communicated in a clear and precise way, the number of design rules cannot be limited to a certain number. Rather, it varies with regards to what they regulate and how they are defined. Often based on their functionalities they are assigned to different pools (groups). In the context of this dissertation there are four pools of rules and each has a finite size [Appendix C]. For some of the pools the list of rules is not exhaustive, while for others it is.

A *policy* frames a set of rules to support decisions, with each rule being responsible for specific aspects. In linguistics, a policy is equivalent to a subject-object-verb (SOV) or subject-verb-object (SVO) structure. Namely, its definition dominates the implicit formulation of a sentence in most natural languages.

In design, policies refer to the process of making important decisions that have impact on a system. The sequential order that each of the respective rules is applied is specific. Occasionally, some rules are not applicable because of constraints introduced earlier during the policy definition. When all rules are

chosen and the policy is confirmed as feasible, it can safely be applied. In this dissertation there is one policy that is applied on networks and transforms them while retaining equilibrium. Transformation policies are applied onto a digital model and provoke a model *transformation.*

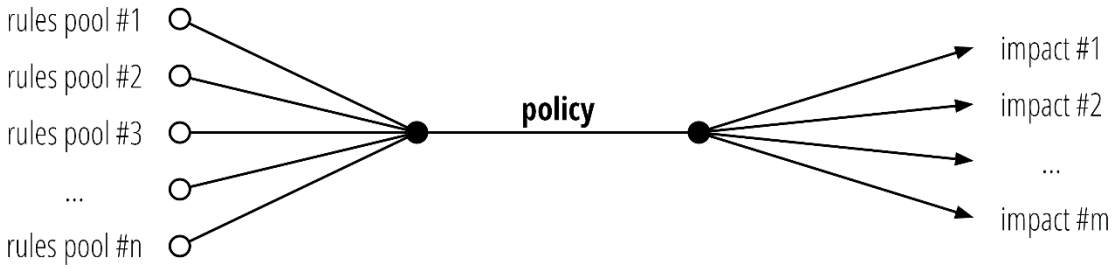


Fig. 13: Hierarchical relationship among rules and policies. Each combination of rules forms a policy that has a different design impact. Thus, the number of pools of rules and possible design impacts do not match ($n \neq m$).

Globally, each transformation is the result of a finite number of parameters, described in various ways and in accordance with certain constraints and design intentions. Different combinations of parameters are likely to result in identical model transformations, but this relationship is not reversible, as identical design transformations typically indicate identical parameters. The generic hierarchical relationship among these three terms is shown at Fig. 13. Later, in section 3.1.4, a detailed representation of their underlying relationship is provided.

PEER is a grammar-inspired design framework which, as opposed to Stiny and Gips [60], does not involve an alphabet of shapes. A list of parameters supports the creation of rules of different aspects, whose combination defines policies that generates diverse design candidates.

2.2.2 Network of bars

A network of bars represents an assembly of applied forces, anchor points – i.e., pin-joined nodes - and bar elements in static equilibrium. In other words, it is a synthetic representation of a structure's static equilibrium by means of bars in compression or tension, as was the case for example in Robert Maillart's design of the Salginatobel bridge in 1928 [45]. Networks of bars are developed for a chosen predominant load-case and are expected to materialize into effective structural forms, be it reticulated or continuum, during subsequent design steps. Throughout history, the radical abstraction offered by such models has proven to be highly useful and versatile for the exploration of unconventional masonry [86], reinforced concrete [87], steel [88] and timber [89] structures.

2.2.2.1 Disconnected / Incomplete

When a bar network is disconnected, possibly the nodes at the end of the graph leaves are not in static equilibrium. The temporary introduction of interim forces allows a network of bars to maintain static equilibrium. The addition of interim forces to a disconnected network also explains the term *network of bas in interim static equilibrium*. In PEER, the introduction of interim forces is imposed by the transformation policy and they remain in the network until the upcoming transformation step that replaces them with bars. When the last pair of interim forces is replaced with a bar, the network gets connected and it is said to be complete.

2.2.2.2 Complete

An example of a complete bar network is a reticulated structure – i.e., a truss structure - together with the applied loads and the support reactions. Such networks are in *global static equilibrium* and no additional bars are expected to be added.

2.2.3 Design domain

The growth of networks, as imposed by transformation policies, strictly occurs within predefined regions. These are expressed as geometric boundaries in two or three-dimensions, as per the designer's decision. Voids are likely to be contained too. The definition of a design domain is a permanent input [Fig. 18 and Fig. 19] and practically frames the field of action for the policy transformations.

2.2.4 Model

It stands for an entity that provides the substances to apply a transformation policy onto. At its minimum entity, a model consists of the design domain and two force vectors (applied loads or support reactions) in equilibrium. No bars are necessary. Any introduced nodes, bars and interim forces as imposed by the transformation policy are contained within the model.

2.3 Intentions and scope

PEER supports an incremental, policy-based design workflow for the exploration of diverse equilibrium representations at the early-stage design. Its implementation aims at facilitating the conceptual design of structural forms, through network transformations. Diversity aims at fighting design fixation and/or lack of creativity. The workflow meets these objectives by means of the following features:

- 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 transformation process
- it ensures growth of the network within specified geometric boundaries
- It transforms networks of bars by means of a policy that is defined as a set of choices of low- or high-level rules

The incremental transformative process operates either in manual or semi-automated mode. When manual, the network grows under the close supervision of the designer, who instantly reacts to arising challenges or deviations from custom preferences and steers the design candidate accordingly. When operating in automated mode, the designer provides the input parameters and if he/she does not interrupt the process, the transformations continue without any supervision for a number of steps. When the transformations are completed the process grants full control to the designer who resumes the process or undoes the growth.

Combining the two modes, designers explore the design space in different pace. The former is preferred for well-structured design intentions and step-by-step transformations, whereas the latter emphasizes blind exploration and multiple transformation steps. As such, the PEER framework is not meant to be a toolkit for numerical analysis of structures. Rather it is orientated around semi-automated exploration of structurally aware spatial forms. Topology is neither defined nor bound to existing structural geometries. Instead, topology is defined during the decision-making process and is the main output of the exploration process.

2.4 Assumptions

A single, bespoke, parametric, equilibrium-aware choice of policy fully controls the entire transformative process of bar networks. The presence of a single policy, as well as its definition, have been decided

and implemented in a way that combines universality and speed of application. The reasoning behind these decisions is presented below. The definition of static equilibrium during the process and the interim network's mechanical behavior, including simplifications and abstraction, are also provided.

2.4.1 Policy

Acknowledging the benefits of policy-based versus numerical design approaches [section 2.1.2], this thesis presents the conceptualization and implementation of a design workflow that builds on a transformation policy. Inevitably, utmost importance has been given on how this policy has been incepted, where it is applied onto and what its application imposes.

Questioning functionality, one could argue that a prospective static equilibrium-aware transformation policy provides either a set of primitive transformations - i.e., divide a force, get the resultant of two forces, split a bar etc.- or a set of specific transformations - i.e., aggregate triangles together, support with vertical columns etc. - to generate standardized structural forms. In the former case, many redundancies among the policies are avoided, but the sequence of transformations for the generation of a design candidate is expected to be long and hard to manage. In the latter case, the policy is prone to become self-constraining and remain type-specific and result-oriented, generating predefined forms that lack diversity and/or creativity. In fact, the proposed policy is a trade-off of the above and sufficient to explore the full design space efficiently. The policy generates any type of topology for any given loads and simultaneously ensures static equilibrium. In other words, the policy itself neither constrains the topology of the generated networks nor recursively replicates existing topology patterns or configurations - i.e., triangles, pyramids etc.

By definition, the policy considers up to three interim forces at each step. At first sight, this seems contrary to a real design challenge where the known forces, even after simplifications and abstractions, could number some dozens. The reasoning behind this choice follows:

- **Inception**: It is evident that there is no way to replace a set of two non-coplanar interim forces with an equivalent network of bars that has fewer interim forces [Fig. 17i]. The consideration of three non-coplanar interim forces allows the above-mentioned replacement [Fig. 17ii]. The policy definition can be further expanded for sets of four, five, six or more interim forces that are manipulated simultaneously and generate new (sub-)networks. However, no exploration boost is anticipated, and no value is finally added to the methodology itself. On the contrary, 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.
- **Implementation**: To retain simplicity and constrain its complexity as much as possible, the minimum number of interim forces to transform is sought after. Simplicity is a key feature that adds value, increases the popularity of any design workflow, multiplies 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 framework. This dissertation comes along with a fully functional design toolkit, and therefore further development and maintenance considerations are necessary.

2.4.2 Inner working

Global static equilibrium

Static equilibrium-awareness imposes several constraints, included in the policy definition already. In physics, a system is in global static equilibrium if translational equilibrium and rotational equilibrium

around any given point are satisfied. This condition is assumed in the beginning of the process and no further checking is required during the incremental transformations with PEER. However, the networks are in equilibrium at every intermediate step. This explicitly means:

- **State A**: At the beginning of the transformative process, the external forces (applied loads and support reactions) are provided and they are assumed to be in static equilibrium. Therefore, no static equilibrium checking is performed. If no bar exists, opposite and equal interim forces are introduced to retain static equilibrium at the force anchor points. At this state, the model is self-contained and in interim static equilibrium, thanks to the introduced interim forces.
- **State B**: Throughout the process interim forces and bars coexist formulating an incomplete network of bars. Again, at this state, the model is valid and in interim static equilibrium. The introduction of interim forces only considers translational equilibrium. The rotational equilibrium of incomplete networks is not checked but it is satisfied.
- **State C**: At the moment that no interim force exists anymore, the network is complete and in global static equilibrium.

Here are the static equilibrium assumptions. To achieve the closure of the network as described at state A, the following condition must be satisfied. The set of external forces, provided as input, must be in rotational and translational equilibrium. The implemented algorithm does not post-process the input forces in order to satisfy static equilibrium. As such, the designer bears the responsibility of providing input that complies with this condition. The algorithm's intervention is limited to the introduction of opposite and equal forces to impose interim static equilibrium.

As long as the network remains incomplete (state B) only translational equilibrium is considered but both translational and rotational equilibrium are guaranteed. Skeptical designers can geometrically validate translational static equilibrium by manually building a force diagram per node at any moment of the process. Placing the interim forces, the applied loads and the supporting reaction in a tip-to-tail configuration, results in a closed polygon; confirming translational static equilibrium [88]. Considering that all the nodes of the interim network are in translational equilibrium (a point in translational equilibrium always satisfies rotational equilibrium) and the external forces are in rotational and translational equilibrium, checking rotational equilibrium is redundant.

At state C, the pool of interim forces is empty, and the network is known to be complete. Each node is in translational static equilibrium and the entire network is finally in *global static equilibrium*, rotational and translational.

Pre- and post-processing

As described before, the algorithmic implementation does not consider any pre- or post-processing of the input parameters or of the final result. Therefore, in cases of applications with equally distributed loading, no translation to point loading should be anticipated, if not considered by the designer in advance.

Combinations of load cases and dynamic loading

The algorithmic implementation does not consider combinations of load cases. Every generated network is load case specific as per the initial model setup. This implies that in order to generate a network which maintains static equilibrium for a combination of load cases, respective considerations must be made in advance to provide the input (external) forces accordingly. In general, the addition of applied loads to different locations - i.e., to the lateral nodes of the network due to lateral loading - is expected to generate different topology or new nodal locations, if not both. Thus, superimposing a number of networks

generated for an equal number of load cases does not guarantee their merger into one “master” network will resist all load cases, because the topology is genuinely different.

Although multiple loading cases - i.e., wind load, snow load etc. - can be tested individually for the generation of different bar networks and without the intention to be superimposed, the consideration of dynamic loading - i.e., seismic load - is not feasible. Aiming at conceptual structural design, answering these problematics is out of the scope of this framework. Acting as early stage, conceptual structural models, the generated networks are indicative force flows and equilibrium representations to be used as first inspirations, prior to comprehensive structural analyses and form refinement.

Material- and cross section-independent equilibrium representations

Other facts that conclude the list of assumptions consider the bars behavior. Bar networks in static equilibrium are synthetic representations of a structure’s static equilibrium by means of weightless bars in tension or compression. As such, only axial forces along the bars are considered, which indicate the use of struts or ties. Materialization and dimensioning are not part of these representations and this research.

2.5 Meta-methodology

The presented theory and its algorithmic implementation into a seamless design workflow pave the way for a digital, equilibrium-based design space explorer. Its conceptual inception dates to Corentin Fivet’s dissertation [49] and his postdoctoral research at MIT while working with Caitlin Mueller and Juney Lee [75]. Using this vision as a starting point, the theory of a transformation policy for bar networks was developed by the author.

First, to control the continuation or the termination of the exploratory process, the term *entropy rate* was introduced [section 3.1.4.1]. Since then, the entropy rate has remained as the dominant input parameter of the process that directly controls the impact of the network transformations (convergence, stagnation, divergence). At that stage, a first prototype was implemented in Python. No other inputs were integrated in the framework keeping the process completely random. From an algorithmic point of view, the implementation mainly included the mathematical solving for the replacement of sets of two interim forces by bars in compression and/or tension. This was proved to be enough to generate random and uncontrolled planar bar networks in static equilibrium.

Aiming at increasing the control over the design process, the concept of two domains was born: *entropy rate domain* [section 3.1.5.2] and *constructability domain* [section 3.1.5.3]. These are both geometric boundaries whose computation is necessary to ensure the transformation feasibility for each entropy rate, especially for occasions of highly non-convex design domains. Their intersection represents the geometric domain where a new node can be safely added, while retaining static equilibrium and ensuring the uninterrupted bars connectivity with existing nodes. Their computation required the algorithmic implementation of numerous functions, such as Isovist [90] in 2D and later in 3D. All along, an extended library of geometric manipulations based on computational geometry was developed, building on existing class objects already provided by RhinoCommon and/or new custom ones. Eventually the successful implementation of the above-mentioned domains confirmed the initial assumptions and allowed for more controlled application studies.

Convinced by its functionality and its robustness to increase/decrease the entropy rate before converging to a closed network of bars, the theory was extended in three-dimensions for the generation of spatial networks in static equilibrium. Considering that a pair of two interim forces (binomial) can never decrease the network entropy [Fig. 17i], sets of three forces (trinomial) were considered [Fig. 17ii]. This

addition brought unforeseen complications. The desired topology of the new incremental sub-network, called *topological configuration*, enlarged the list of input parameters. The categorization of different connectivity patterns inspired the representation of a sub-network as a tetrahedron [Fig. 16, top]. Three out of four vertices superimpose the anchor points of a trinomial of forces, while the fourth one represents a new node that is introduced [see section 2.6.2]. The tetrahedron edges superimpose all possible connecting bars between these four vertices. Meanwhile, a tetrahedron expresses the transformation scenarios for sets of one, two or three interim forces, for all topological configurations and all entropy rates. More importantly, it allows the mathematical description of each transformation through a single matrix-vector equation and a parametric policy. The updated implementation evidently confirmed the feasibility of policy-based generation of spatial bar networks in static equilibrium.

Later, in an attempt to increase the level of control on the transformative process, bar length bounds were introduced [section 3.1.5.4]. Last but not least, the proposed framework was further enriched by fusing it with the existing platform of the *Biomorpher* plugin [part 5]. Its UI and its IGA operability not only augmented the exploration capacity of the methodology itself, but also allowed the simultaneous visualization of multiple phenotypes [section 4.5.2].

2.6 Methodology

This section uses elementary networks - i.e., sets of one, two or three forces - to exemplify how the network transformations work. The transformation step described in section 2.6.2 is universal and is repeated identically, regardless of the network size.

2.6.1 Overview

The design process consists in transitioning from a disconnected network [section 2.2.2.1 / Fig. 14, left] in interim static equilibrium to a complete network [section 2.2.2.2 / Fig. 14, right] of bars in global static equilibrium.

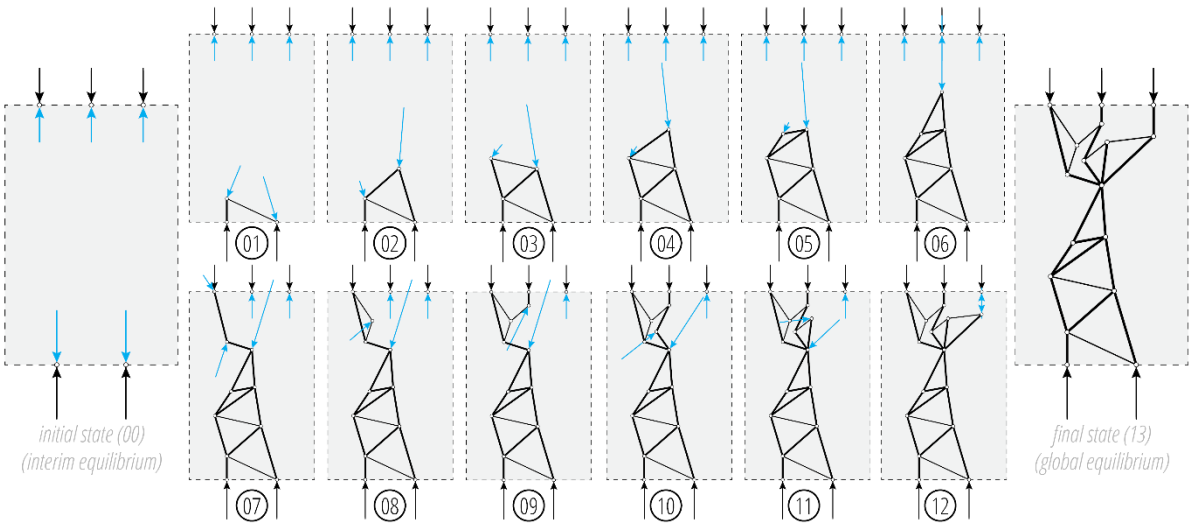


Fig. 14: Incremental policy transformations for the generation of a pylon-like network of bars in compression and tension.

The process starts with a set of force vectors - i.e., applied forces and support reactions - applied to nodes – i.e., anchor points - contained within the geometric boundaries of the *design domain*. A disconnected network retains (interim) static equilibrium with the help of interim forces. Their elimination through the addition of bars incrementally transforms a disconnected network into a complete network,

namely a complete equilibrium representation acting as a conceptual design candidate that satisfies static equilibrium. The transition consists of: (a) the elimination of all interim forces; (b) the introduction of a new node *P* for every transformation step; (c) the creation of new bars in compression (struts) or tension (ties).

The process being incremental enables the designer to explore the design space at custom pace and towards desired directions. Each transformation step is controlled by a designer-defined policy that by construction considers static equilibrium and therefore filters out all design candidates that do not satisfy static equilibrium. Thus, designer safely explores the design space strictly among candidates that satisfy static equilibrium. Other non-quantitative criteria – e.g., aesthetics – are assessed visually by the designer at the end of the transformation(s) as per the requested number of steps.

Precisely, the personalized design preferences are communicated to the transformative mechanism through a set of rules that formulate the policy. The description of the input and output for each transformation step follows.

2.6.2 Transformation aspects

The network transformation is the result of a defined policy which is described by a set of four rules, each responsible for different aspects:

- the speed of the growth - i.e., the number of interim forces to add or delete
- the starting point of the growth - i.e., the choice of interim forces to eliminate
- the structural behavior of the growth - i.e., the type (compression/tension) and magnitude of the axial forces developed along the bars
- the geometry of the growth - i.e., the position of the new node *P*

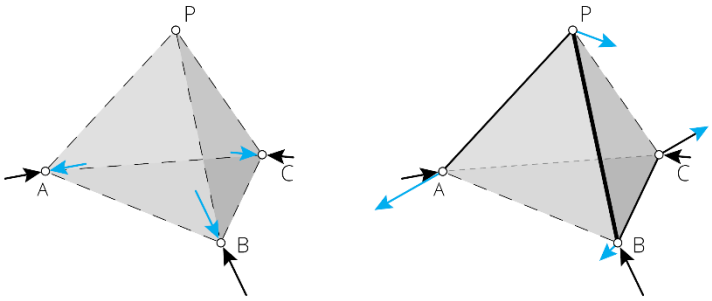


Fig. 15: Before/After transformation in space. Black arrows represent force actions that are applied externally to the system or by other compression or tension bars. Cyan arrows represent interim forces. Bars in compression are thick lines. Bars in tension are thin lines.

The *entropy* is a measure of the number of interim forces in the network at a given moment. An intended increase, decrease or retainment of the network entropy triggers a network transformation. Hence, the *entropy rate* of a transformation step represents the difference between the number of interim forces present in the disconnected network before and after the transformation step. A negative, null or positive, entropy rate leads to the *convergence*, *stagnation* or *divergence* of the transformative process. The desired entropy rate has a direct impact on the transition from a disconnected network to a connected one and it imposes certain constraints to the process – i.e., the geometric domain for the new node placement is entropy rate specific, the feasibility of certain topological configurations is subject to the desired entropy rate. Its definition consists of an explicit choice from a balanced ternary system {-1,0,1}. More information is provided in section 3.1.4.1.

Each transformation step operates on a set of three interim forces, which can be coincident with each other. Therefore, three types of sets of forces are considered: (a) monomial; (b) binomial and (c)

trinomial. Every disconnected network has a pool of interim forces. At every transformation step the respective pool is retrieved and the available interim forces are grouped in sets, either explicitly or by means of a high-level *force(s) selection rule*. In three-dimensional design domains, the transition to a complete network is possible only through transformative operations on trinomials, whereas the use of other binomials and/or monomials is not excluded. Instead, in two-dimensional design domains, the use of binomials and/or monomials is sufficient for the transition to complete networks. More information about the available rules of force(s) selection, their nature and their implementation are provided in section 3.1.4.2.

Following the defined policy, up to three bars are created. Each corresponds to one of the six edges of a fictitious tetrahedron built from four vertices: the new node *P* and the existing anchor points *A*, *B* and *C*. With the help of this representation, all the available sequences of continuous bars that connect all four vertices are recognized [Fig. 16]. For trinomials, three different *topological configurations* are recognized (*in-between*, *peripheral* and *central*), named after the sequential between the *P* node and the anchor points. Binomials and monomials formulate fewer topological configurations [Fig. 17].

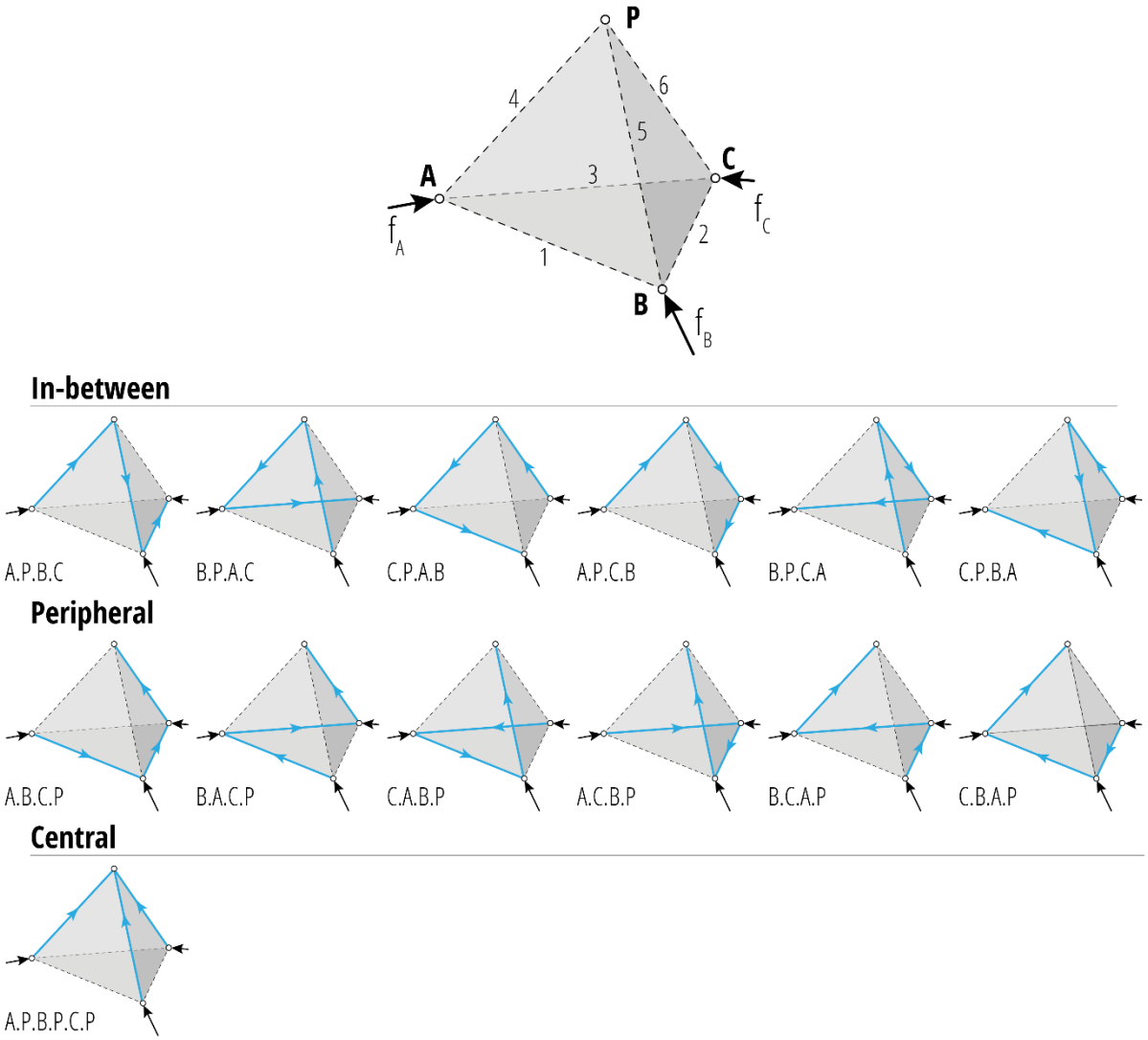


Fig. 16: All possible topological configurations of a trinomial

Besides the replacement of interim forces by new bars, the interim static equilibrium condition imposes the addition of new interim forces applied on *A*, *B*, *C* and *P*. The direction and magnitude of these new forces is calculated in relation with the anchored forces (external forces or stresses from existing bars)

and the newly introduced bars. When multiple equilibrium stages exist, the forces magnitude in the newly introduced bars is provided explicitly or by means of a high-level *force indeterminacies rule.* More information about this rule is provided in section 3.1.4.4.

The completion of a transformation step includes the introduction of a new node *P*. Its location, highly constrained by the desired entropy rate [section 3.1.5.2], relates to the end point of one, two or three of the introduced bars. Its preferred location is defined explicitly by coordinates or implicitly by means of a high-level *node placement rule*. More information about the constraints that this rule brings is provided in section 3.1.4.3.

The output of each transformation step consists of:

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

The input as described before is used identical for a single transformation step or multiple transformations. The designer is free to update it according to personalized preferences as often as needed. The output is integrated directly to the existing (disconnected) network and contributes to the incremental transition to a complete network.

Primitive examples of equilibrium representations (sub-networks) that exemplify the entire concept are illustrated at Fig. 17. The figure considers all possible combinations of different entropy rates, topological configurations, types of interim forces and demonstrated the transformation step impact on a sub-network in interim equilibrium. Hence, it represents the design vocabulary of PEER framework.

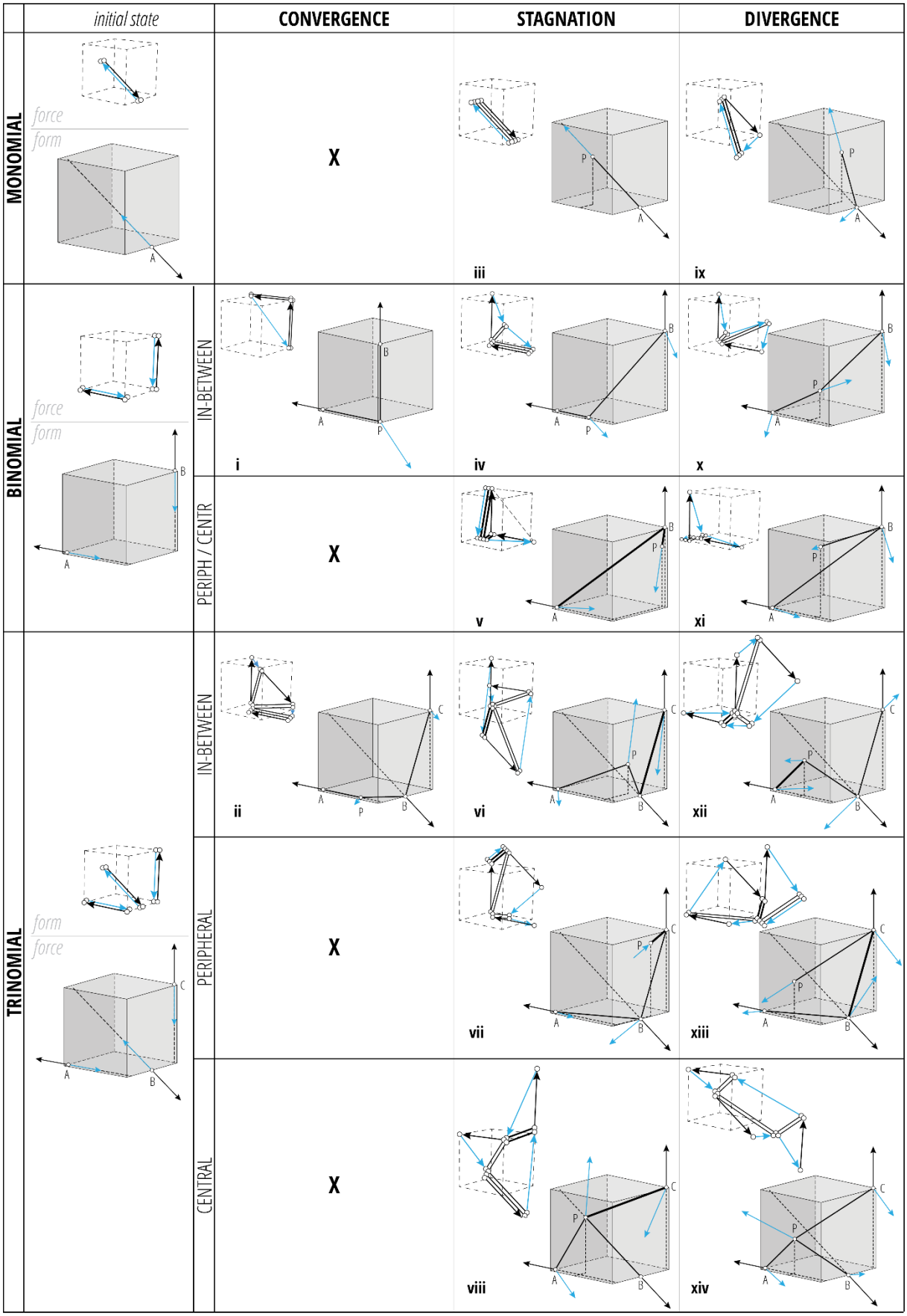


Fig. 17: Transformation step for all combinations of entropy rates, numbers of interim forces and topological configurations

2.6.3 Transformation process

The transition to a complete network implies the successive application of the transformation step as described before. Each transformation step consists of five stages illustrated at Fig. 18. Sets of different inputs control the execution of each stage.

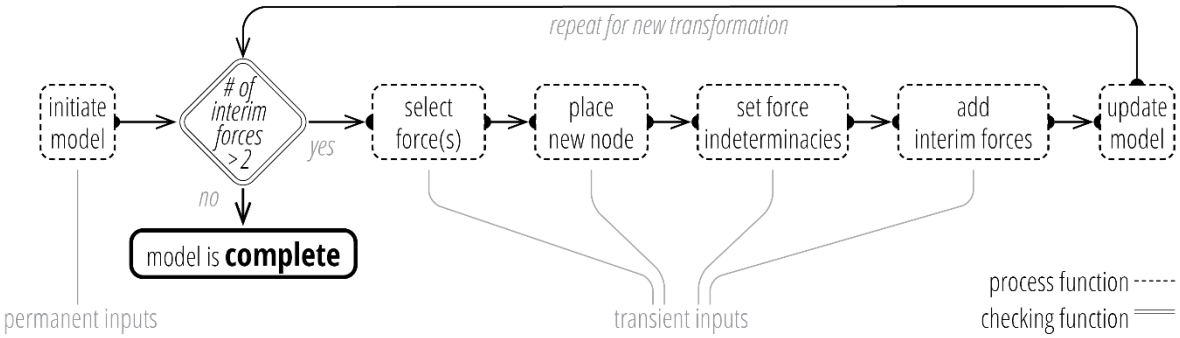


Fig. 18: Schematic workflow of successive policy-based transformations

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.

2.6.3.1 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 build 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 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 must lie on the same line of action, they must have the same length and they must be 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 model.

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

2.6.3.2 Select interim force(s) and bars 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. 16]. The selection is operated either explicitly or 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. If the interim forces are selected explicitly, an explicit definition of the intended topological configuration of the bars to create must be provided by the designer. Otherwise, the desired topological configuration is embedded into the force(s) selection rule.

2.6.3.3 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 by means of a *node placement rule* and imposes the geometry of the growth.

2.6.3.4 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 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.

2.6.3.5 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.

2.6.3.6 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.

3 Implementation of Policy-based Exploration of Equilibrium Representations (PEER)

This part proceeds with the detailed description of the implementation and the illustration of different behaviors that highlight the impact of the main input parameters to the generative process. The part concludes with the presentation of the computational tool that implements the theory and a series of application studies that demonstrate indicative design candidates for different design domains, stretching into two and three dimensions.

3.1 Computational implementation

A schematic description of the stages has already been provided [Fig. 18]. This section describes in depth the undergoing algorithmic operations of every recursive stage and occasionally supported by pseudocode.

3.1.1 Overview

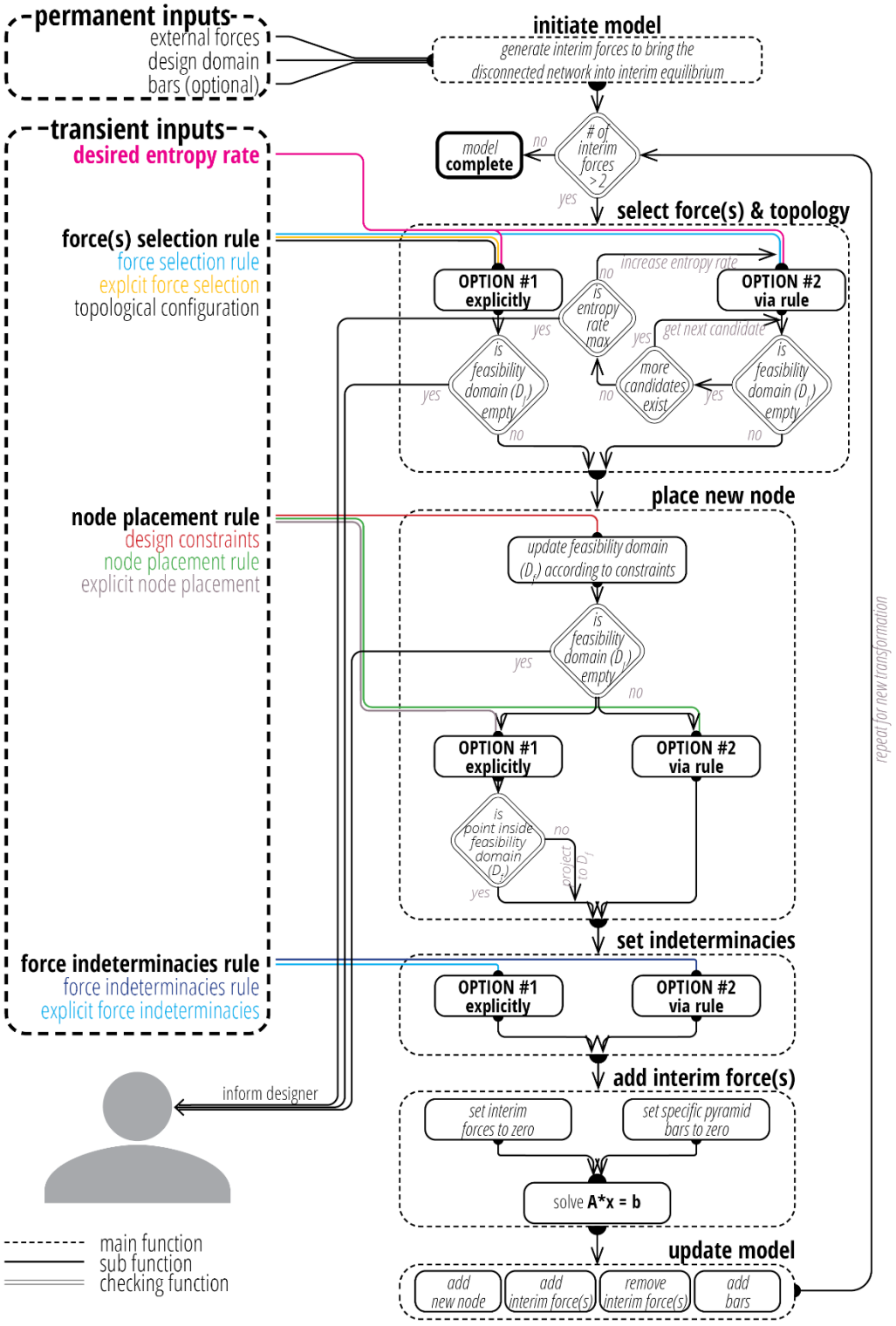


Fig. 19: Algorithmic workflow of successive policy-based transformations

Fig. 19 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 by means of a rule. At every stage that receives explicit input, checks are carried out in order 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.

3.1.2 Input parameters

Section 2.1 describes the value of parameterization in the scope of parametric design. As a parametric framework, PEER exploits different types of input parameters, according to their functionality within the algorithmic workflow. Two of their features are investigated: (a) the *definition type* – i.e., how it is described; numerically or by means of a rule [section 2.1.2] - and (b) the *definition frequency* – i.e., how often a parameter is (re-)defined throughout the process.

Regarding the definition type, both *numerical* and *policy-based* input parameters are considered. As the name of the framework indicates, extra weight is put on policy-based parameters, without excluding the use of numerical values as supplementary inputs. Specifically, numerical parameters accurately describe all four rules.

Regarding the definition frequency, both *permanent* and *transient* input parameters are considered in the entire workflow. The execution frequency of each stage of the process controls the definition frequency of each associated parameter.

3.1.3 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 build 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.

3.1.4 Transient parameters

This section exemplifies how different sets of numerical inputs support the policy definition. Each transformation is the result of a policy which describes design intentions in a descriptive way. Its definition goes beyond numerical input parameters. As it has already been described, four rules define a policy as shown at Fig. 20. The next sections each corresponds to the notion of each rule.

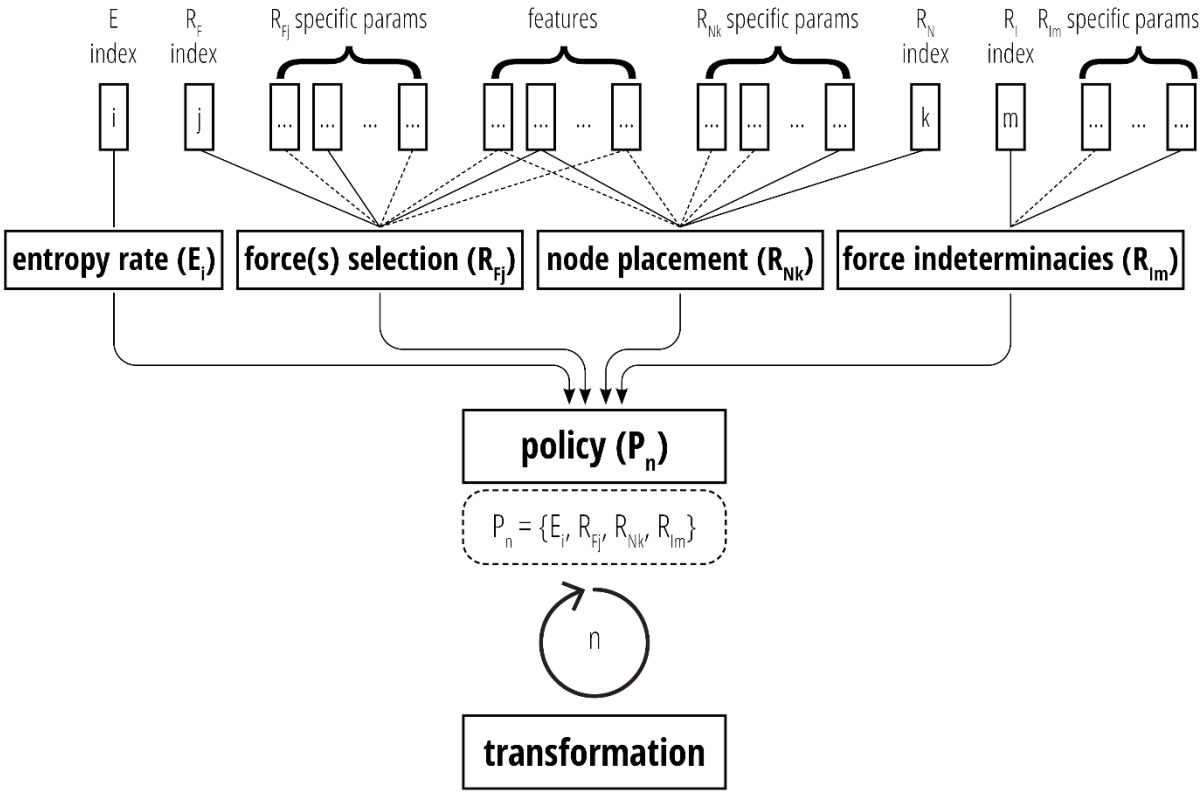


Fig. 20: The set of transient parameters that describe the rules and subsequently the policy which leads to the network transformation

3.1.4.1 Entropy rate

Entropy rate is a balanced ternary system that controls the impact of a transformation on the model. Specifically, it measures the speed of the growth - i.e., the number of interim forces to add or delete. Its definition consists of a selection of a value ***i*** out of a pool of three options {-1, 0, 1}. The meaning of each value is described below:

- **convergence (-1):** decrease the total number of interim forces and help the model converge
- **stagnation (0):** retain the total number of interim forces and let the model stagnate
- **divergence (1):** increase the total number of interim forces and help the model diverge

Its use as an input parameter is associated with three stages which it influences. In *select force(s) & bar topology* stage, it indicates the minimum number of forces to choose – i.e., if $E = -1$ the selection of monomials should be excluded, and binomials or trinomials should be considered instead. In *place new node* stage, the entropy rate along with the selected forces constrains the geometric domain [section 3.1.5.2] where the new node is introduced – i.e., if $E = -1$ there is a single location for the new node. In *add interim forces* stage it imposes the introduction of interim forces to retain the static equilibrium of the nodes – i.e., if $E = 1$ a non-zero interim force is introduced for all involved nodes.

3.1.4.2 Force(s) selection rule

It controls the starting point of the network growth and indicates the set of forces to eliminate through the transformation. Its definition consists of a selection of a rule ***j*** out of a pool of finite options [Appendix A.2]. Each rule describes geometric or other criteria, used to sort the list of feasible sets of forces. General examples of such criteria are:

- proximity of anchor points
- proximity of anchor points, on average, to existing bars / design domain boundary / design domain centroid / designer-defined location
- angle of the introduced bars
- age in the generative process
- randomness
  etc.

Its use as an input parameter is associated with the *select force(s) & bar topology* stage. In the beginning of this stage all sets of one, two or three interim forces in the pool of all current interim forces are computed as potential sets of interim forces and they are stored in a list. During the pairing process, compatibility with the desired entropy rate is checked. Additionally, each force selection rule integrates the topological configuration of the introduced bars. Therefore, the rule itself filters out sets of interim forces that do not comply with the design entropy rate and those that do not satisfy specific connectivity constraints – i.e., the construction of bars between the forces anchor points is interrupted by voids. The rule selection is independent of the location of the *P* node and thus connectivity to *P* is not checked yet.

All feasible sets of forces are ranked according to the ***j*** rule. The sorted list is travelled starting from the best ranked set of forces. The size of the feasibility domain [section 3.1.5.1] it creates is checked. If empty the group of forces is deemed unfeasible and the list travelling continues until a feasible set of interim forces is found. When the list of feasible sets of forces gets empty, the entropy rate increases and the generation, ranking, sorting and feasibility checking of new sets of forces is repeated. The final selection of interim forces is guaranteed compatible because of a number of checks operated internally by the algorithm [Fig. 21]. If $E = 1$ and the list of sets of forces is empty, the designer is informed accordingly, and the process terminates.

Some of these rules require further parameters to be fully described. For example, for *random* rules a seed number is required, for *proximity to designer-defined location* a set of coordinates is expected etc. Some other rules, require additional information (features), like the existing network bars, the design domain etc.

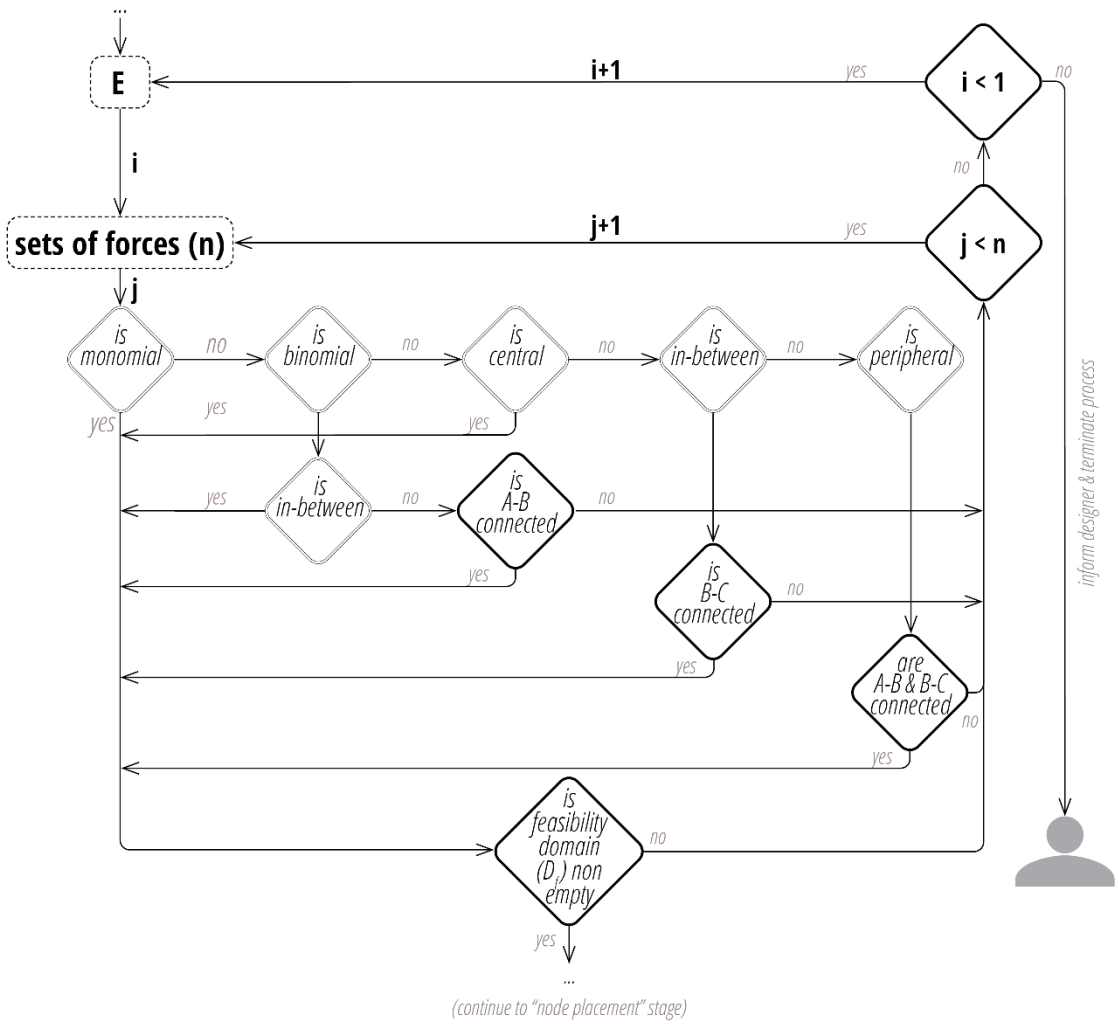


Fig. 21: Internal connectivity checks while creating monomials, binomials, trinomials.

The force selection is managed by rules but for a course of a single transformation, explicit selection of forces is possible too. If explicit, the selection needs the index of the interim forces and the choice of topological configuration that defines the topology of the newly created bars. After carrying out the same checks, if the set of inputs is deemed non-feasible, a new one is directly requested from the user, as no other alternative sets of forces exist.

3.1.4.3 Node placement rule

It controls the geometry of the growth and indicates the position of the new node P. Its definition consists of a selection of a rule $\boldsymbol{k}$ out of a pool of finite options [Appendix A.3]. Each rule describes custom geometric sub-domains to constraint its location. General examples of such rules are:

- the position of P that leads to the smallest sum of bar lengths;
- the position of P that is the closest to the geometric center of all nodes in the system;
- any random position of P that is not further than a length l from all force anchor points";
- any random position of P that lies on a given axis;
- any random position of P that lies between custom X, Y, Z bounds;
  etc.

Its use as an input parameter is associated with the *place new node* stage and similarly to the *force(s) selection & bar topology* stage, the node placement is managed by rules. For a course of a single

transformation, explicit selection of the location is possible too. If explicit, the node location is described via cartesian or homogeneous coordinates. If the explicit location is not located within the feasibility domain, the closest projection to it is computed and chosen for P.

However, the actual position of P is constrained within a subset of the design domain, named *feasibility domain*. The latter ensures compliance with the entropy rate (otherwise the network does not retain static equilibrium), the topological configuration and the expected connectivity (otherwise bars interrupted by voids cannot be introduced) and any other custom constraints requested by the designer. Section 3.1.5 describes the mechanisms to compute the feasibility domain for each entropy rate, topological configuration and number of interim forces, as well as other supportive domains (entropy rate domain, constructability domain and auxiliary domain).

Like few of the *force selection rules*, some *node placement rules* require supplementary parameters to be fully described. For example, for *random* rules a seed number and an axis or the bounds of a domain are further required etc. Some other rules, require additional info (features), like the existing network bars, the design domain etc.

3.1.4.4 Force indeterminacies rule

It controls the structural behavior of the growth and indicates the type (compression/tension) and magnitude of the axial forces developed along the bars, when $E = 0$ or $E = 1$. In these cases, despite knowing the exact location of node P, the calculation of the interim forces to be introduced is not feasible. In other words, the system of equilibrium equations is indeterminate. *Force indeterminacies rules* reduce the indeterminacies by imposing the force magnitudes in the bars and making the system determinate. Its definition consists of a selection of a rule $\boldsymbol{m}$ out of a pool of finite options [Appendix A.4]. General examples of such rules are:

- material utilization within custom bounds;
- randomness;
  etc.

Its use as an input parameter is associated with the *set indeterminacies* stage and provide additional design freedom to the designer expressed as force magnitudes in the new bars. Like the other two rules (*force(s) selection rule* and *node placement rule*) the force indeterminacies can be described explicitly or by means of a rule. If explicit, their use is not bounded to a single transformation, contrary to the explicit definition of force(s) selection rules and node placement rules.

Some of these rules require further parameters to be fully described. For example, for *random* rules a seed number is required, for *material utilization* the bounds of a custom domain are expected etc.

3.1.5 Domains

The final location of node P, as opposed to the desired location, is imposed by the transformative policy and thus depends on multiple parameters. Specifically, its location must be such that:

- it maintains static equilibrium (subject to the chosen set of forces) – i.e., convergence on two interim forces whose lines of action intersect at a point outside the design domain is not feasible.
- it allows constructability of the necessary bars (subject to the chosen topological configuration) – i.e., the creation of bars that cross a void in the design domain is not possible [Fig. 22, constructability domain]
- it complies with the desired change of entropy (subject to the chosen entropy rate) – i.e., the orientation of adjacent bars and the developed force magnitude are such that no interim force is

needed to maintain static equilibrium of A when stagnation is aimed for an in-between binomial [Fig. 22, entropy rate domain]

- it satisfies additional constraints – i.e., the location respects the chosen bar length bounds

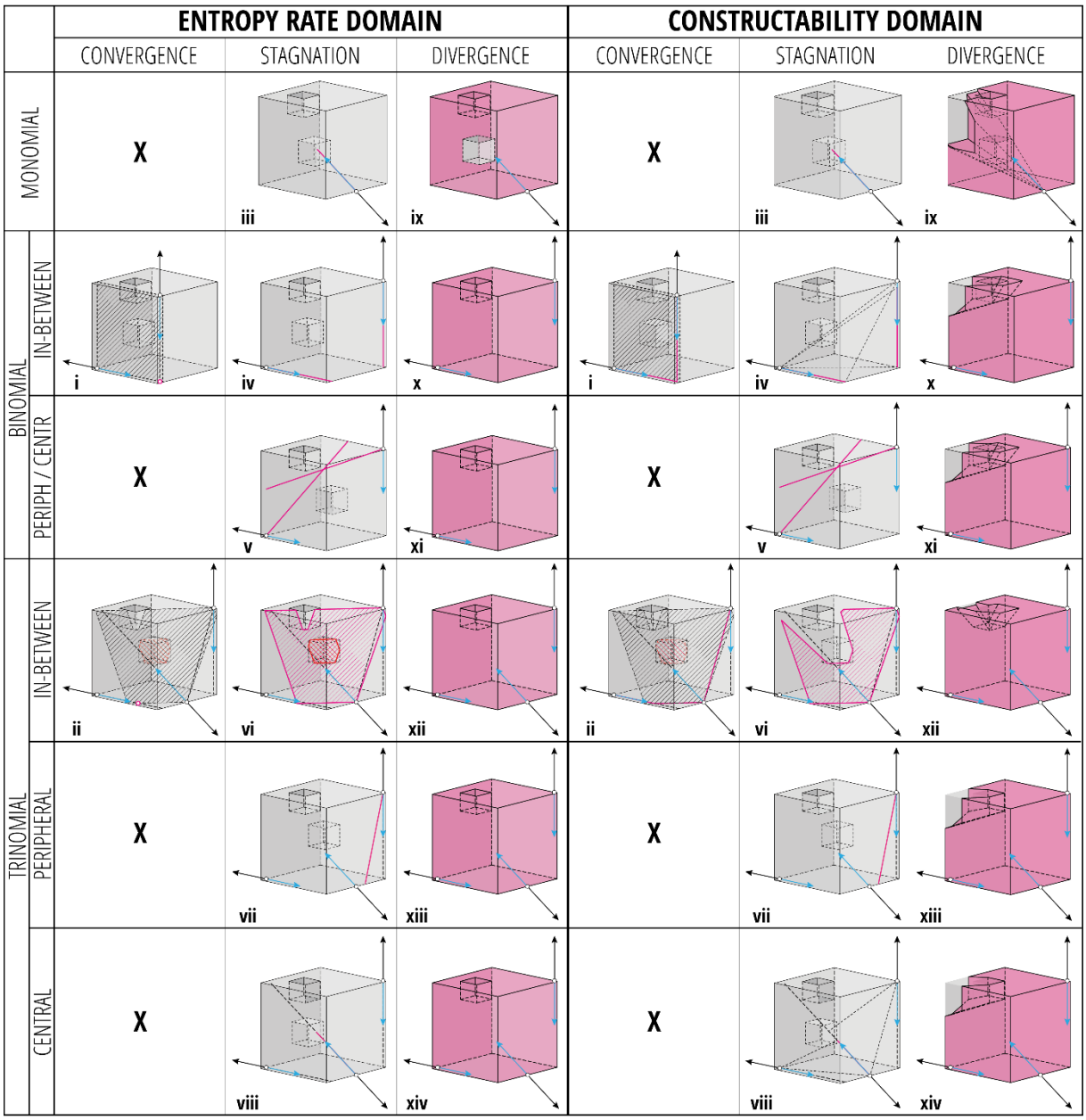


Fig. 22: Typologies of entropy rate and constructability domains. Black arrows are applied loads or reactions, cyan arrows are interim forces, all circumscribed within a primitive design domain (grey). Magenta regions on the left are the intersections of the design domain (a non-convex solid with a void) 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

3.1.5.1 Feasibility domain

The list of these requirements is satisfied by expecting each new node P to lie within a *feasibility domain* (D_f). If the node placement rule introduces node P outside of this domain, its projection to the domain is taken for P. Namely, the D_f is the geometric domain that contains all feasible positions of P. A non-empty D_f makes the choice of interim forces and the transformation feasible too. If empty, the choice of entropy rate and/or the interim forces and/or the additional constraints must be reevaluated.

The D_f is the Boolean intersection between the *entropy rate domain* (D_e), the *constructability domain* (D_c) and the *auxiliary domain* (D_a, if any), each described in a separate section below. Out of the four input choices that define the transformation policy, the *entropy rate*, the *force(s) selection rule* and the *node placement rule* directly affect the size of the feasibility domain, whereas the *force indeterminacies rule* has no impact on it.

3.1.5.2 Entropy rate domain

The *entropy rate domain* describes a geometric domain of positions for P that ensure the feasibility of the chosen entropy rate when a transformation policy is applied on a set of interim forces for a specific topological configuration. The mathematical expression to describe the geometric domain for each case is provided in Table 1. Its size ranges from a single point – i.e., in the case of convergence – to a segment or a flat enclosed region – i.e., in the case of stagnation – or a volume equal to the full design domain – i.e., in the case of divergence. A visual illustration of the geometric domain per case is provided in Fig. 22 (left).

		convergence	**stagnation**	**divergence**
monomial		-	$\overrightarrow{f_A}$ *(line of action)*	D_d
binomial	*in-between*	$\boldsymbol{P} = \begin{bmatrix} \boldsymbol{x_A} \\ \boldsymbol{y_A} \\ \boldsymbol{z_A} \end{bmatrix} \pm \frac{\lVert \overrightarrow{\boldsymbol{f_B}} \times \overrightarrow{\boldsymbol{AB}} \rVert}{\lVert \overrightarrow{\boldsymbol{f_A}} \times \overrightarrow{\boldsymbol{f_B}} \rVert} \overrightarrow{\boldsymbol{f_A}}$	$\overrightarrow{f_A}$ and $\overrightarrow{f_B}$ *(line of action)*	
	peripheral central	-	$\overrightarrow{AB} + \overrightarrow{f_A}$ and $\overrightarrow{AB} - \overrightarrow{f_B}$ *(line of action)*	
trino-mial	*in-between*	$\begin{bmatrix} \overrightarrow{f_A} & -\overrightarrow{f_B} & -\overrightarrow{BC} \end{bmatrix} \begin{bmatrix} t \\ u \\ v \end{bmatrix} = \overrightarrow{AB}$ $\mathrm{P} = \begin{bmatrix} x_P \\ y_P \\ z_P \end{bmatrix} = \begin{bmatrix} x_A \\ y_A \\ z_A \end{bmatrix} + \overrightarrow{f_A} \times \begin{bmatrix} t \\ u \\ v \end{bmatrix}$	$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \begin{bmatrix} x_B \\ y_B \\ z_B \end{bmatrix} + \lambda \overrightarrow{f_B} + \mu \overrightarrow{BC}$ *(plane equation;λ and μ take all values to give all positions on it)*	
	peripheral	-	$\overrightarrow{BC} - \overrightarrow{f_C}$ *(line of action)*	
	central	-	$\overrightarrow{f_B}$ *(line of action)*	

Table 1: Mathematical expression of entropy rate domain for each case of entropy rate, set of interim forces and topological configuration

3.1.5.3 Constructability domain

The *constructability domain* describes a geometric domain of positions for P that ensure the uninterrupted constructability of the adjacent to it bars. Namely, every new bar connecting nodes A, B or C with P is fully contained within the design domain – i.e., does not intersect its boundaries. Its size depends on the entropy rate, the number of interim forces and the topological configuration of the new bars. A visual illustration of the geometric domain per case is provided in Fig. 22 (right).

The D_c computation is supported by Isovist algorithms in two and three dimensions, which can be time consuming due to the Boolean intersections they rely on. However, the D_c is only necessary in cases of (a) non-convex design domains and (b) voids present within the design domain regardless of the flat regions and/or bounded volumetric representations being convex or not. This simple observation implies that for all other cases the generation of the D_e is sufficient for the feasible transformation and the D_f is equivalent to the D_e.

For convergence on binomials and trinomials, the computation of the D_e is necessary. For both sets of forces (binomials and trinomials) uninterrupted connectivity with nodes A and B is checked by confirming that P lies on the f_A and f_B lines of action and visibility of P from A and B is uninterrupted.

The tables below provide the pseudo code to compute the D_c for all other cases and entropy rates.

ALGORITHM 1: STAGNATION MONOMIAL

input: *interim force ($\overrightarrow{f_A}$, A), design domain (D_d)*
output: *segment*

if the line of action of $\overrightarrow{\boldsymbol{f_A}}$ *is interrupted by voids*
    return the segment that contains ***A***
else
    return the segment bounded by $\boldsymbol{D_d}$

Table 2: Pseudocode to compute the constructability domain for stagnation on a monomial

ALGORITHM 2: STAGNATION ON BINOMIAL (IN-BETWEEN)

input: *set of interim forces ($\overrightarrow{f_A}$, $\overrightarrow{f_B}$, A, B), design domain (D_d)*
output: *segment*

if the line of action of $\overrightarrow{\boldsymbol{f_A}}$ *is interrupted by voids*
    get the segment that contains ***A*** *(seg)*
else
    get the segment bounded by $\boldsymbol{D_d}$ *(seg)*
if $\boldsymbol{D_d}$ *is planar*
    compute Isovist in 2D for ***B*** *on the plane of* $\boldsymbol{D_d}$ *(polyIso)*
else
    build a surface between ***seg*** *and* ***B*** *(surf)*
    compute the intersection between ***surf*** *and* $\boldsymbol{D_d}$ *(inter)*
    if ***inter*** *is empty*
        return ***seg***
    else
        compute Isovist in 2D for ***B*** *on the plane of* ***inter*** *(polyIso)*
return the intersection between ***polyIso*** *and* ***seg***

Table 3: Pseudocode to compute the constructability domain for stagnation on an in-between binomial

ALGORITHM 3: STAGNATION ON BINOMIAL (PERIPHERAL-CENTRAL)

input: *set of interim forces ($\overrightarrow{f_A}$, $\overrightarrow{f_B}$, A, B), design domain (D_d)*
output: *segment*

will ***A*** *connect to P*
    get the line of action of $\overrightarrow{AB} + \overrightarrow{f_A}$ $(\vec{f})$
else
    get the line of action of $\overrightarrow{AB} - \overrightarrow{f_B}$ $(\vec{f})$
if the line of action of $\vec{\boldsymbol{f}}$ *is interrupted by voids*

*return the segment that contains **A** (or **B** respectively) (seg)*

else

return the segment bounded by $\boldsymbol{D_d}$

Table 4: Pseudocode to compute the constructability domain for stagnation on a peripheral/central binomial

ALGORITHM 4: STAGNATION ON TRINOMIAL (IN-BETWEEN)

input: *set of interim forces ($\overrightarrow{f_A}$, $\overrightarrow{f_B}$, $\overrightarrow{f_C}$, A, B, C), design domain (D_d)*

output: *polygon*

*compute Isovist in 3D for **A** (brepIso)*

*compute Isovist in 2D for **B** on **plane** (polyIso)*

get the plane of the respective D_e(plane)

*compute the intersection between **brepIso** and **plane** (inter)*

if is not empty

*return the intersection between **inter** and **polyIso***

else

return NONE

Table 5: Pseudocode to compute the constructability domain for stagnation on an in-between trinomial

ALGORITHM 5: STAGNATION ON TRINOMIAL (PERIPHERAL)

input: *set of interim forces ($\overrightarrow{f_A}$, $\overrightarrow{f_B}$, $\overrightarrow{f_C}$, A, B, C), design domain (D_d)*

output: *segment*

get the line of action of $\overrightarrow{BC} - \overrightarrow{f_C}$ ($\vec{f}$)

if the line of action of $\vec{\boldsymbol{f}}$ is interrupted by voids

*return the segment that contains **C***

else

return the segment bounded by $\boldsymbol{D_d}$

Table 6: Pseudocode to compute the constructability domain for stagnation on a peripheral trinomial

ALGORITHM 6: STAGNATION ON TRINOMIAL (CENTRAL)

input: *set of interim forces ($\overrightarrow{f_A}$, $\overrightarrow{f_B}$, $\overrightarrow{f_C}$, A, B, C), design domain (D_d)*

output: *segment*

if the line of action of $\overrightarrow{\boldsymbol{f_B}}$ is interrupted by voids

*get the segment that contains **B** (seg)*

else

get the segment bounded by $\boldsymbol{D_d}$ (seg)

*build a surface between **seg** and **A** (surfA)*

*build a surface between **seg** and **C** (surfC)*

*compute the intersection between **surfA** and $\boldsymbol{D_d}$ (interA)*

*compute the intersection between **surfC** and $\boldsymbol{D_d}$ (interC)*

*if **interA** AND **interC** are empty*

*return **seg***

create an empty list of segments (segsA)

*If **interA** is not empty*

*compute all Isovist in 2D for **A** on **interA** (polyIsoA)*

*If **polyIsoA** is not empty*

*explode **polyIsoA** to segments*

else

return NONE

else

*add **seg** to **segsA***

create an empty list of segmets (segsB)

*If **interB** is not empty*

*compute all Isovist in 2D for **B** on **interB** (polyIsoB)*

*if **polyIsoB** is not empty*

*explode **polyIsoB** to segments*

else

Return NONE

else

*add **seg** to **segsB***

*merge **segsA** and **segsB** (segs)*

*iterate through **segs** and find the segments that are not coincident to **seg** (segs)*

*return that segment that is simultaneously coincident to all segments in **segs***

Table 7: Pseudocode to compute the constructability domain for stagnation on a central trinomial

ALGORITHM 7: DIVERGENCE ON MONOMIAL

input: *interim forcs ($\overrightarrow{f_A}$, A), design domain (D_d)*

output: *segment*

*return Isovist in 3D for **A***

Table 8: Pseudocode to compute the constructability domain for divergence on a monomial

ALGORITHM 8: DIVERGENCE ON BINOMIAL (IN-BETWEEN) &TRINOMIAL (IN-BETWEEN)

input: *set of interim forces ($\overrightarrow{f_A}$, $\overrightarrow{f_B}$,, A, B) or ($\overrightarrow{f_A}$, $\overrightarrow{f_B}$, $\overrightarrow{f_C}$, A, B, C), design domain (D_d)*

output: *segment*

*compute Isovist in 3D for **A** (brepIsoA)*

*compute Isovist in 3D for **B** (brepIsoB)*

*return the intersection between **brepIsoA** and **brepIsoB***

Table 9: Pseudocode to compute the constructability domain for divergence on an in-between binomial or trinomial

ALGORITHM 9: DIVERGENCE ON BINOMIAL (PERIPHERAL / CENTRAL)
input: *set of interim forces ($\overrightarrow{f_A}$, $\overrightarrow{f_B}$, A, B), design domain (D_d)*
output: *segment*
return Isovist in 3D for ***B***

Table 10: Pseudocode to compute the constructability domain for divergence on a peripheral/central binomial

ALGORITHM 10: DIVERGENCE ON TRINOMIAL (PERIPHERAL)
input: *set of interim forces ($\overrightarrow{f_A}$, $\overrightarrow{f_B}$, $\overrightarrow{f_C}$, A, B, C), design domain (D_d)*
output: *segment*
return Isovist in 3D for ***C***

Table 11: Pseudocode to compute the constructability domain for divergence on a peripheral trinomial

ALGORITHM 11: DIVERGENCE ON TRINOMIAL (CENTRAL)
input: *set of interim forces ($\overrightarrow{f_A}$, $\overrightarrow{f_B}$, $\overrightarrow{f_C}$, A, B, C), design domain (D_d)*
output: *segment*
compute Isovist in 3D for ***A*** *(brepIsoA)*
compute Isovist in 3D for ***B*** *(brepIsoB)*
compute Isovist in 3D for ***C*** *(brepIsoC)*
return the intersection between ***brepIsoA****,* ***brepIsoB*** *and* ***brepIsoC***

Table 12: Pseudocode to compute the constructability domain for divergence on a central trinomial

3.1.5.4 Auxiliary domain

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 (e.g., longer bars decrease the number of nodal connections), and where l_{max} may relate to buckling limits [91]. Geometrically speaking, the D_{con} is the intersection of two hollowed spheres, with inner and outer radiuses equal to l_{min} and l_{max} respectively [Fig. 23]. It therefore only depends on the choice of interim forces.

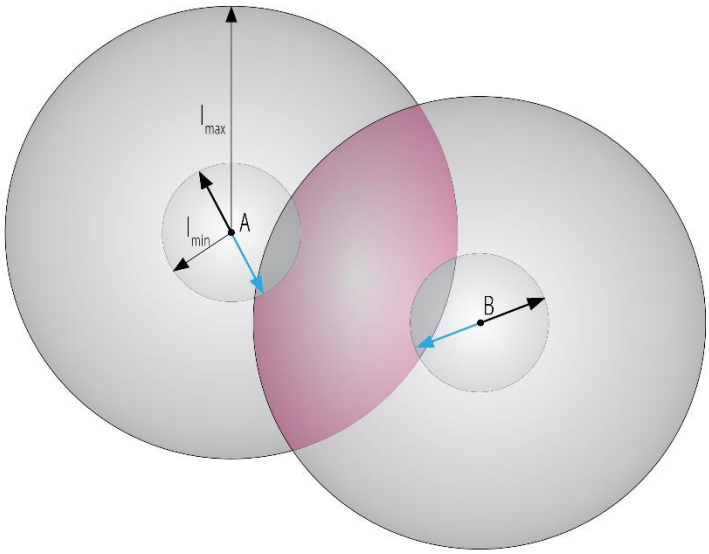


Fig. 23: Example of a domain constraining bar lengths (in magenta)

3.1.6 Algebraic solving

The fourth stage in the transformation loop [section 2.6.3.5] guarantees static equilibrium in the model by introducing interim forces. Their direction and magnitude are calculated at this stage by solving equilibrium equations. At the same time, the developed forces along the introduced bars are calculated too. When the system is indeterminate, the force indeterminacies rule imposes the magnitude for as many bars as the degree of the indeterminacy.

Every transformation step considers a maximum of three nodes, already available in the network, plus a new node P that is introduced. Together with the interim forces that keep them in static equilibrium they form a sub-network. If at the end of each transformation step static equilibrium is ensured for the sub-network, then static equilibrium is ensured for the entire network.

Because static equilibrium is checked for every single node, computing translational equilibrium 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 – i.e., x, y, z - the translational equilibrium of the sub-network is ruled by 12 equations. The system of linear equations to solve is of the form $\boldsymbol{A} \times \boldsymbol{x} = \boldsymbol{b}$, where:

- $\boldsymbol{A}$ is a 12×18 matrix describing the topology and geometry of the tetrahedron [Fig. 16];
- $\boldsymbol{x}$ is a 18×1 vector containing the x, y, z components of the interim forces as well as the force magnitudes of the introduced bars;
- $\boldsymbol{b}$ is a 12×1 vector containing the x, y, z components of the external to the sub-network forces;

More precisely:

$$\begin{bmatrix} \mathbf{I} & \overline{\boldsymbol{\Delta}_{\mathbf{BA}}} & \overline{\mathbf{0}} & & \overline{\boldsymbol{\Delta}_{\mathbf{CA}}} & \overline{\boldsymbol{\Delta}_{\mathbf{PA}}} & \overline{\mathbf{0}} & \overline{\mathbf{0}} \\ & \overline{\boldsymbol{\Delta}_{\mathbf{AB}}} & \overline{\boldsymbol{\Delta}_{\mathbf{CB}}} & & \overline{\mathbf{0}} & \overline{\mathbf{0}} & \overline{\boldsymbol{\Delta}_{\mathbf{PB}}} & \overline{\mathbf{0}} \\ & \overline{\mathbf{0}} & \overline{\boldsymbol{\Delta}_{\mathbf{BC}}} & & \overline{\boldsymbol{\Delta}_{\mathbf{AC}}} & \overline{\mathbf{0}} & \overline{\mathbf{0}} & \overline{\boldsymbol{\Delta}_{\mathbf{PC}}} \\ & \overline{\mathbf{0}} & \overline{\mathbf{0}} & & \overline{\mathbf{0}} & \boldsymbol{\Delta}_{\mathbf{AP}} & \overline{\boldsymbol{\Delta}_{\mathbf{BP}}} & \overline{\boldsymbol{\Delta}_{\mathbf{CP}}} \end{bmatrix} \begin{bmatrix} \overline{\mathbf{t}_{\mathbf{A}}} \\ \overline{\mathbf{t}_{\mathbf{B}}} \\ \overline{\mathbf{t}_{\mathbf{C}}} \\ \overline{\mathbf{t}_{\mathbf{P}}} \\ n_1 \\ n_2 \\ n_3 \\ n_4 \\ n_5 \\ n_6 \end{bmatrix} = \begin{bmatrix} \overline{\mathbf{f}_{\mathbf{A}}} \\ \overline{\mathbf{f}_{\mathbf{B}}} \\ \overline{\mathbf{f}_{\mathbf{C}}} \\ \overline{\mathbf{0}} \end{bmatrix}$$

Where, $\boldsymbol{I}$ is the 12×12 identity matrix, $\overline{\mathbf{0}}$ is the 3×1 null vector, n_k $(k = 1,2,\dots,6)$ are the force magnitudes in the six bars of the tetrahedron [Fig. 16], and for any point $i = [x_i \quad y_i \quad z_i]^T$ or $j = [x_j \quad y_j \quad z_j]^T$ equal to A, B, C or P - i.e., the four vertices of the tetrahedron:

- $\overline{\boldsymbol{\Delta}_{\mathbf{ij}}}$ is the $\left[\frac{x_i - x_j}{l_k} \quad \frac{y_i - y_j}{l_k} \quad \frac{z_i - z_j}{l_k}\right]^T$ difference vector where points $\boldsymbol{i}$ and $\boldsymbol{j}$ are both ends of bar $\boldsymbol{k}$ and $\boldsymbol{l_k}$ is the length of bar $\boldsymbol{k}$;
- $\overline{\boldsymbol{t_i}}$ is the $[t_{i,x} \quad t_{i,y} \quad t_{i,z}]^T$ interim force applied at point $\boldsymbol{i}$;
- $\overline{\boldsymbol{f_i}}$ is the $[f_{i,x} \quad f_{i,y} \quad f_{i,z}]^T$ external force applied at point $\boldsymbol{i}$;

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 steps of the transformation process.

The choice of desired topological configuration not only defines the sequence of the introduced bars but also indicates the bars of the fictitious tetrahedron that actually exist [Fig. 16]. Precisely, up to three bars are introduced in every transformation step. The remaining three to five bars of the tetrahedron do not need to be constructed. Hence, in the x vector their force magnitudes (n_k) are known to be equal to zero. In the same vector, when $E = 0$ or $E = 1$, at least one force magnitude is provided through the force indeterminacies rule to make the system of equations determinate. Additionally, for each node i where no interim force is introduced, based on the entropy rate $\overline{t_i}$ is set as a 3×1 null vector. Consequently, the entire matrix description is simplified. These simplifications significantly reduce the problem size. The unknown new interim forces $\overline{t_i}$ and bar force magnitudes n_k are obtained after inverting matrix $\boldsymbol{A}$.

The next section illustrates through simple bar networks different behaviors of the transformation policy as a result of different choice of rules or other parameters (i.e., shape of design domain).

3.2 Illustrated behaviors

This chapter demonstrates primitive examples of recursive application of multiple transformative steps. Precisely, it illustrates different behaviors of bars network growth as a result of different parameters. First, policy variations are investigated by changing individually each of the four aspects that define a policy (transient parameters). After, the impact of the geometric domain, its form and the presence of voids, are highlighted (permanent parameters). Finally, the impact of additional constraints that bound the bar lengths are demonstrated. All examples consist of five incremental steps starting from a disconnected domain. The exact values used as input parameter for each demonstrated example [Fig. 24 - Fig. 30] is provided in Appendix B.

3.2.1 Policy variations

This section focuses on the variations in the incremental growth of a bars network as a result of different policy definitions. Each of the subsections, investigates one of the four aspects that define a policy. Hence, each time the remaining three aspects are described through identical parameters. The design domain is identical, and no bar length bounds are considered.

3.2.1.1 Entropy rate

The choice of different entropy rate not only controls the speed of the network growth, but also tends to keep the network growth compact when converging. The transformation impact sourcing from the three possible entropy rate options is shown at Fig. 24.

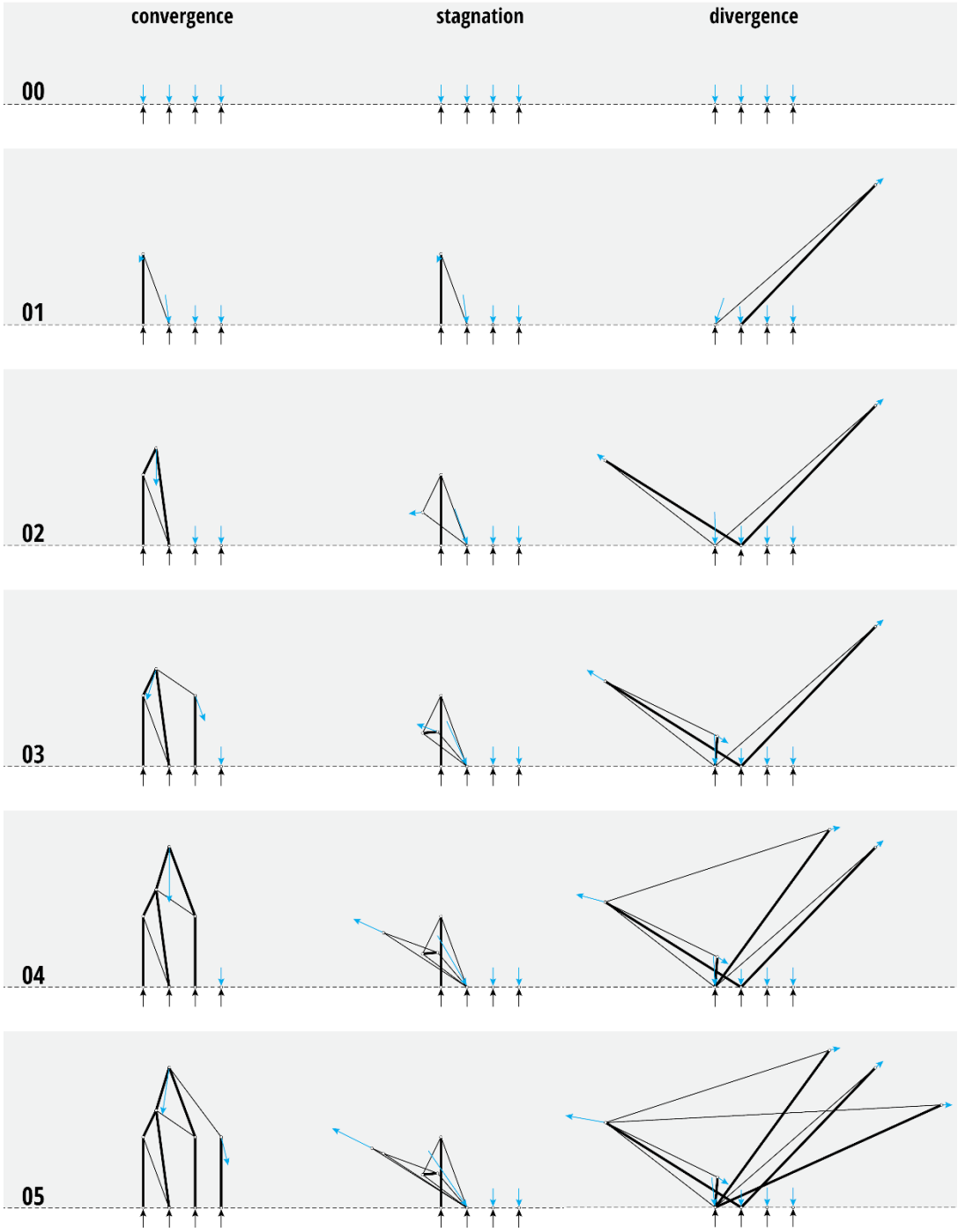


Fig. 24: Five steps evolution of a bars network for different entropy rate choice

3.2.1.2 Force selection

The force selection sets the starting point of the growth and eventually impacts the network topology. Contrary to the ternary choice of entropy rate, there are numerous implicit rules to select sets of interim forces [Appendix A.2]. Consider that for each rule, the number of interim forces ranges from one to three, fact that triples the total number of possible rules. Below, the result of binomial sets of interim forces is demonstrated, when constructed via different force selection rules.

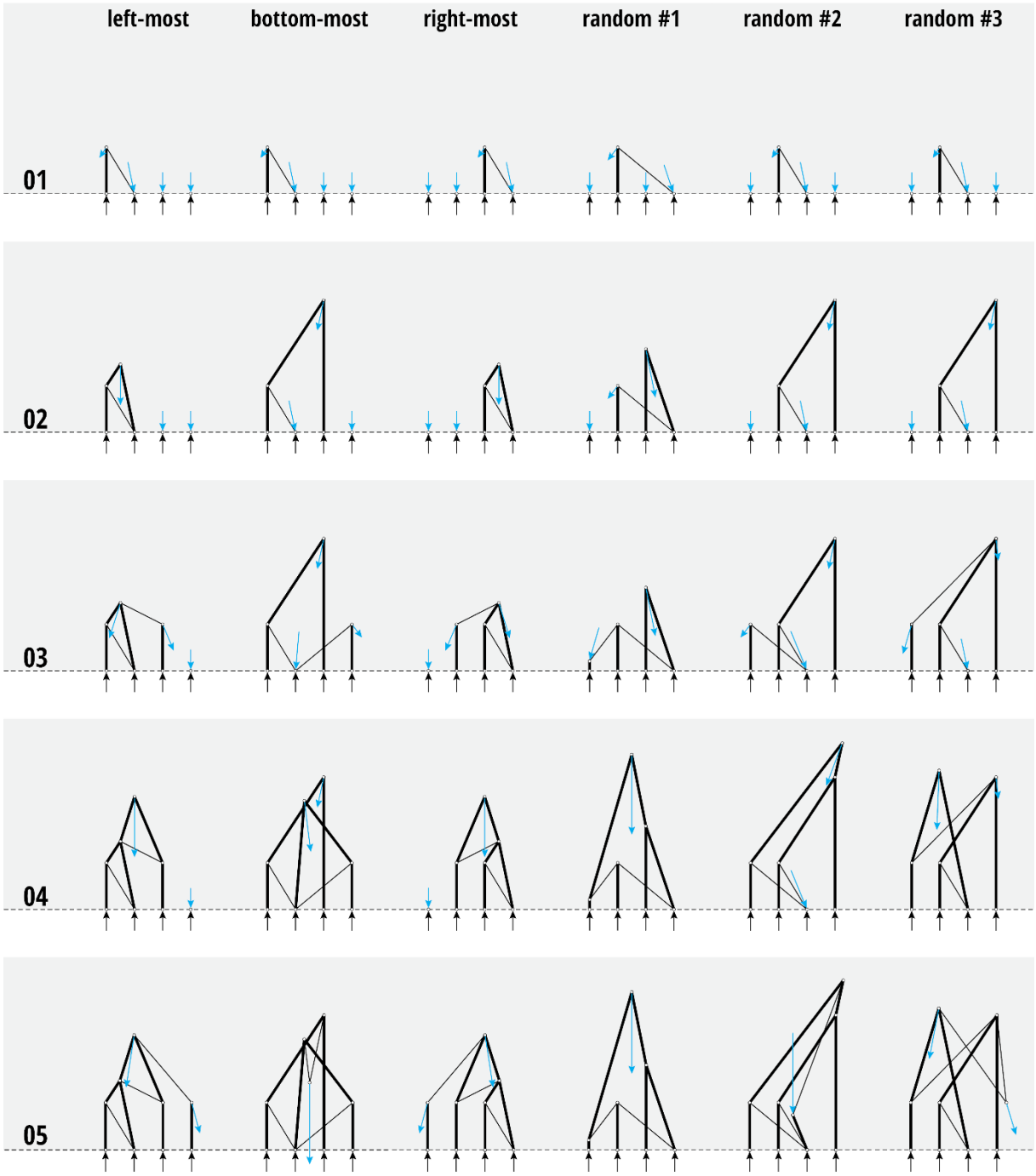


Fig. 25: Five steps evolution of a bars network for different force(s) selection choice

The networks generated when selecting left-most or right-most forces are symmetric to each other. When the choice of forces is made randomly, there is no control over the network topology.

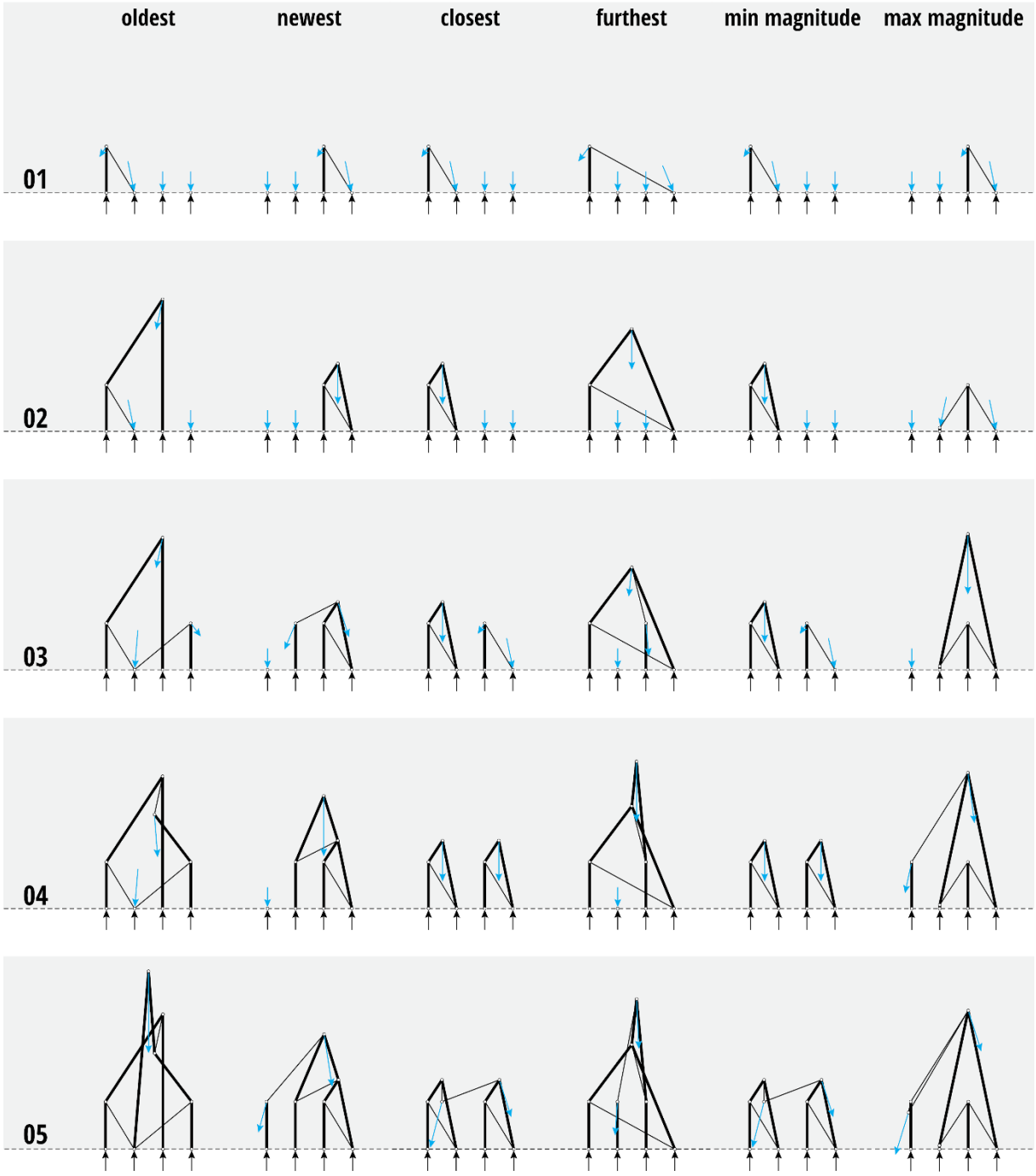


Fig. 26: Five steps evolution of a bars network for different force(s) selection rule

The same topology can be replicated by different force selection rules, like showcased in Fig. 26 where the *closest* interim forces and the ones having the *minimum magnitude* form binomials generate identical networks. This behavior is neither representative of the specific rules nor occurs regularly.

3.2.1.3 Node placement

This is the most influential aspect of the policy definition. When the new node location is not associated with any optimization objective and is not constrained by other aspects – i.e., bar length bounds - it is likely to lead to uncontrolled topologies. Fig. 27 only demonstrates a tiny fraction of the diversity it offers.

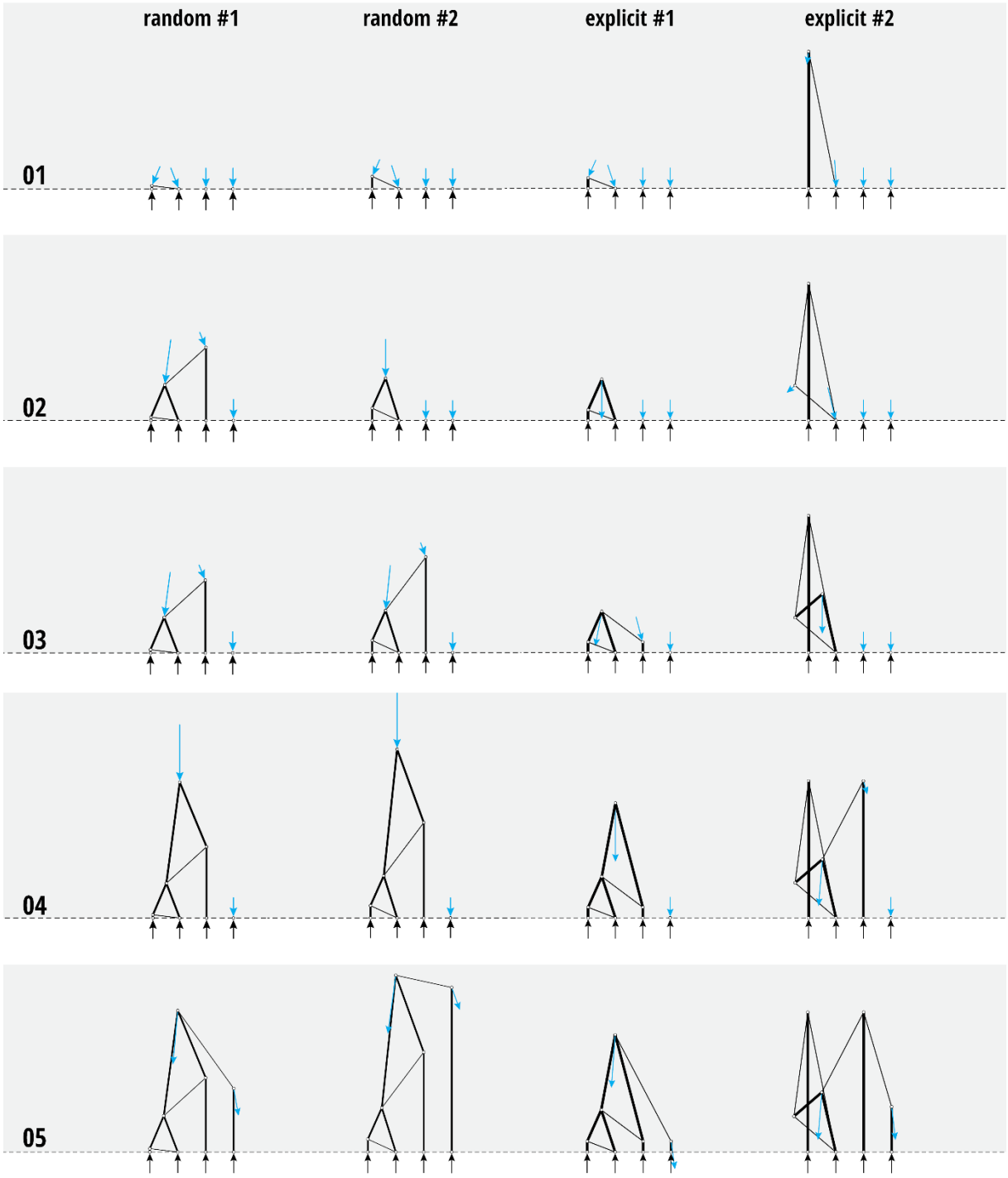


Fig. 27: Five steps evolution of a bars network for different node placement

3.2.1.4 Force indeterminacies

The force indeterminacies influence the structural behavior of the network. and eventually impacts the overall speed of the network growth as well as the network topology. Currently, there are not many implicit rules to define the force indeterminacies [Appendix A.4] but still it is evident that the network topology is highly dependent on them.

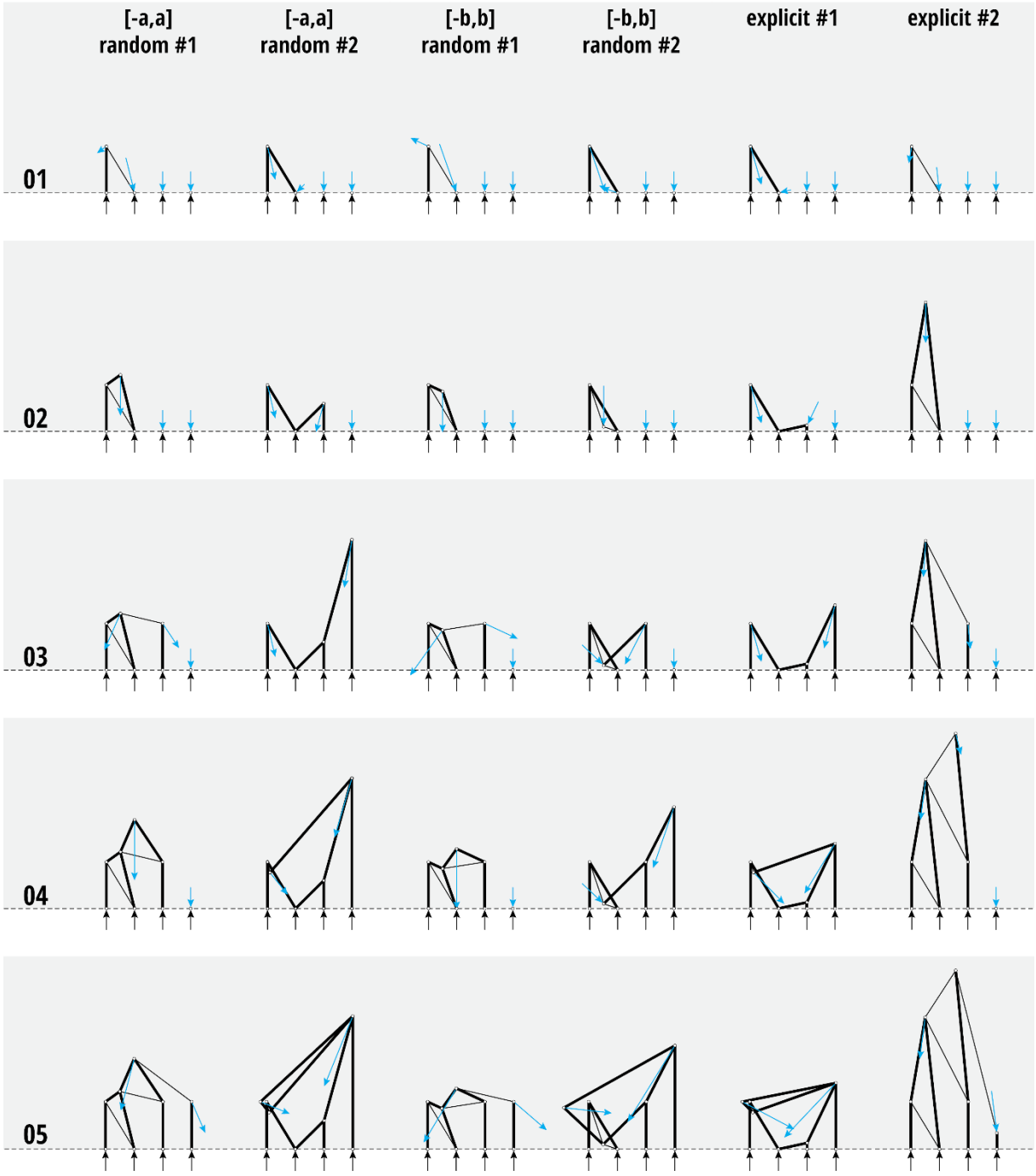


Fig. 28: Five steps evolution of a bars network for different force indeterminacies

3.2.2 Design domain geometry variations

The design domain geometry imposes the computation of the constructability domain [section 3.1.5.3] as voids and non-convex boundaries do not ensure the uninterrupted construction of bars between nodes. Therefore, the final bars network is influenced by such geometric features. Indicative examples of non-convex and voids are demonstrated in Fig. 29.

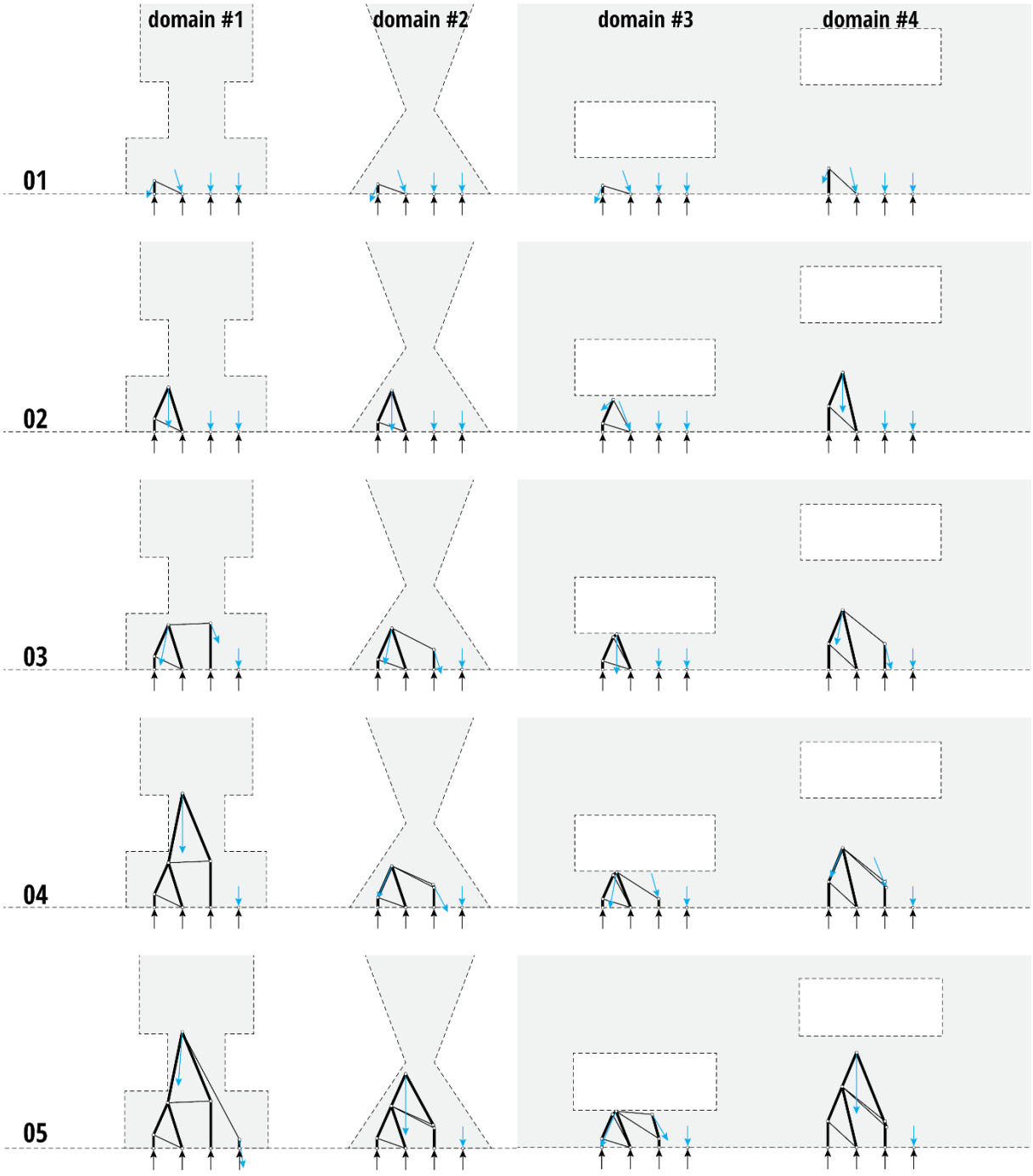


Fig. 29: Five steps evolution of a bars network for different design domains

3.2.3 Constraints addition

The bar length bounds constrain the introduction of too long or too short bars and hence influence the feasibility of certain sets of interim forces to transform the network in specific ways – i.e., convergence of a binomial where node *P* is introduced further than a fixed length is deemed unfeasible, and other sets of interim forces are checked for their feasibility before the entropy rate increases.

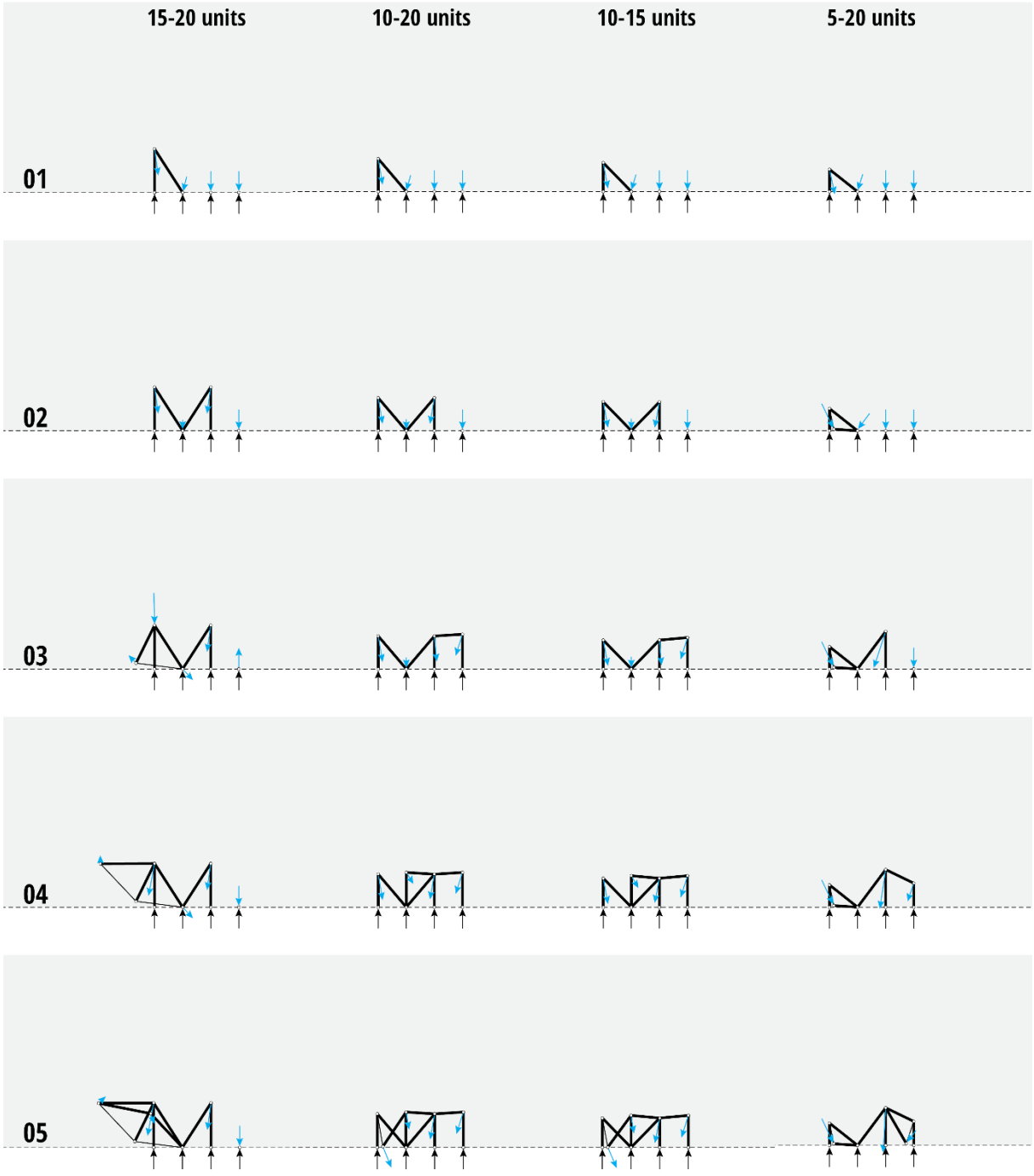


Fig. 30: Five steps evolution of a bars network for different bar length bounds

3.3 Dissemination

The PEER framework is disseminated through the parametric toolkit of *Libra*[2], a plugin operating in Rhinoceros Grasshopper platform, soon to be published in www.food4rhino.com. Besides granting

[2] *Equilibrium* contains a root from the Latin **libra**, meaning *weight* or *balance*.

access to the PEER framework to external users beyond the author himself, *Libra* is a proof of concept and confirms the initial hypothesis about thorough design space exploration through policies. Starting with an overview of the workflow on Grasshopper canvas, this section aims at binding the methodology theory with the provided user interface. Through examples of explicit and implicit rule definitions the section exemplifies how design preferences of future designers are communicated.

3.3.1 Overview

A primitive setup of the transformative design process with PEER is demonstrated below [Fig. 31]. 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 built 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.

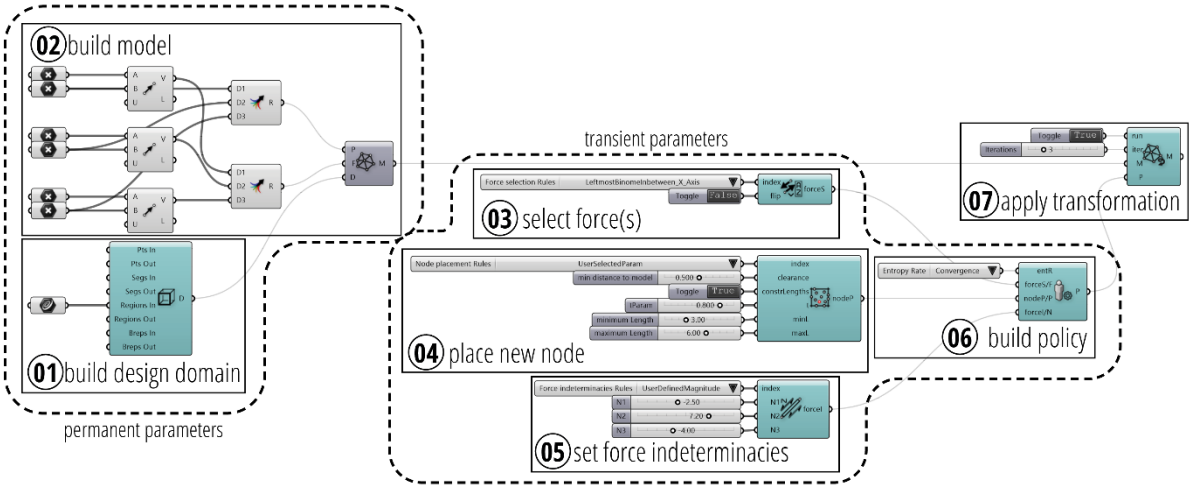


Fig. 31: Grasshopper canvas overview of a complete transformation setup that transforms the interim network for three steps according to a policy defined via rules.

3.3.2 Build design domain

At the very beginning, the design domain (D_d) is defined. Typically, flat closed regions and bounded volumetric representations 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.

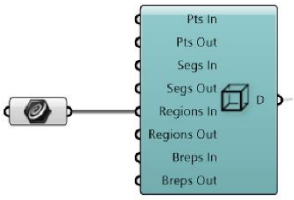


Fig. 32: Build design domain Grasshopper component

INPUT	DESCRIPTION
pts in	collection of points included in D_d *(Point3d)*
pts out	collection of points excluded from D_d *(Point3d)*
segs in	collection of segments included in D_d *(Curve)*
segs out	collection of segments excluded from D_d *(Curve)*
regions in	collection of planar closed regions included in D_d *(Curve)*

regions out	collection of planar closed regions excluded from D_d *(Curve)*
breps in	collection of bounded representations included in D_d *(BRep)*
breps out	collection of bounded representations excluded from D_d *(BRep)*

OUTPUT	DESCRIPTION
D	final D_d to grow the network of bars into *(Domain)*

3.3.3 Build 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.

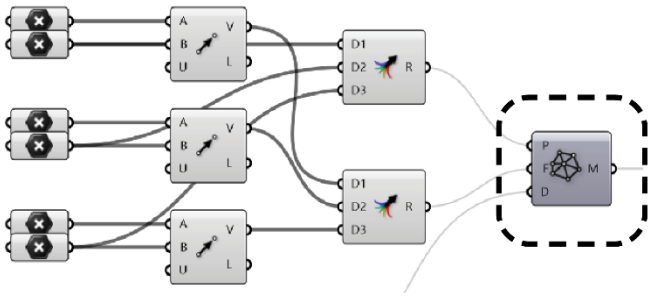


Fig. 33: Build model Grasshopper component

INPUT	DESCRIPTION
P	collection of anchor points for the external force vectors *(Point3d)*
F	collection of external force vectors *(Vector3d)*
D	D_d to grow the network of bars into (*Domain*)

OUTPUT	DESCRIPTION
M	initial model to transform incrementally *(Model)*

The next steps 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. 20]. Hence, the following sections [3.3.4 - 3.3.6] describe steps operated in a random order.

3.3.4 Select force(s)

According to section 3.1.4.2, the selection of forces is possible either explicitly or by means of rules. Fig. 34 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 A.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. 34i], demonstrates the standard configuration of the *Build force selection rule* component. Most of the rules do not require additional parameters for their definition. Therefore, the rule index is enough. Through the second input parameter, the designer optionally flips the order of forces. These two 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. 17v and Fig. 17xi] – and ultimately the feasibility

domain. It is remarkable that the chosen force selection rule includes the desired number of interim forces and the topological configuration of the new bars. In the second example [Fig. 34ii], 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. 34iii], demonstrates the construction of a custom force selection rule, based on the proximity of the force anchor points to a designer-defined point. The number of the provided forces, as well as the topological configuration are explicitly provided too.

Fig. 34: Select force(s) and bar topology via rules

INPUT		DESCRIPTION
i	**index**	***j*** index of the force(s) selection rule *(int)*
	flip	boolean operator that flips the forces order in binomials *(bool)*
ii	**S**	seed number for randomness *(int)*
iii	**type**	type of interim forces *(int / 0: monomial, 1: binomial, 2: trinomial)*
	config	topological configuration *(int / 0: in-between, 1: peripheral, 2: central)*
	P	manually defined point of reference *(Point3d)*

OUTPUT	DESCRIPTION
forceS	force(s) selection rule, expressed as an object *(ForceSelectionObj)*

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 forces* component [Fig. 35i]. After, he/she provides them as input to the *Custom set of interim forces* component [Fig. 35ii], along with the desired entropy rate, the desired topological configuration and the model at the state where the interim forces were selected from.

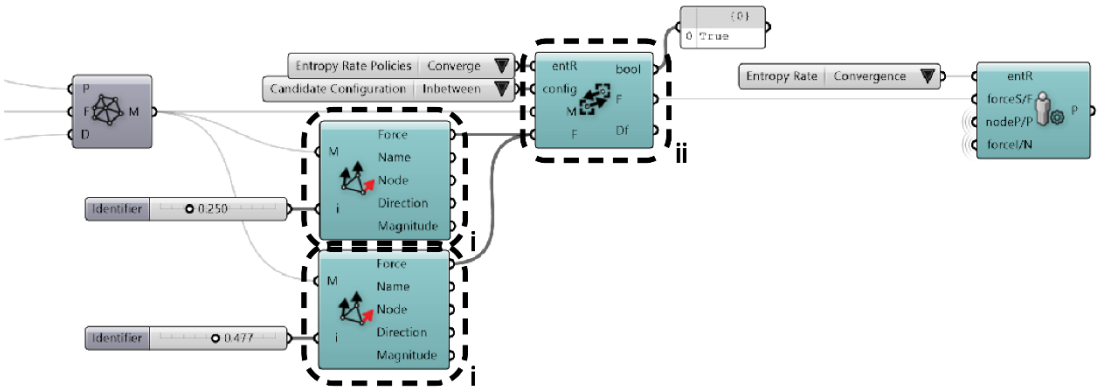


Fig. 35: Select force(s) and bar topology explicitly

The *Select forces* 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 *(Model)*
i	browse indicator between 0 and 1 *(double)*

OUTPUT	DESCRIPTION
Force	retrieved force *(Force)*
Name	force name *(string)*
Node	anchor point / node *(Node)*
Direction	direction vector of the force *(Vector3d)*
Magnitude	force magnitude *(double / "+" if force is pushing, "-" otherwise)*

The *Custom set of interim forces* component constructs a selection of forces and checks its feasibility to be used in order to achieve the desired entropy rate in the model in compliance with all the provided input parameters. The designer is responsible to confirm that the feasibility is approved, as for explicit force(s) selection, if feasibility is not ensured no mechanism seeks after an alternative set of forces. The feasibility check implies the construction of the feasibility domain. If true, the feasibility domain is returned along, and the selection of forces can be fed directly to the *Build policy* component [section 3.3.7].

INPUT	DESCRIPTION
entR	desired entropy rate *(int / -1: convergence, 0: stagnation, 1: divergence)*
config	topological configuration *(int / 0: in-between, 1: peripheral, 2: central)*
M	interim model where the interim forces are selected from *(Model)*
F	collection of external force vectors *(Force)*

OUTPUT	DESCRIPTION
bool	boolean operator indicating the feasibility of achieving the desired entropy rate for the set of choices made *(bool)*
F	selection of forces, expressed as an object *(ForceSet)*
$\boldsymbol{D_f}$	feasibility domain (*Domain*)

3.3.5 Place new node

According to section 3.1.4.3, the placement of a new node is possible either explicitly or by means of rules. Fig. 36 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 A.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. 36i], demonstrates the standard configuration of the *Build 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. Part of the standard configuration of the component is the (optional) construction of the auxiliary domain [section 3.1.5.4]. When bar length

bounds are desired two additional slots to define the lower and upper bounds are added, which is the case in second example [Fig. 36ii]. In the third example [Fig. 36iii] the new node placement rule constrains one of the node coordinates within lower and upper bounds per axis. The bounds are not defined explicitly (numerically) but by the index of nodes included already in the bars network.

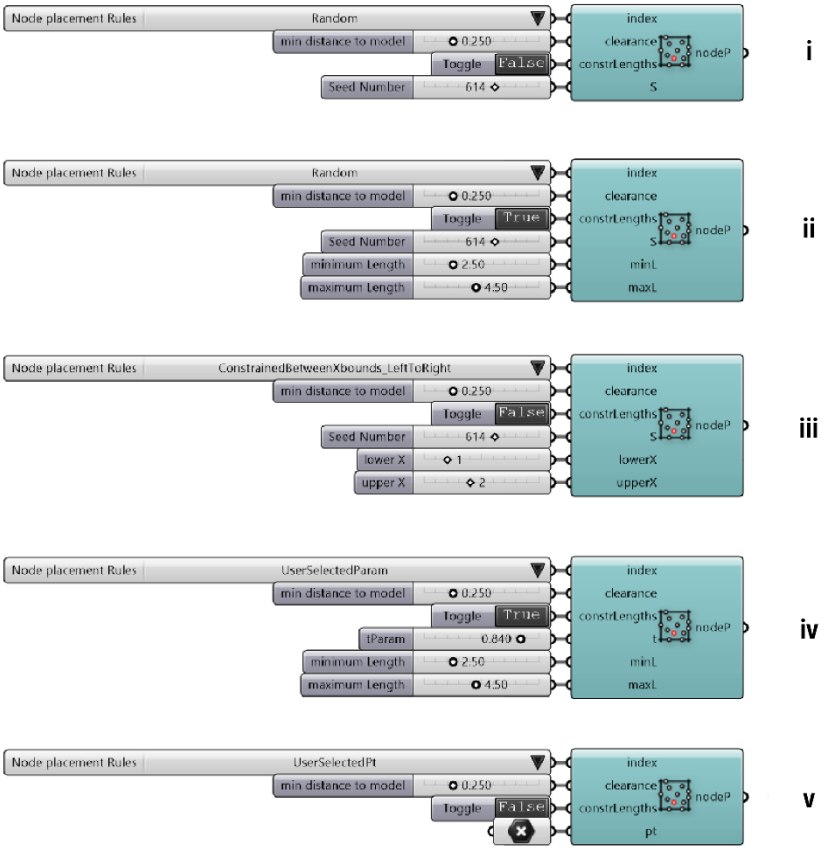


Fig. 36: Place new node via rules (i-iii) and explicitly (iv-v)

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. 36iv demonstrates the former option whilst bar length bounds are considered. The latter is demonstrated in Fig. 36v. Imposing bar length bounds in this case is not possible.

INPUT		DESCRIPTION
i	**index**	*k* index of the node placement rule *(int)*
	clearance	minimum distance to network *(double)*
	constrLengths	boolean operator to consider the auxiliary domain *(bool)*
	S	seed number for randomness *(int)*
ii	**minL**	minimum bar length *(double)*
	maxL	maximum bar length *(double)*
iii	**lower*X***	index of a node in the network; its coordinates define the lower bound in the axis indicated by the rule *(int)*
	upper*X*	index of another node in the network; its coordinates define the lower bound in the axis indicated by the rule *(int)*
iv	**t**	homogeneous coordinate that describes the new node location *(double)*
v	**pt**	cartesian coordinates that describe the new node location *(Point3d)*

OUTPUT	DESCRIPTION
nodeP	node placement rule, expressed as an object *(NodePlacementObj)*

Alternatively, the location of the new node can be fed directly [Fig. 37] to the *Build policy* component [section 3.3.7].

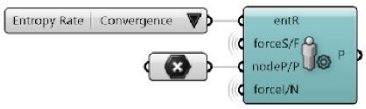


Fig. 37: Place new node explicitly and directly

3.3.6 Set force indeterminacies

According to section 3.1.4.4, setting the force indeterminacies is possible either explicitly or by means of rules. Fig. 38 demonstrates three 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 A.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 three rules exist. Their construction follows. The first example [Fig. 38i], 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 minimum and maximum stresses are provided. The second example [Fig. 38ii], 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. The third example [Fig. 38iii], demonstrates how the *User defined utilization* force indeterminacies rule is defined. The material yield strength and the cross section determine the maximum developed stress that the specific bar elements withstand. The utilization bounds constrain the generation of three random stresses. A seed number controls the generation randomness.

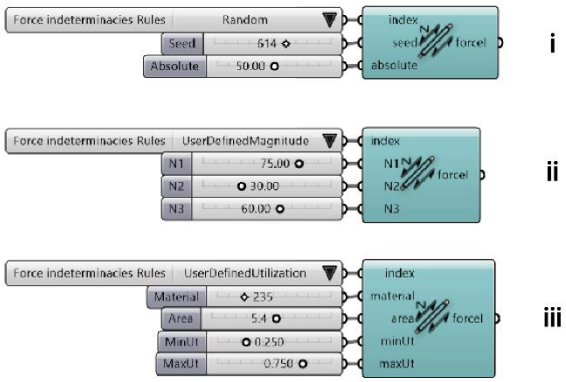


Fig. 38: Set force indeterminacies via rules

INPUT		DESCRIPTION
i	**index**	***m*** index of the force indeterminacies rule *(int)*
	S	seed number for randomness *(int)*
	absolute	absolute magnitude of developed stress *(double)*
ii	**N1**	stress along bar #1 *(double)*
	N2	stress along bar #2 *(double)*
	N3	stress along bar #3 *(double)*
iii	**material**	material yield strength *(double)*
	area	cross section area *(double)*

S	seed number for randomness *(int)*
minUt	minimum material utilization in percentage [0, maxUt) *(double)*
maxUt	maximum material utilization (minUt,100%] *(double)*

OUTPUT	DESCRIPTION
forceI	force indeterminacies rule, expressed as an object *(ForceIndeterObj)*

To explicitly set the force indeterminacies, the designer can be feed directly the *Build policy* component [section 3.3.7] with a list of custom force magnitudes [Fig. 39].

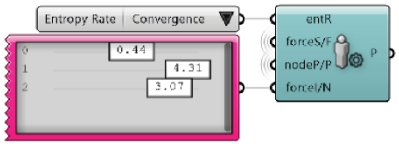


Fig. 39: Set force indeterminacies explicitly

3.3.7 Build 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 *Build policy* component. Similarly to the entropy rate, when the first three aspects are defined explicitly, they can be provided directly to the *Build policy* component [Fig. 35, Fig. 37 and Fig. 39]. Otherwise, the provision of the respective rules is expected [Fig. 31].

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

OUTPUT	DESCRIPTION
P	transformation policy *(Policy)*

3.3.8 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.

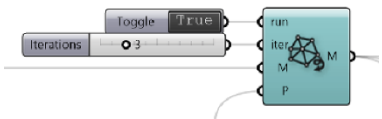


Fig. 40: Apply transformation Grasshopper component

INPUT	DESCRIPTION
run	boolean operator that launches the network transformation(s) *(bool)*
iter	number of transformation steps *(int)*

M	interim model to transform *(Model))*
P	transformation policy *(Policy)*

OUTPUT	DESCRIPTION
M	transformed model *(Model)*

3.3.9 Metrics and history

Evaluation and comparison between interim or complete networks is possible through supplementary Grasshopper components. Their output includes the calculation of metrics, the construction of a spider graph or the plotting of the applied parameters that defined the transformation policy for each transformation step.

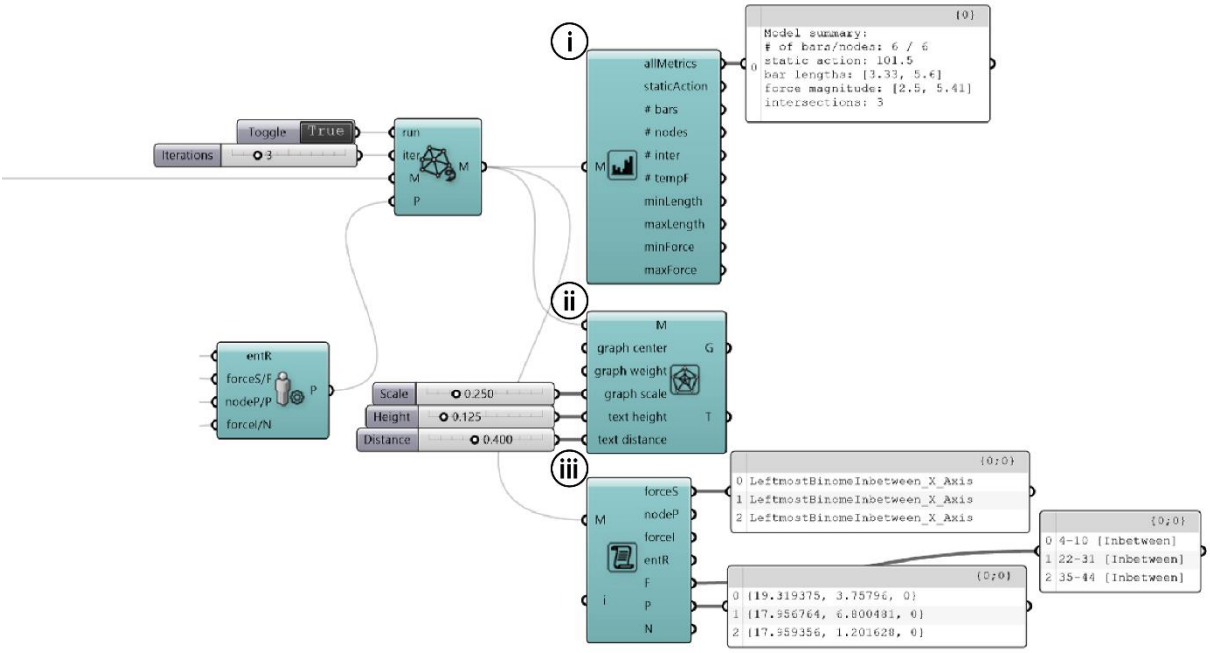


Fig. 41: Model related metrics and history

Fig. 41i demonstrates the *Model metrics* component that 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 objective like described in Fig. 55 and section 4.5.1.

INPUT	DESCRIPTION
M	transformed model *(Model)*

OUTPUT	DESCRIPTION
allMetrics	description of all model metrics *(string)*
staticAction	model static action *(double)*
# bars	number of network bars *(int)*
# nodes	number of network nodes *(int)*
# inter	number of intersecting bars *(int)*
# tempF	number of interim forces present in the network *(int)*
minLength	shortest bar length in the network*(double)*
maxLength	longest bar length in the network *(double)*
minForce	lowest developed stress in the network *(double)*
maxForce	highest developed stress in the network *(double)*

Fig. 41ii demonstrates the *Model spider graph* component that plots a spider graph with eight of the most important metrics of the bar networks. Hence, the comparison between networks is easy [Fig. 50, Fig. 57, Fig. 59]. The input parameters are responsible for configuring the visual representation of the spider graph.

INPUT	DESCRIPTION
M	transformed model *(Model)*
graph center	spider graph center *(Point3d)*
graph weight	weight for each of the 8 metric values *(GenePool)*
graph scale	spider graph scale *(double)*
text height	legend text height *(double)*
text distance	legend-graph distance *(double)*

OUTPUT	DESCRIPTION
G	spider graph *(Curve)*
T	spider graph legend *(Text)*

Fig. 41iii demonstrates the *Model history* component that returns all the actual parameters used for the model transformation. The returned 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 with the help of the *Model history* component.

INPUT	DESCRIPTION
M	transformed model *(Model)*
i	transformation step whose information is requested *(int)*

OUTPUT	DESCRIPTION
forceS	force(s) selection rule at ***i*** transformation step *(ForceSelectionObj)*
nodeP	node placement rule at ***i*** transformation step *(NodePlacementObj)*
forceI	force indeterminacies rule at ***i*** transformation step *(ForceIndeterObj)*
entR	applied entropy rate at ***i*** transformation step *(string)*
F	set of interim forces selected at ***i*** transformation step *(ForceSet)*
P	actual location of *P* node at ***i*** transformation step *(Point3d)*
R	force magnitudes at ***i*** transformation step *(double)*

The next section presents various application studies made with the use of *Libra*.

3.4 Application studies

This section presents various application studies, all generated with the use of *Libra* at its earliest stage [Fig. 42 - Fig. 44] all along the current one [Fig. 45 - Fig. 49]. The detailed settings for the network generation in every study are provided in Appendix C. For those studies demonstrating the incremental generation of a network, the last transformation consists of introducing a bar element (in compression or tension) without the introduction of a new node *P*.

Fig. 42 and Fig. 43 present resembling case studies with equally distributed load forces applied on a complex, non-convex design domain. The support reactions have different angles in these two studies. The orientation of the lines of action of the interim forces at the supports and their relationship with the design domain border imposes different behaviors and results in different network topologies. Although convergence is eventually ensured for both studies, its rapidity is slowed down by the non-convex domain and by the random selection of most input parameters. Randomness and lack of real control over the process are reflected on the large valence of many nodes as well as the large number of bar intersections. These are the first studies that nicely prove the concept of PEER framework and have been the early motivation that urged the framework development forward.

Fig. 44 is another variation of an arch-line design with a symmetric design domain and loading. The choice of a curved design domain aims at the generation of a smooth-arched network. Comparing the network with those shown at Fig. 42 and Fig. 43 it is confirmed that the design domain geometric features are impactful on the generative process as emphasized at Fig. 29. At the same time, the curved design domain exemplifies that non-segmented regions are compatible with the definition of a design domain. The network topology signifies added control over the topology and the planarity of the final design, due to carefully chosen force(s) selection rule(s).

The next application study is a symmetric electric pylon, Fig. 45. The design domain presents cavities and acute angles that make it non-convex. Applied policies include variant force selection rules. The node placement is always random, with varying random seeds. For the symmetric result, symmetric choices were made through the force selection rule and identical entropy rate was intended. Therefore, through the random node placement rule and the identical randomness seed, the actual location of P was always symmetric. Local narrowness of the design domain greatly constrains the geometry and topology of the flow of forces and consequently the introduction of the necessary bars. However, as the rule application acts blindly the process continues unaffected until completion.

Fig. 46 is a three-dimensional case study to prove that PEER is not only limited to the generation of planar networks. The choice of rules is made on a step-by-step basis. The initial design objective is to bring the interim forces from the supports up to the deck level, using force selection rules that consider trinomials. After, all interim forces are co-planar and the transition to a connected network is straight forward, using force selection rules for binomials. Binomial transformations constrain the exploration strictly on their plane of action.

Fig. 47 displays three cantilevering trusses, all resulting from the same problem consisting in one force applied parallel to two given reaction supports. The choice of rules is made on a step-by-step basis and is different in every step to achieve diversification. The design domain is convex and eases the fast convergence. Restrictions on minimum and maximum bar lengths are applied to the auxiliary domain. The number of transformations is highly dependent of the desired bar length bounds. Fig. 48 illustrates the incremental transformation and highlights the shrunk feasibility domain as a result of the auxiliary domain. The tree cantilevers are topological and geometric variations of the same family of structures. Key metrics are provided for each design outputs and allow their quantitative comparison. Static action [92] 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.

Fig. 49 is another example that underlines the impact of the choice of force(s) selection rules to the result. For the study, alternative flows of forces for tree-like structures supported at a single point, are generated. The careful selection of forces allows the generation of networks with less intersecting bars. For the last three design candidates [Fig. 49ii - Fig. 49iv], the bar length constraints had to be deactivated

when applying the last transformation to reach the state of global equilibrium without additional divergence - i.e., without the introduction of a greater number of bars. The use of different force selection rules policies has allowed distinctive variance and topology among the proposed design candidates [91].

For every design problem, the number of design candidates is infinite. The same design problem was given to multiple users [Fig. 50]. The non-convex design domain adds complexity to the generative process and many designer intentions, at transformation level, cannot be achieved. Consequently, selections of different input parameters result in completely different designs. Some networks result from a step-by-step control by the user. Others – i.e., cases ii, iii, v, ix, x, xiv, and xvii - iterate the same rules for multiple steps. The fact that cases x, xiv and xvii each result from the application of a single predefined set of rules throughout all generation steps, but do not showcase repeated or regular bar layouts, emphasizes how controlling high-level policies is useful to generate bar networks in static equilibrium that are both geometrically and topologically diverse. The input parameters and their values for each design candidate are provided in Appendix C.

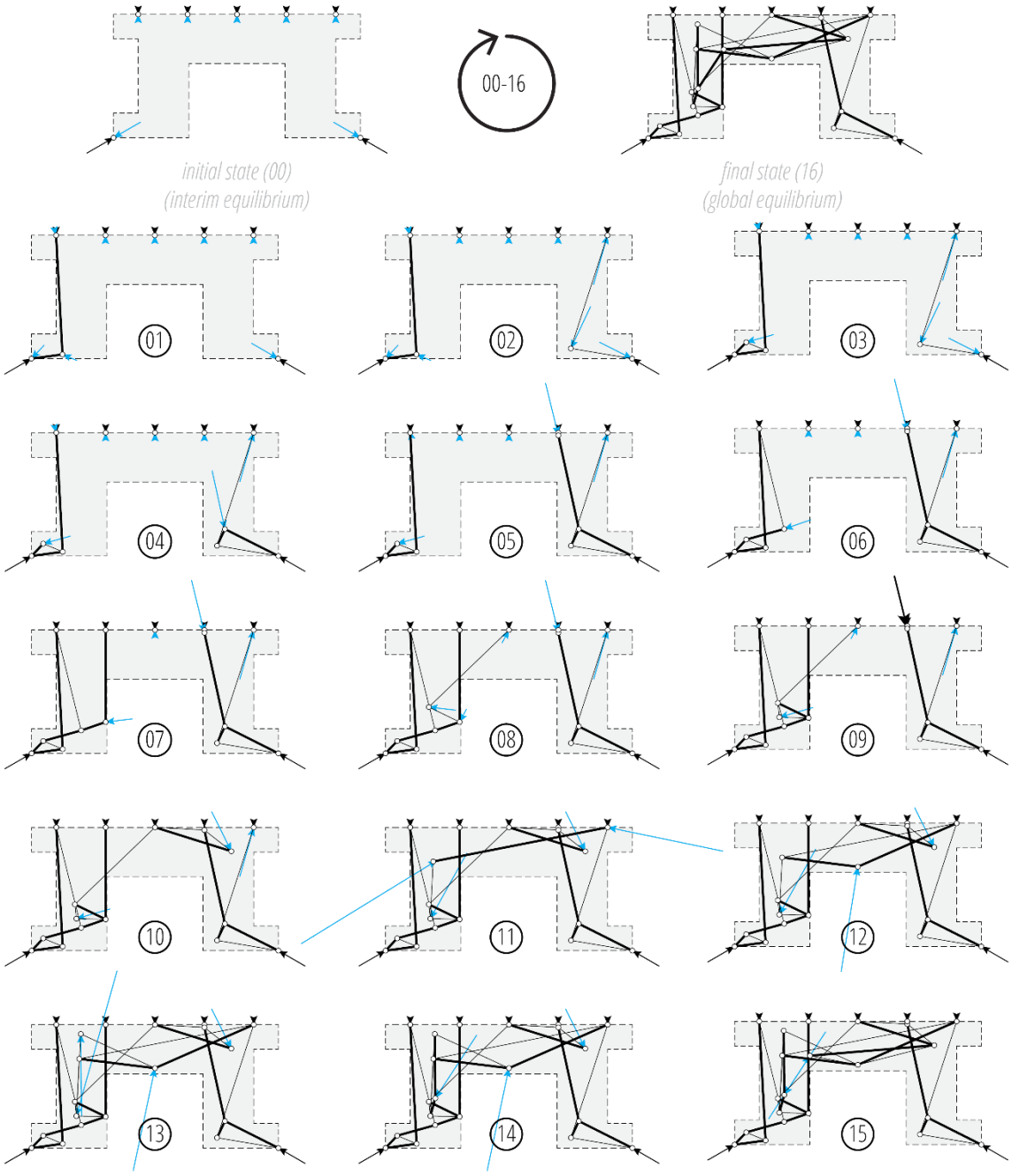


Fig. 42: Incremental policy transformations for the generation of an arch-like network of bars in compression and tension within a non-convex orthogonal domain

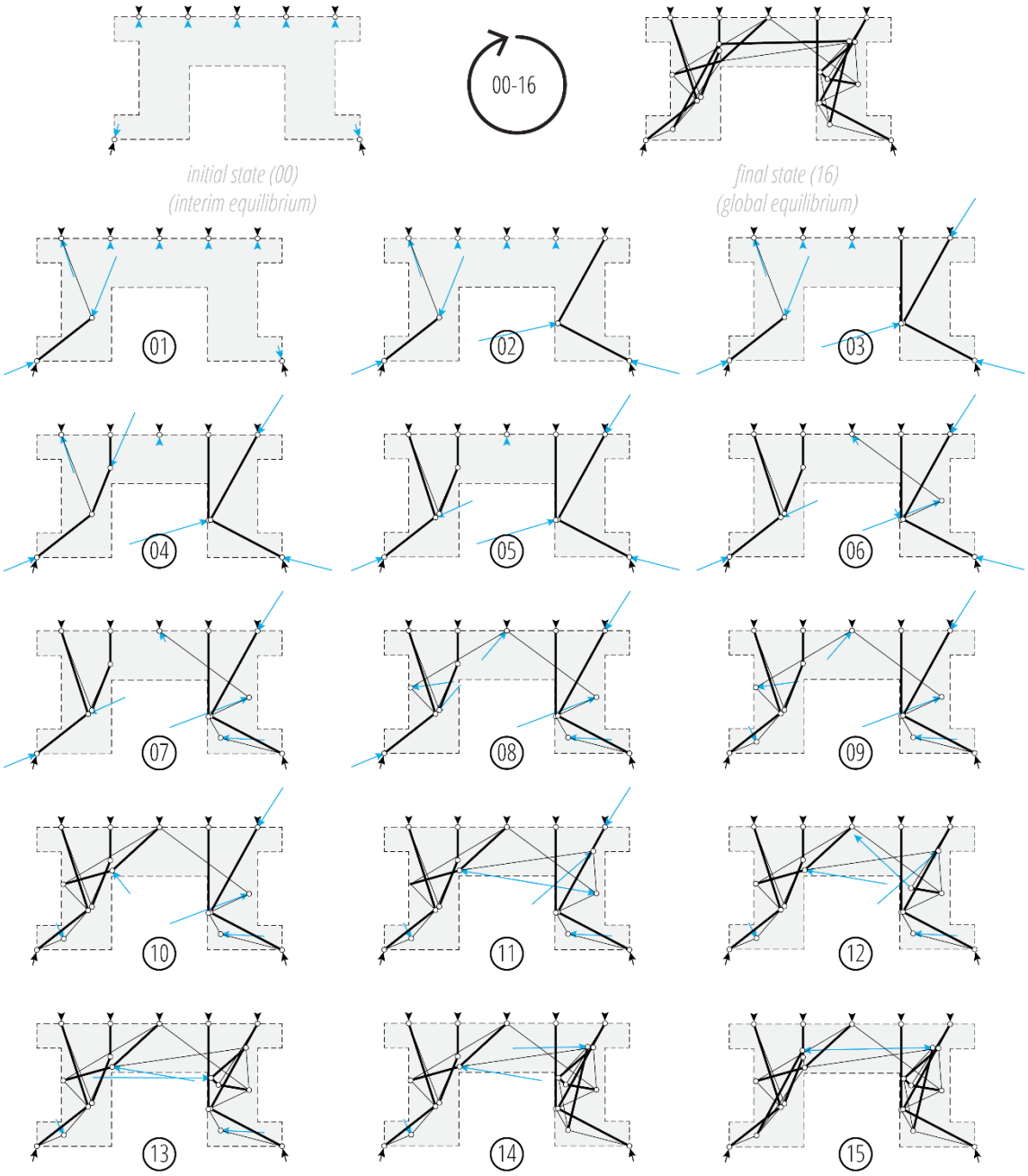


Fig. 43: Incremental policy transformations for the generation of an arch-like network of bars in compression and tension within a non-convex orthogonal domain

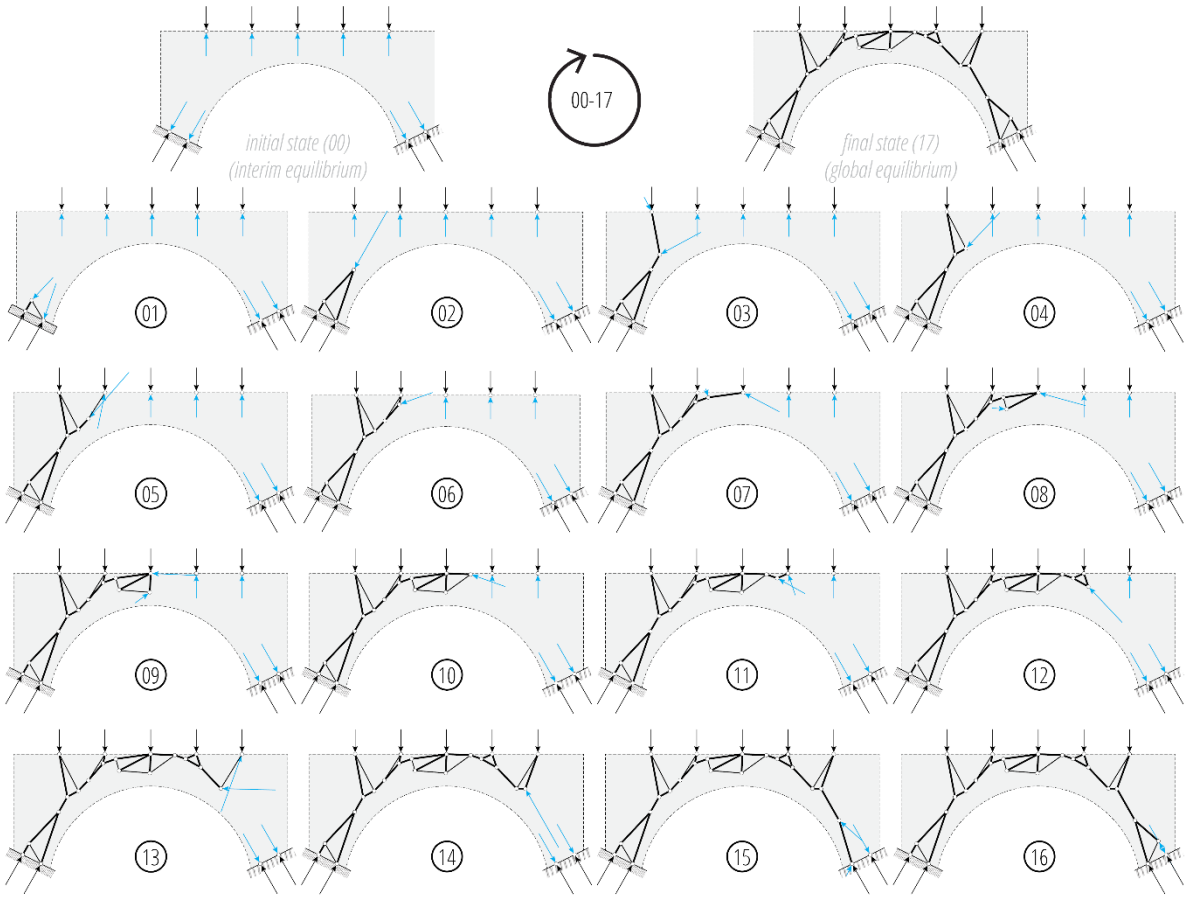


Fig. 44: Incremental policy transformations for the generation of an arch-like network of bars in compression and tension within a non-convex curve-lined domain.

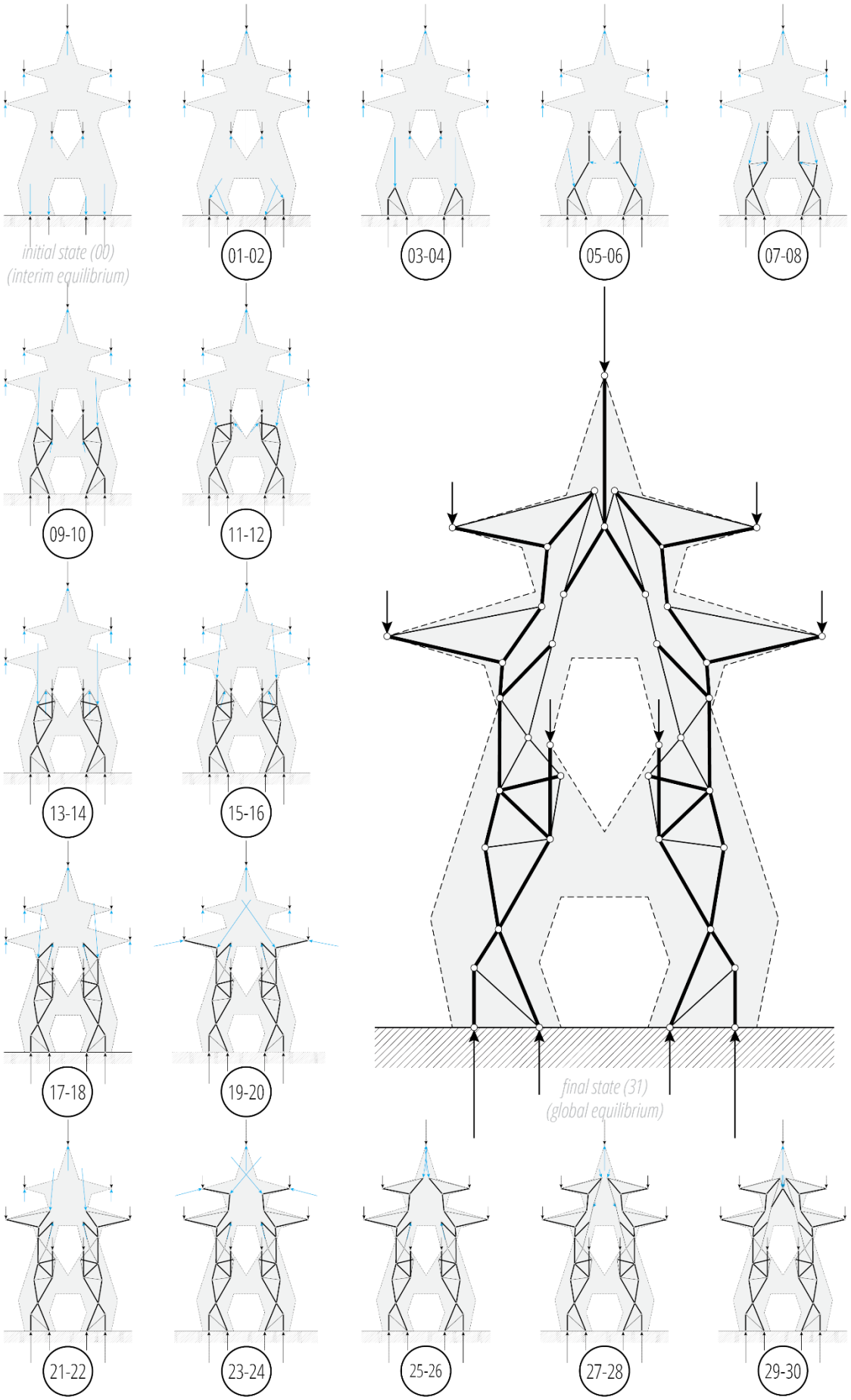


Fig. 45: Incremental policy transformations for the generation of a symmetric pylon-like network of bars in compression and tension within a non-convex domain

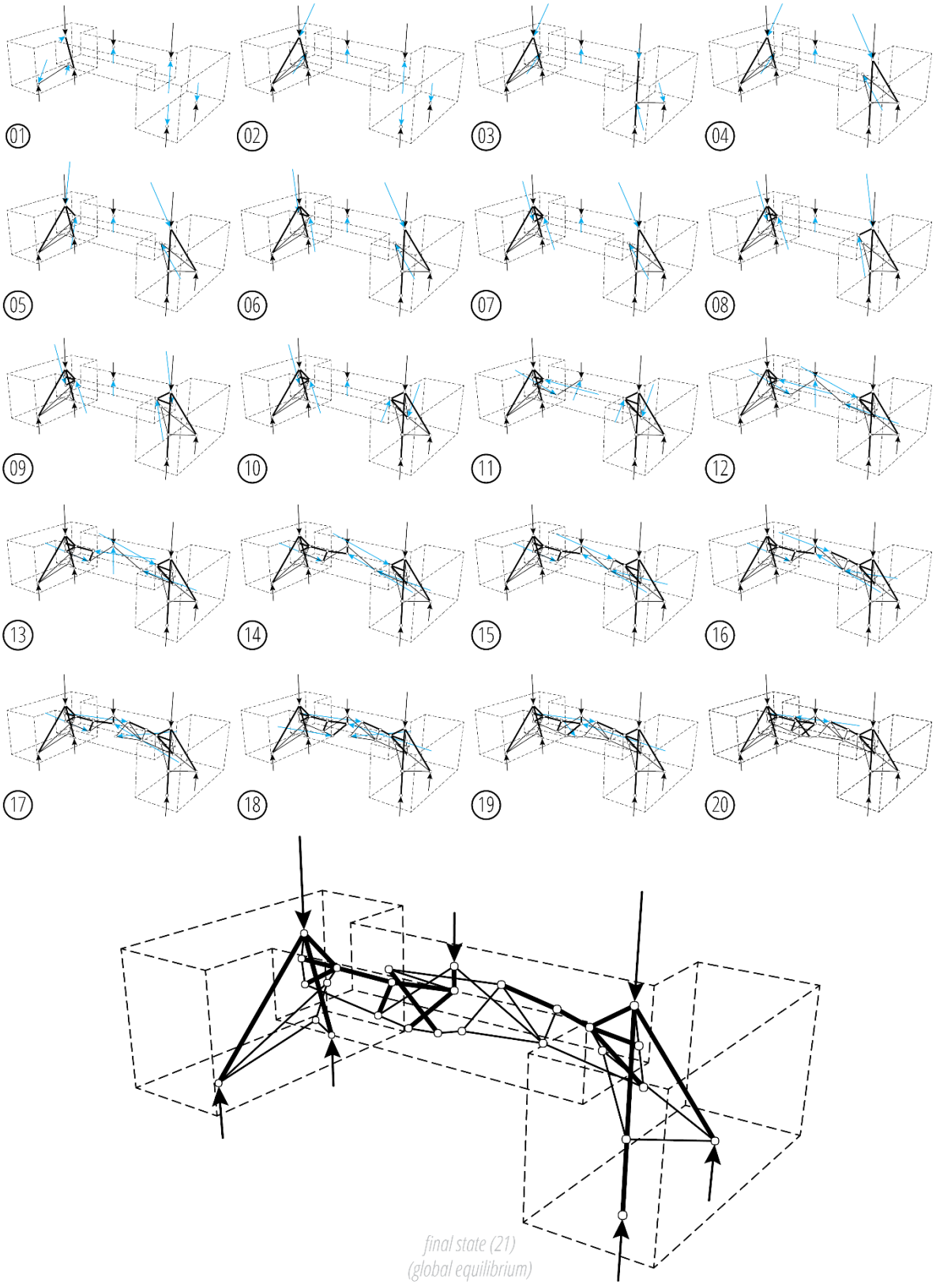


Fig. 46: Spatial bridge-like network of bars in compression and tension [93]

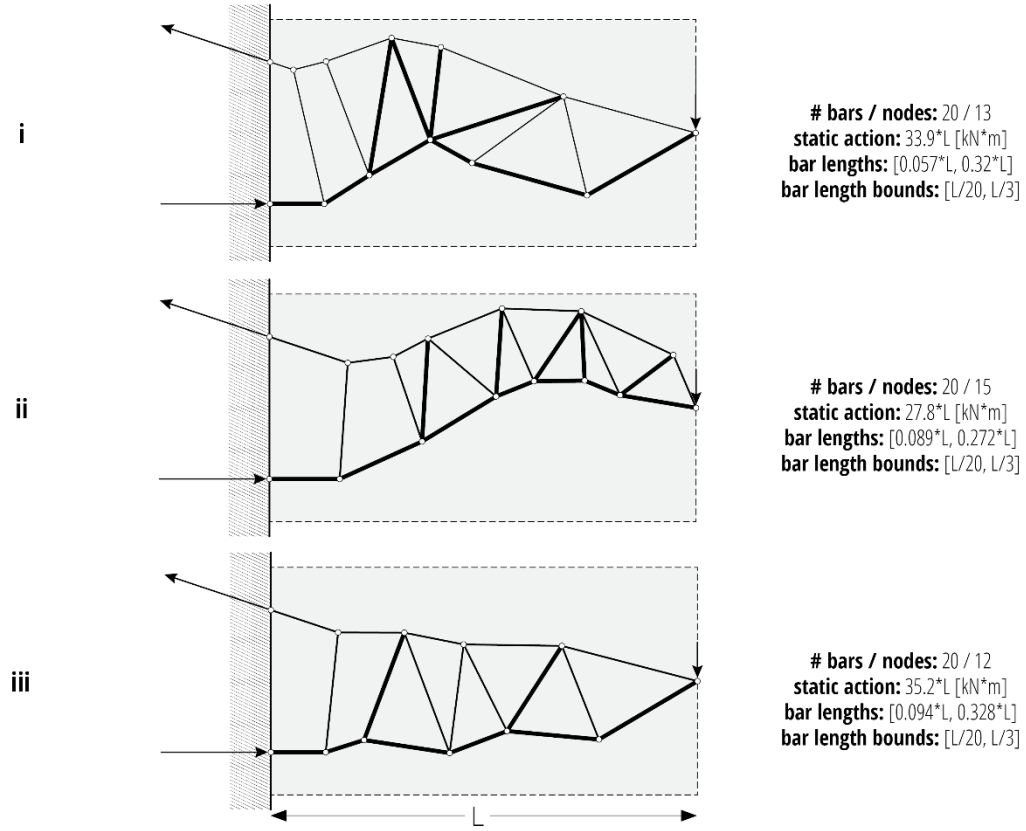


Fig. 47: Diverse cantilevering networks of bars in compression and tension as a result of different node placement rule

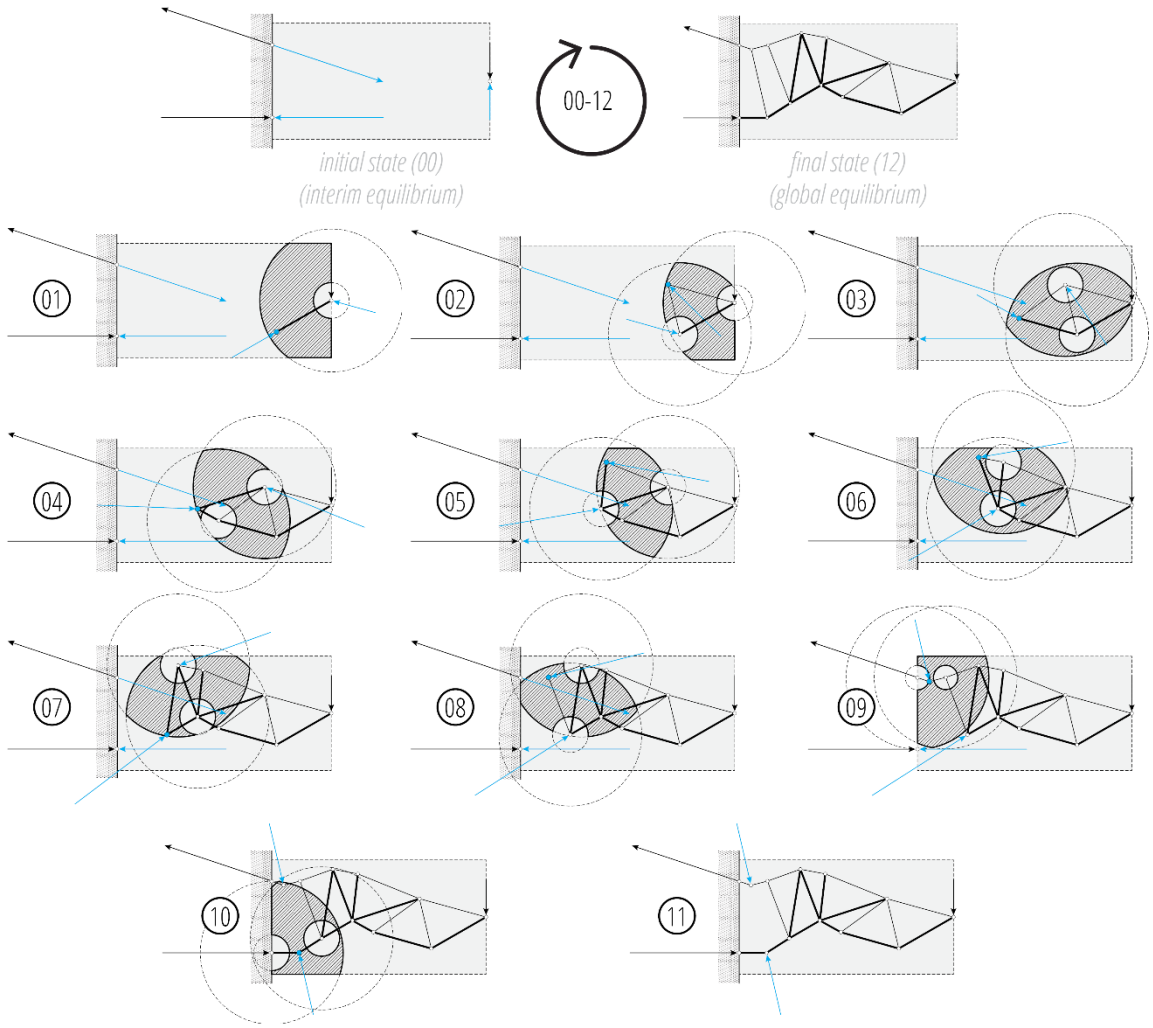


Fig. 48: Incremental policy transformations for the generation of a cantilevering network of bars in compression and tension [Fig. 47i]. The hatched regions represent the auxiliary domains, as defined by the bars' minimum and maximum lengths (represented with dashed circles). The newly introduced nodes are highlighted in cyan.

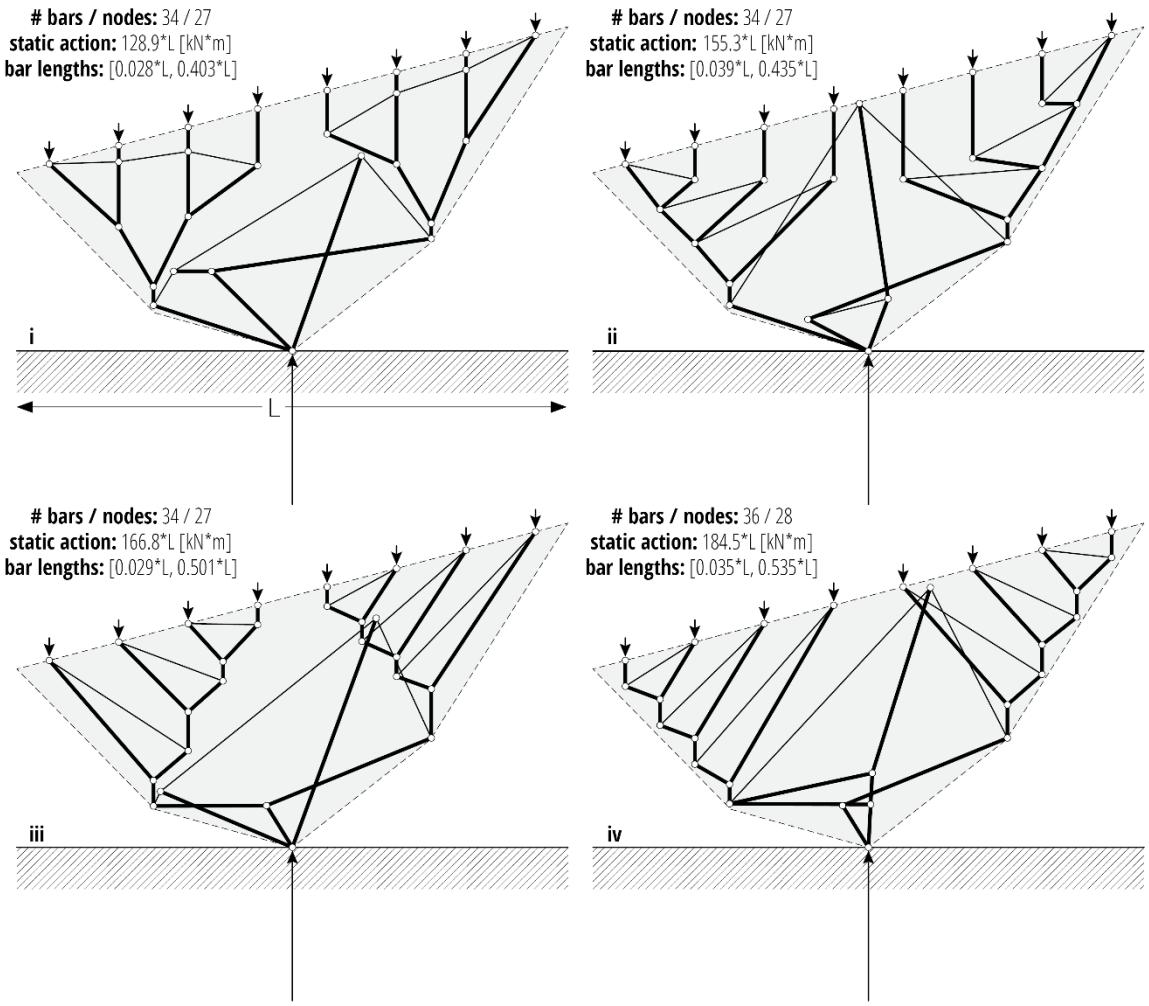


Fig. 49: Diverse tree-like networks of bars in compression and tension as a result of different force(s) selection rules

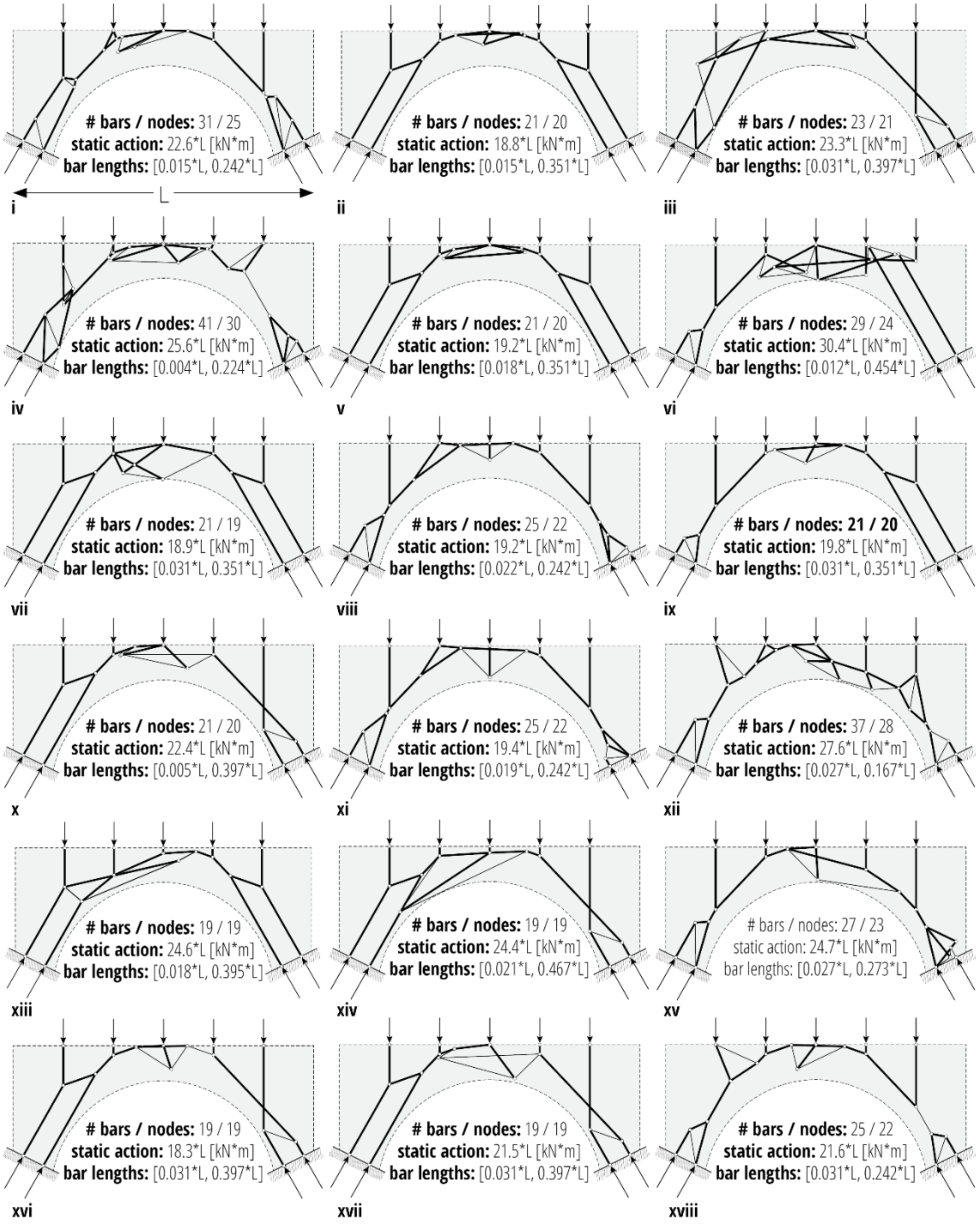


Fig. 50: Collection of diverse arch bridge-like networks of bars as a result of different input parameter

4 Interactive design space exploration

The wicked nature, the emerging requirements and constraints, the contradictory solution pathways, the absence of quantitative criteria and the lack of global optimality in design, are thoroughly described in 1.2.2. Regardless of the design field, be it architectural or structural, the main mechanisms and specific methods that people have come up with to tame the abovementioned particularities, are explained in 1.2.3. The notion of interactive evolutionary design is introduced in 1.2.3.4.

The current part explains the practice of automated design space exploration along with the PEER framework, namely the generation of multiple bar networks at once through evolutionary computing. First, it presents the existing framework that has been considered for the exploration process. It continues with its integration within the PEER framework and discusses the added value it brings. The last section concludes with two application studies that showcase the potential of Policy-design Exploration of Equilibrium Representations.

4.1 Concept

Part 2 thoroughly explained the methodology of incremental–node-by-node–growth of a network of bars. The 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. Genetic algorithms (GA) have been integrated in the PEER framework to produce diverse, but still structurally relevant, networks more efficiently–i.e., lower static action, less intersecting bars etc.–than before, when brute-force and random values were applied to generate networks with PEER frame-work alone. The stochastic nature of metaheuristic algorithms augments the breadth of design candidates but also returns optimized networks for various metrics.

As described in 3.3 the implementation of PEER has been hosted by *Libra*, a plug-in developed by the author that operates in the parametric environment of Rhinoceros Grasshopper. Evolutionary computing is already available in Grasshopper. Fusing evolutionary computing with PEER in the same platform is sensible, as exploration and post-processing of the most favorable design candidates perform in the same environment. To achieve this goal, the selection of the most suitable existing plug-in, to couple PEER with, must satisfy the following features:

- Operate on bi-directional metaheuristic solvers;
- Consider single and multi-objective optimization;
- Maintain easily accessible search history;
- Provide interaction with the user;

4.2 Framework selection

After evaluating the various evolutionary solvers that have been implemented for the environment of Grasshopper, the following conclusions are made. The out-of-the-box component of Galapagos is limited to single objective optimization. Multi-objective evolutionary algorithms (particularly the SPEA-2 and HypE) are accessible through *Octopus* plug-in [41], but it does not keep record of the search history. Other plug-ins that operate on stochastic evolutionary algorithms include *Optimus* [94], *Wallacei* [95], *DesignSpaceExploration* [96–98], *Goat*[3] and *Biomorpher* [3], each coming with its own assets and flaws. Most of these tools emphasize performance quantitative criteria, which is different to interactive evolution, where no quantitative performance criteria are required [3].

Among the prospective candidates, only *Wallacei* and *Biomorpher* meet all four abovementioned strict requirements. Both are well targeted to stochastic exploration and consider multi-objective optimization. The search history is safely stored for all generation of the process. Browsing through and backtracking to previous design candidates is intuitive and user friendly too. On top of that, the latter one is open access (MIT license) and its implementation is inspired by interactive evolution as presented by Dawkins [30]; exploration of the solution space without predefined/expected quantitative criteria/performance.

Getting access to the source code has been decisive for the selection of *Biomorpher*. Though it is developed before *Wallacei*, the tremendous work by John Harding, its constant maintenance and robustness after extensive testing and multiple examples from both academia and industry qualify it as the ideal framework to support an exploratory framework like PEER within *Libra*.

4.3 Methodology

The current *Libra* version integrates the latest *Biomorpher* release (0.7.2), as implemented in https://github.com/johnharding/Biomorpher. This section goes through a brief presentation of *Biomorpher* functionalities, available to any standard Grasshopper user. Their integration with *Libra* is described in 4.4.

4.3.1 Framework overview

Biomorpher is a minimal, but still functional, practical and efficient, implementation of interactive evolutionary algorithms. As an evolutionary algorithm, its implementation corresponds to the primitive workflow provided at Fig. 3. The interactivity addition updates the same workflow as such:

[3] *https://www.rechenraum.com/en/goat.html*

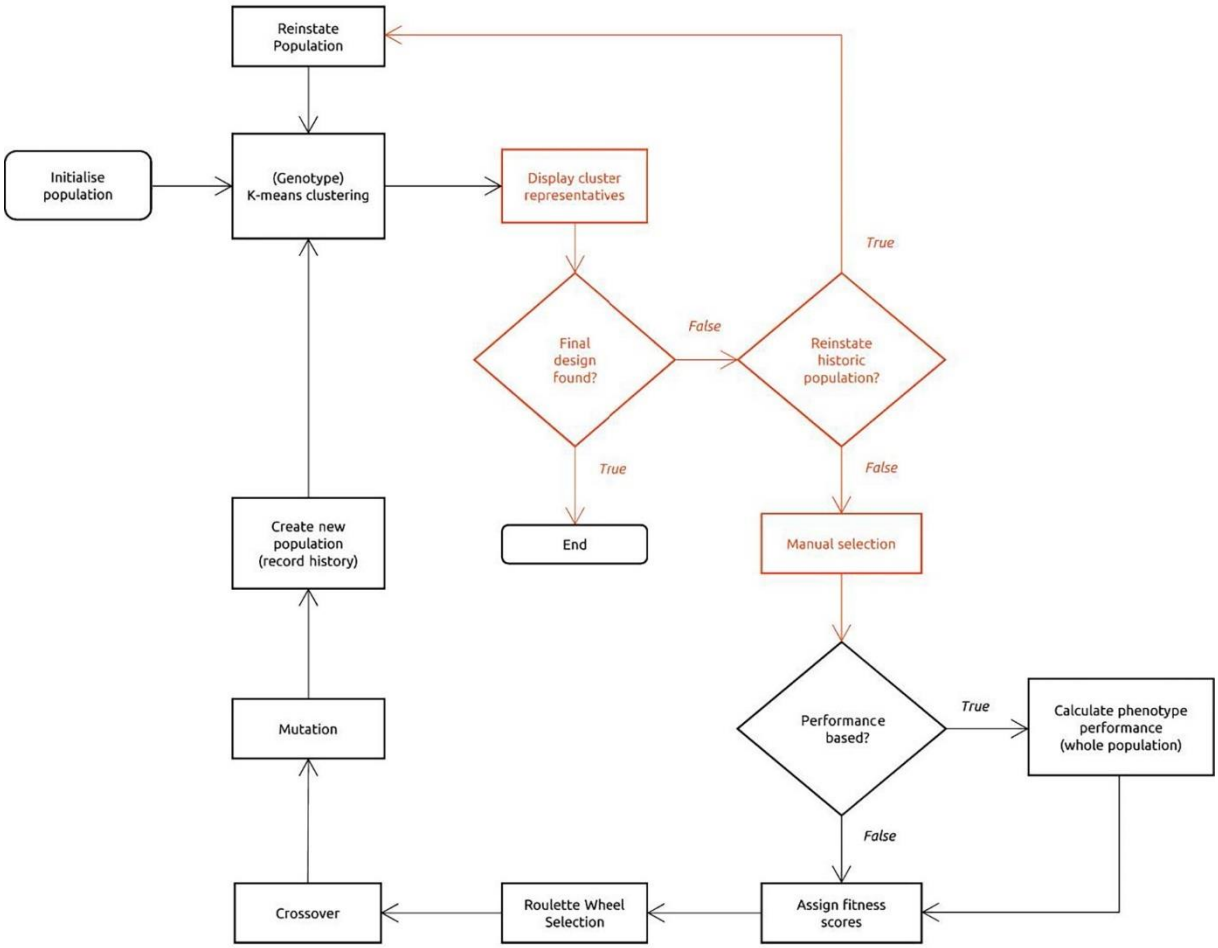


Fig. 51: Biomorpher algorithmic workflow [3]

4.3.2 Basic and optional input

The design space exploration is realized through interactive evolutionary computing after launching the *Biomorpher* Grasshopper component which hosts the entire process. The component itself requires three types of inputs: (a) the model geometry (originally *Brep*, *Mesh*, *Box*, *Surface*, *Arc*, *Line* or *Curve* objects); (b) the genomes (originally *number sliders* and/or *gene pools*); (c) the numerical performance (optionally). By double clicking on the component a separate window that supports the interactive exploration opens [Fig. 52].

Originally in evolutionary computing, the genotypes of each generation are assessed based on an objective function. In interactive evolutionary algorithms the goal is the replacement of automated evaluation process by a hybrid mode that allows either manual or automated parents selection, according to the designer's wish to intervene or not. This mode does not exclude the existence of an objective function and thus does not make interactive evolution inappropriate for quantitative optimization problems. In other words, the process still runs without any objective function, driven by the user's input instead. Considering the latter, the provision of performance metrics is optional.

4.3.3 User interface

The popping up window, which takes after the interface demonstrated by Dawkins more than 30 years ago, originally includes six tabs, each corresponding to different stage and functionalities of the process:

- **Population**: The place where most of the functional settings are made prior to initiating the population evolution. The dashboard displays 12 genotype distributions. Each comprises one representative out of the 12 clusters that describe the current population. At this stage, grouping is based on genotypes. Namely, in biology terms, the set of genes an organism has. In the case of design, the genes represent specific input parameters that have been provided by the user. The physical features of an organism (i.e., morphology, behavior, physiology), named phenotype, result from the genotype.
- **Designs**: The phenotype display dashboard of the 12 representative design candidates corresponding to the clusters in the previous tab. The quantitative score of each representative with respect to the input performance is displayed on the left bottom. Each dashboard tile hosts a checkbox that allows for manual cross and mutation selection of parents. The number of generations the (upcoming) exploration lasts for is defined in this tab. The evolution start/resume is requested here too.
- **History**: The recorded search history is visualized in this tab. It offers an overview of the entire process to the designer.
- **Performance**: If performance objectives are set, their values throughout the process are plotted on an informative dotted line graph in this tab. The dot-indicated values correspond to the performance of the 12 representative design candidates as well as the population's average performance.
- **Scatter**: This tab offers another type of performance overview allowing the designer to compare two performance criteria against each other on a scatter graph.
- **About**: Code dependencies and contact information are provided into this last tab.

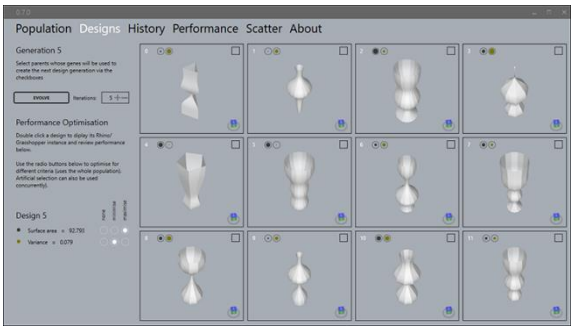


Fig. 52: Biomorpher user interface

4.3.4 Interactive decision making

As described in 1.2.3.4, three moments of interactivity are witnessed in evolutionary design. The following sections describe how two of them (interactivity at *declaration time* and during *execution time*) get integrated in the framework of *Biomorpher*. The interactive provision of input parameters reflects the designer's preferences and intentions, which are likely to change throughout the exploration process.

4.3.4.1 Functional

The functional level of interactivity includes all the permanent or transient input parameters that need to be provided prior to the execution of evolutionary algorithms and are independent of their sequential execution. In other words, their values have no influence on the algorithmic workflow and its structure.

In *Biomorpher*, the permanent input parameters are provided at the Grasshopper component [Fig. 53]. The transient inputs refer to the *population size*, *crossover* and *mutation rates*, as described in the *Population* tab and the *iterations* number described in the *Designs* tab [Fig. 52]. The values of these *evolutionary parameters* are subject to change as per the designer's intentions, when either a new generation is populated or a retrieved one is resumed and further explored. The generally recommended values are used by default, but occasionally to avoid local optimum solutions further tuning is advisable. Their significance in the exploration and exploitation notions of GA is already acknowledged and further investigation of their influence in the process is out of the scope of this work. However, for more information related to efficient exploration and exploitation with GA, the reader is invited to have a look at [99].

It is remarkable that the entity of all these values is closely related to the algorithms' efficiency and outcome and retains the algorithm's sequential execution and structure uninterrupted. As such, the inputs definition is at a functional level of interactivity. Higher level interactivity is discussed below.

4.3.4.2 Operational

The operational level of interactivity includes all the input parameters that are provided during the execution of evolutionary algorithms and are influential in the algorithmic workflow, their sequential execution and their structure.

In *Biomorpher*, the selection of "parent" phenotypes corresponds to operational interactivity. Feeding the new generation with preferred parents not only steers the exploration but also reorganizes the internal structure of the genetic algorithm. For example, lower performing genotypes are qualified by force for the next generation while higher performing ones are completely rejected as prospective parents for the upcoming generations. This high-level interactivity occurs in the *Designs* tab [Fig. 52], with the help of the checkbox found at the top right corner of the dashboard tiles.

Selecting one of the tiles, respectively updates the Rhinoceros viewport and Grasshopper canvas. Like this, the designer can plot and manually further manipulate the phenotype in the viewport. In the canvas, the designer has an overview of the specific parameters, which generate the indicated model, and the possibility to get back to *Biomorpher* window, select new parents and continue the exploration.

4.3.4.3 Backtracking

Recording, and later overviewing, the entire search history adds value to evolutionary processes that aim at exploration of alternative design candidates. *Biomorpher* in the self-explanatory tab of *History* [Fig. 52] visualizes the evolution history in the form of the clustered representatives. Once again, selecting any tile in this powerful database retrieves all the input parameters in Grasshopper canvas and recalculates the phenotype in Rhinoceros viewport. Backtracking is not limited to a certain number of steps/generations and thus it successfully implements an exploration mechanism like the one presented in Fig. 56. Further manipulation, manual or through the *Biomorpher* window, are not excluded.

4.3.5 Output

After exploring the design space though interactive computing, a detailed report of all the generated genotypes, beyond the 12 representatives of each generation, is available in the *Biomorpher* native component and it is accessible to the designer upon request. Retrieving the genotypes requires the supplementary *BiomorpherReader* Grasshopper component. By providing the generation and genotype indices the designer revives the explored phenotypes in Rhinoceros and Grasshopper. Repeating the process for the population of each generation, the complete exploration map for the specific model is built. Currently, this is a manual process. An example of such a map is illustrated in Fig. 59.

4.4 Implementation

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 [3]. More precisely, as described before, it integrates any Grasshopper native object into the workflow. The object itself is not part of the evolutionary mechanism but its type (*Line*, *Curve*, *Brep*) is necessary to visualize the phenotypes in the *Designs* and *History* tabs [Fig. 52].

Libra generates its own objects and thus fusing the two frameworks is not straightforward. The addition of *Model* objects (built in *Libra*) in the list of input geometries at the *Biomorpher* component [Fig. 53] is the main required modification. For visualization reasons within the separate window's interface, the *Model* object is deconstructed into segments (*Line* objects) which are colored according to their stress condition (red for tension members, blue for compression members). The modified *Biomorpher* component joins Libra under the name *DesignSpaceExplorer* [Fig. 54].

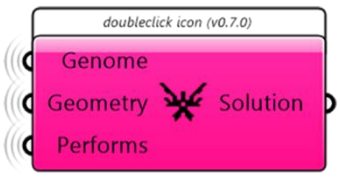


Fig. 53: Native Biomorpher component

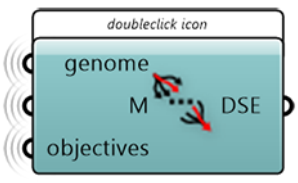


Fig. 54: DesignSpaceExplorer Libra component

Fig. 55 provides a complete overview of the evolutionary process. The designer builds 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. An applied example of this workflow is presented in section 4.5.1.

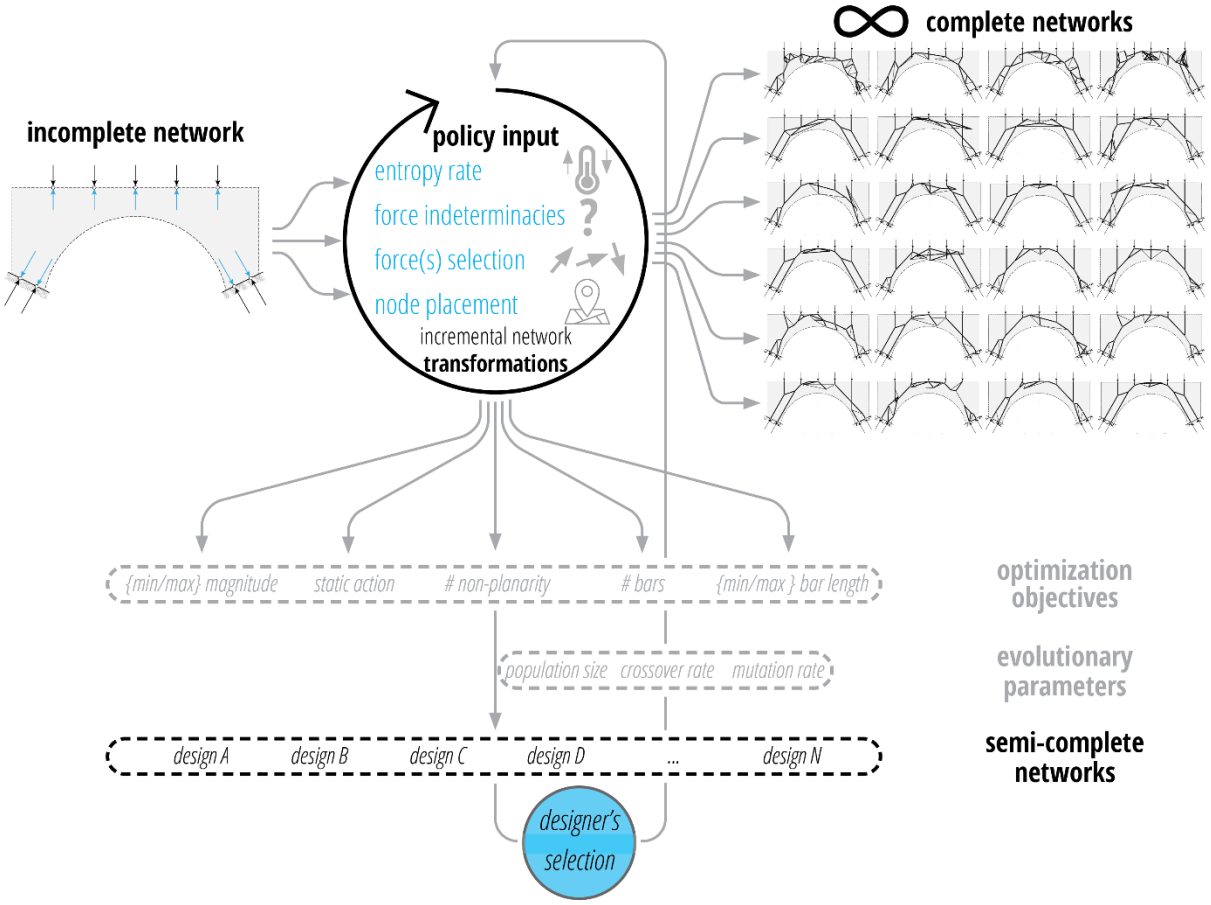


Fig. 55: Integration of PEER with interactive genetic algorithms

Biomorpher itself, enriches the arsenal of available design space explorers, upon recognizing that multiple simultaneous display is crucial for effective design space exploration [43]. Its integration into Libra plug-in augments the exploration capacity of the PEER framework. The following application studies apply the extended PEER framework for fast but thorough design space exploration.

4.5 Application studies

Design space exploration reaches a great level of efficiency through the extended version of PEER that takes advantage of interactive genetic evolution. The most significant features that facilitate design space exploration include the possibility to (a) visualize and evaluate qualitatively or quantitatively multiple phenotypes simultaneously and (b) backtrack to previous states and resume exploration towards new branches based on on-the-go evaluation. To showcase these features that highlight the significance of PEER, two application studies are illustrated.

The first study recreates the Chiasso sheds designed by Robert Maillart in 1924. During the design process, various design candidates are browsed. Backtracking practically allows undoing transformations or directly retrieving earlier stages. GA provide large populations of bar networks of diverse performance and aesthetics as the result of transformations for the same number of steps.

The second study explores alternative design candidates for the Eiffel tower. The study aims to simplify the topological complexity of the networks and reduce the static action with the help of genetic algorithms. Namely, it aims to control production costs (i.e., avoid intersecting bars that increase the number of nodes) and lower the environmental impact (i.e., balance higher material efficiency).

4.5.1 Chiasso shed

Maillart has established himself as a creative engineer who has managed to build highly efficient structures. It is extremely interesting that one of his masterpieces, the Salginatobel bridge, designed in 1928, has been constructed on a remarkably brief technical report. The final geometry is the outcome of a tailored design exploration process that considered early-stage schematic representations of static equilibrium. The structural calculations that approve the final geometry of the bridge and ensure its static equilibrium are made with the help of graphical methods [45]. Maillart mastered graphic statics. Through his work he managed to design prominent structures that attract the attention of engineers until nowadays, and he proved that static equilibrium is only a matter of geometry.

His understanding of static equilibrium through graphic statics is proven once more in the design of Chiasso shed, which represents an elegant but highly efficient and well thought structure. This subsection provides alternative design candidates through the PEER framework. The initial setting considered (i.e., the geometric boundary for the shed design, the axial distance of 260 centimeters between the vertical members) are those identified in [87]. Although the same publication provides the approximate loads that Maillart considered for the original design, the following exploration process operates on unitless applied loads, a particularity that identifies the PEER framework strengths.

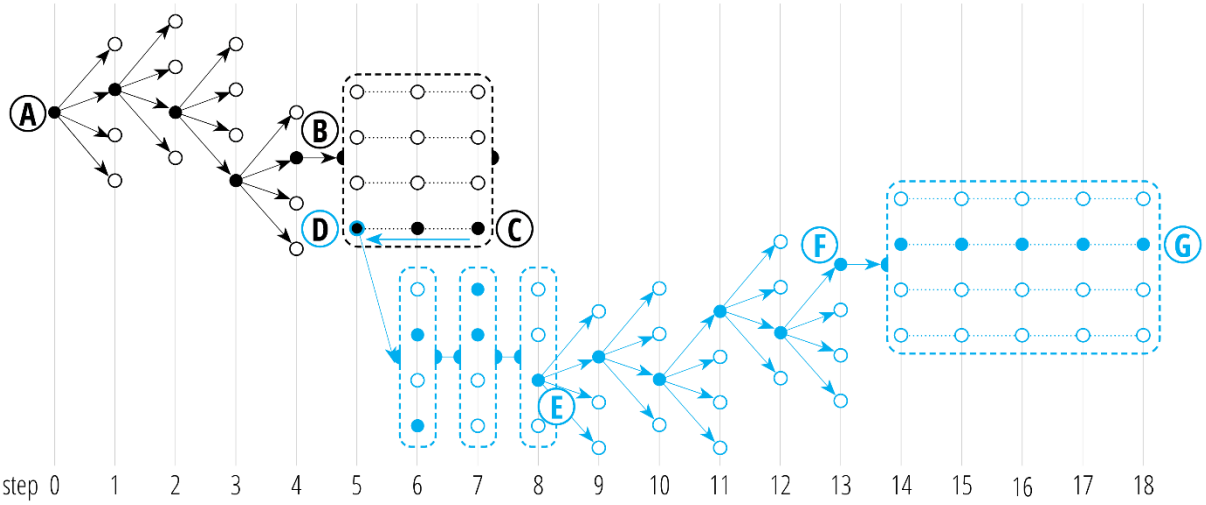


Fig. 56: Design space exploration itinerary for Chiasso shed. Cyan represents a design branch that shares part of it with the black one. Empty circles represent few of the numerous design candidates that could come out at the respective transformation step; solid circles represent the generated (or manually selected) design candidates as a result of the provided rules. The dashed rectangles indicate the use of genetic algorithms. For those rectangles that stretch for more than one transformation step (5-7 and 14-18) there is no intermediate human intervention. For the rest, manual selection of parents to crossover/mutate were selected at each step.

Fig. 56 visualizes the itinerary of the exploration from *state A* to *state G* and Fig. 57 illustrates selected intermediate design candidates for the Chiasso shed. The figures work supplementary to each other and depict the entire process of design exploration. The specific application study consists of parts of manual

and parts of automated generation of design candidates, backtracking to previous states, manual selection of parents and setup of optimization objectives later during the process. The design space exploration aims at a symmetric structure and thus considers half of the design domain.

Precisely, the exploration of diverse design candidates starts [state A] with the manipulation of the interim forces at the supports and their fast replacement with compressive bars [step 1]. This decision aims at dealing with the narrowness of the support columns since the very beginning. The next short-term goal is to replace the interim forces at the edge of the shed as early as possible to avoid the non-convexity of the geometric boundary (design domain) and continue the growth towards the middle of it. For 3 transformations the forces and the rules are selected manually at a step-by-step basis [step 2-4, state B]. The manual intervention steers the network based on human made design decision that potentially – as it is hard to evaluate - facilitate the efficient growth of the network until its final completion. At this state no geometric obstacles exist anymore, and human intelligence is neither deemed necessary nor efficient to explore the design space. Stochastic algorithms take over, the network is transformed 3 times [steps 5-7] and 12 representatives are returned. The designer reviews the outcome but none of the design candidates at that state satisfies his/her preferences. One of the close-to-favorable candidates [state C] is chosen and its undergone transformations are undone twice. The design branch of state C gets abandoned and a new state that has never been visualized is revealed [state D]. The new state becomes the starting point for a new design branch illustrated in cyan in Fig. 56 and Fig. 57.

Stochastic algorithms are exploited again to guide the upcoming generations through the manual selection of parents at the end of each transformation [steps 6-8]. At step 8, the designer selects a design candidate [state E] among the provided representatives and manually selects the forces and the rules for the next 5 steps [step 9-13]. The designer has actively participated in the exploration process and has contributed according to custom preferences to the growth of the network, in a way that is hard to evaluated regarding its efficiency [state F]. The following transformations [steps 14-16] are handled by stochastic algorithms. As the shed's final geometry is roughly imposed by the previous transformation, minimization of static equilibrium and minimization of bar intersections are set as objectives for these steps. At the end of the stochastic optimization process, the designer simply picks up the most favorable design candidate out of the list of representative candidates [state G]. The completion of the network of bars requires the introduction of a few more bars. The direction of the interim force vectors is such that convergence can be applied twice consecutively [steps 19-20]. The complete network of bars for the above design choices is illustrated at state H.

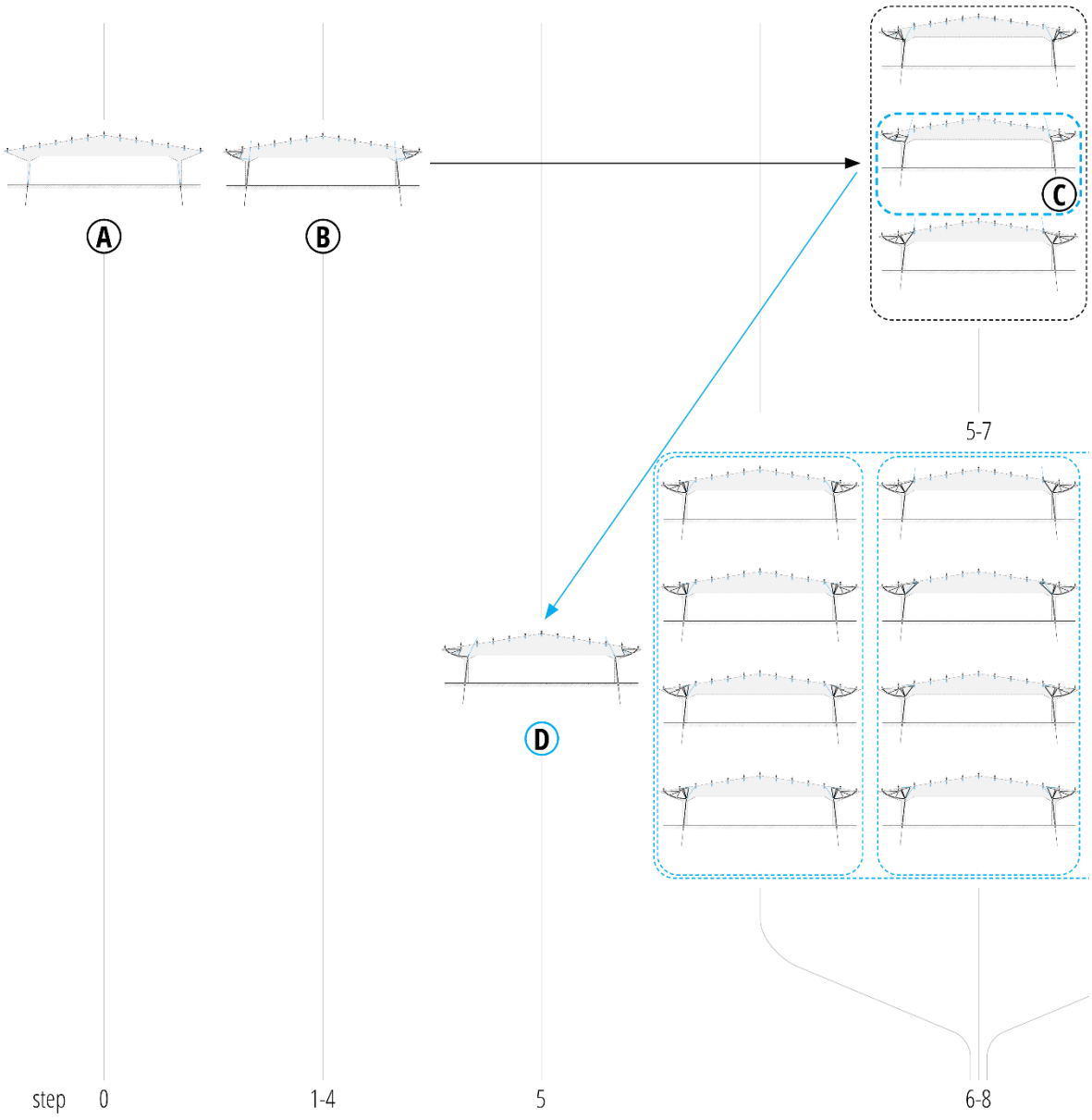
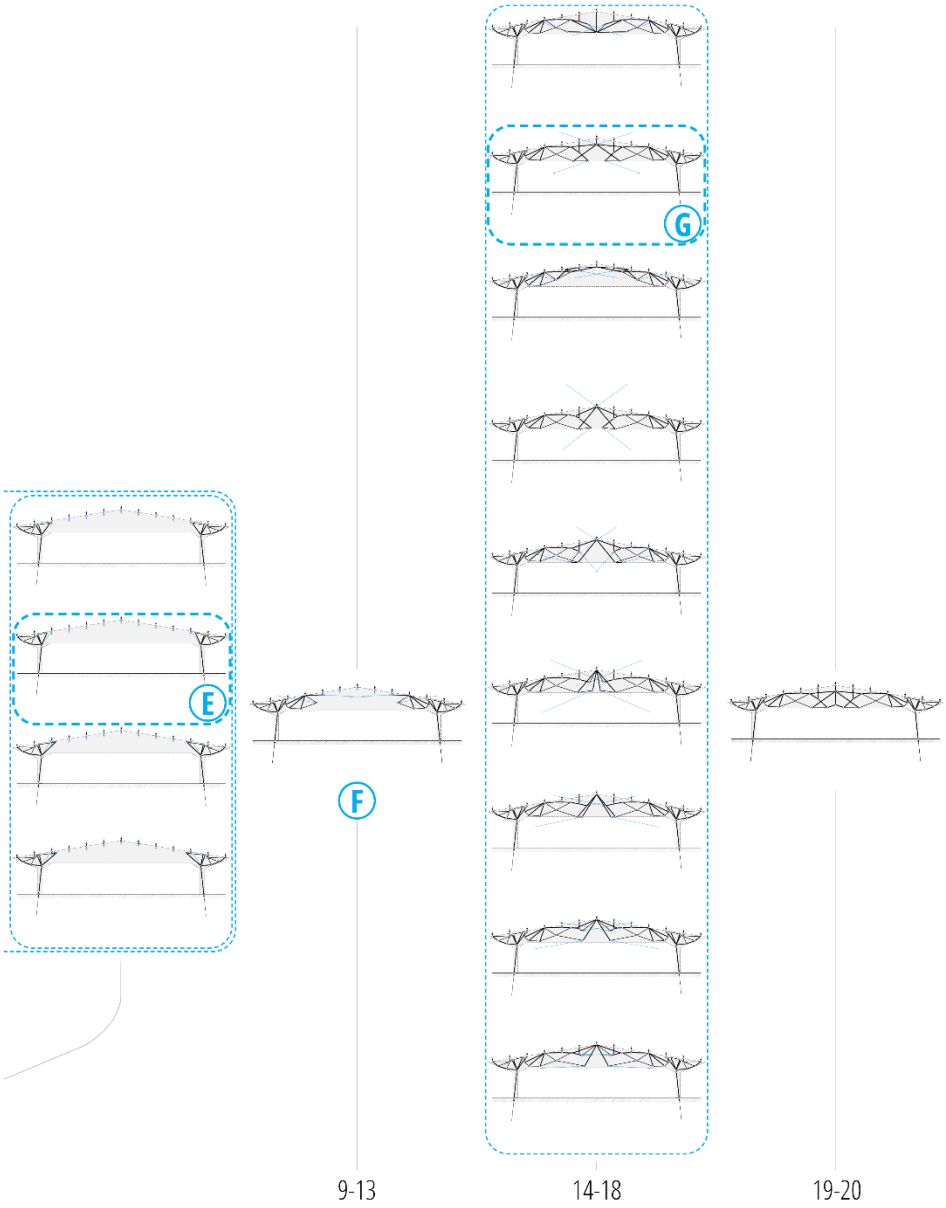


Fig. 57: Design space exploration and indicative design candidates for Chiasso shed


E
F
G
9-13
14-18
19-20

4.5.2 Eiffel tower

Eiffel tower is one of the most iconic landmarks in the globe with millions of visitors per year and its aesthetics has inspired many designers and artists. It is a megastructure selected as the winning entry in a competition for the Paris Exposition of 1889. Its elegance and efficient design are justified by using cast iron. Iron also highlights its potential in high rise construction through the Eiffel tower example too.

High rise structures are subject to resisting strong wind loading which is nor uniform as we move farther from the ground level. The wind pressure on the Eiffel tower is stronger near the top than at the bottom, but the wind force is uniform for simplicity. The assumption made during the design of the tower is that the wind is a uniform load acting all along the tower, but a conservatively high force was used to simulate high wind speeds. Based on this assumption, the curved shape (close to a parabola) was selected because it is the most efficient shape in resisting the wind load [Fig. 58].

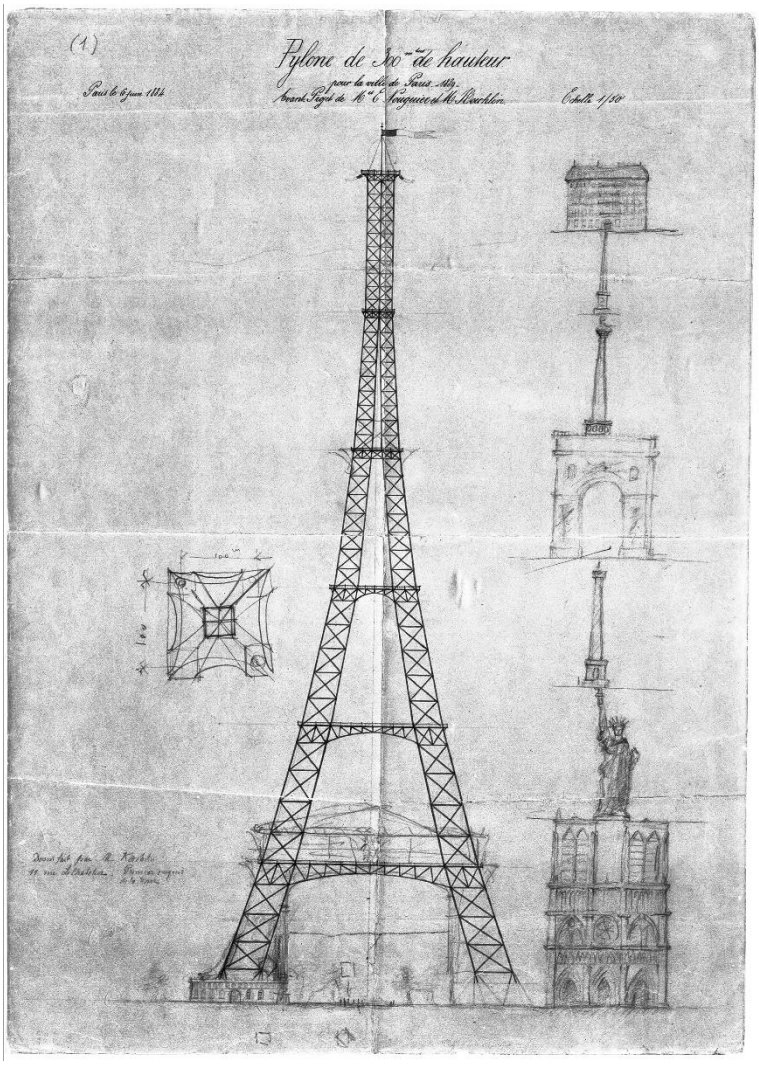


Fig. 58: Original drawing from Eiffel tower (1884) [100]

In the presented design exploration and for implication reasons the wind load is also assumed to be uniform. Like in the Chiasso shed study, the exploration aims at a symmetric structure and thus considers half of the design domain [Fig. 59, bottom left]. The setup is fairly simple and considers three supports, an applied load at the top and equally distributed lateral loading that simulates wind load. The application of bounded bar lengths imposes constraints to the exploration process but also resembles a more realistic design/engineering challenge.

Evolutionary algorithms are exploited for the generation and optimization of diverse Eiffel towers. The process is automated, linear – without any backtracking - and does not involve any intervention by the designer and aims at the minimization of the *static action* and the *degree of non-planarity* of the networks. The *DesignSpaceExplorer* component uses a predefined set of rules provided by the designer and transforms the initial interim network. The process considers an arbitrary total of 30 transformation steps per network. The exact number of transformations until the network has no interim forces, namely until it gets complete, is hard to predict. The bar length bounds make such a prediction impossible. Thus, at the end of 30 transformations most of the networks still contain interim forces.

The process considers a population size of 48 design candidates per generation and arbitrarily continues for 8 of them. Fig. 59 presents 288 (48 phenotypes x 8 generations) incomplete tower designs generated effortlessly and at once, in an automated way, and proves the succeeded breadth of exploration and the achieved diversity. Within 8 generations both objectives have been reached as illustrated in the graphs at the bottom right of the same figure. The latest generations demonstrate more efficient use of material and reduce the required number of node components. Consequently, both objectives lead to more sustainable and economic structures (within the population) without neglecting the initial goal of the PEER framework, the design space exploration. The designer, as always can either select the best performing design candidate, backtrack to previous stages, or build a new design branch to further explore.

A selection of some appealing networks of bars from Fig. 59 that can be further explored is illustrated in Fig. 60.

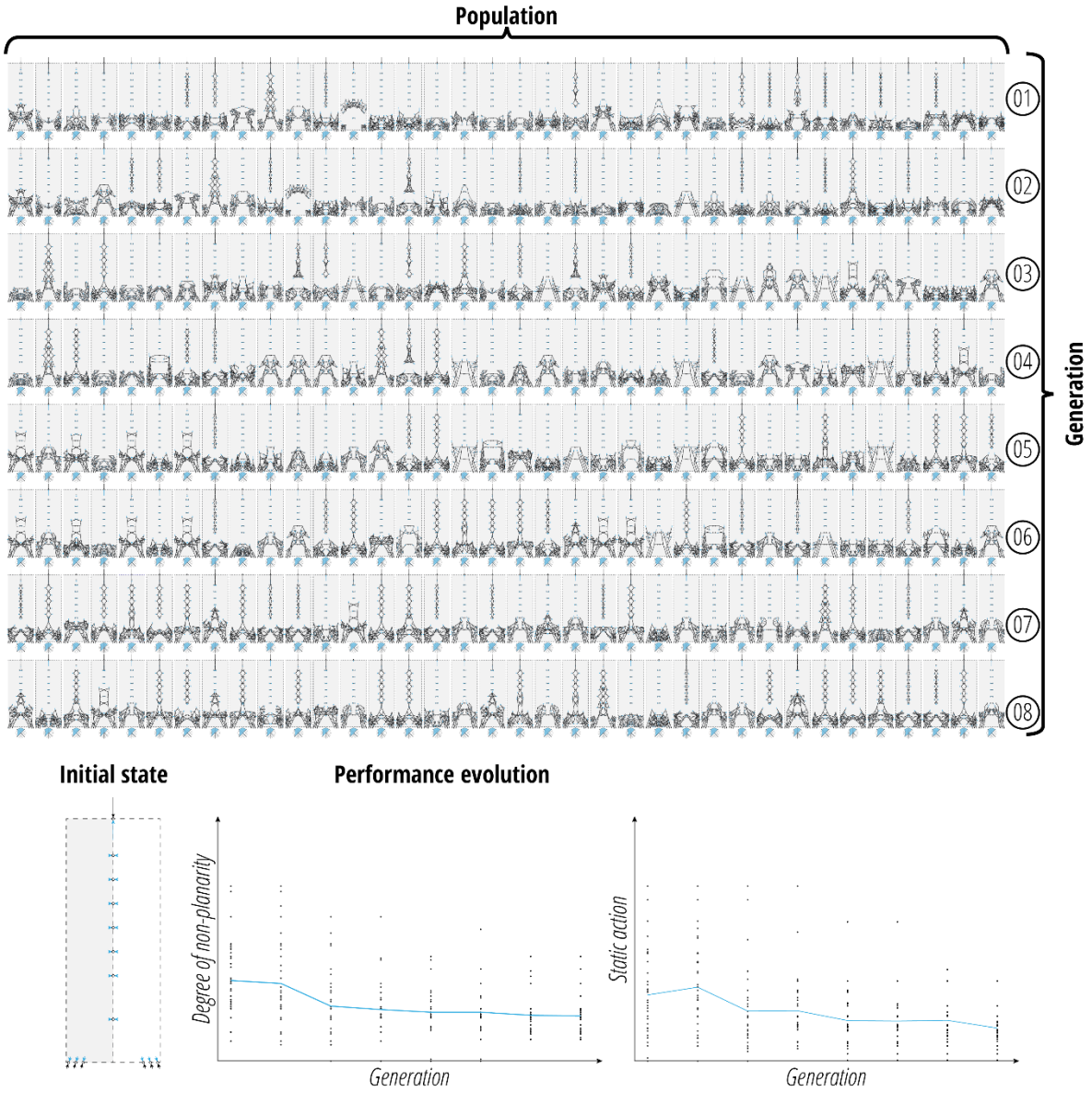


Fig. 59: Design space exploration map for the Eiffel tower

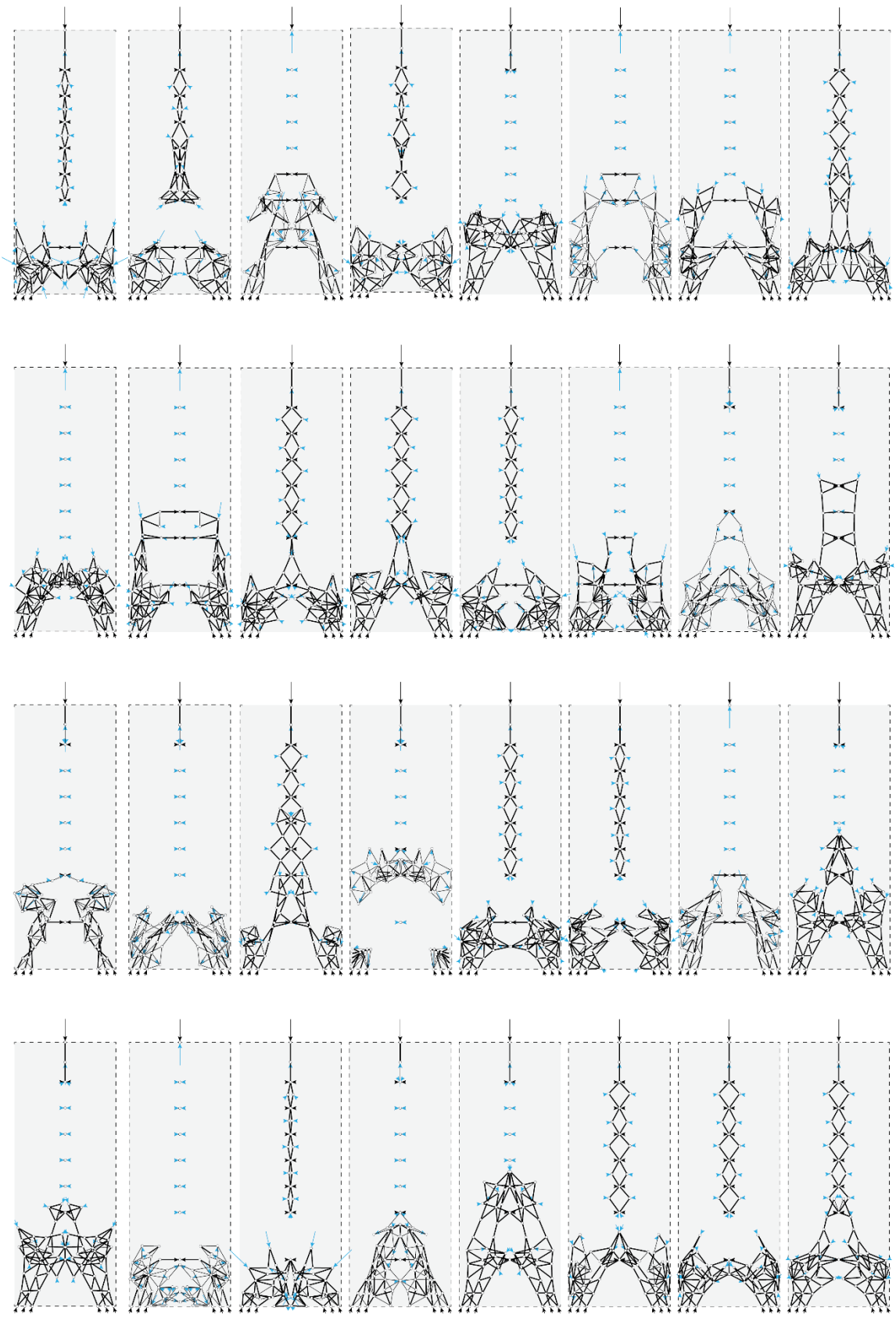


Fig. 60: Selection of appealing incomplete networks of bars

5 Discussions

This dissertation presented PEER – Policy-based Exploration of Equilibrium Representation - a computational design framework which supports a new design workflow to explore diverse conceptual structural design candidates within specified geometric domains by means of incremental transformations. Case studies demonstrated how PEER generates diverse networks of bars in static equilibrium. The current part reviews the entire framework and the supported workflow, thoroughly reports their contributions in the field of DSE, in comparison with their precedents, underlines their weaknesses and their limitations and provides recommendations for further development. Precisely, section 5.1 presents a list of specific contributions that PEER makes in the field of structural design exploration. Section 5.2 highlights short- and long-term developments in the PEER framework and implementations that will increase the designer control over the process and will subsequently promote *Libra* – the disseminated plug-in - as a powerful and versatile design space explorer. In the near future, its value is expected to be acknowledged both in academia and in practice. Section 5.3 outlines the types of structures that can or cannot be explored through PEER and teases future applications supported by human-machine peer collaboration. The part ends with section 5.4, a general conclusion on the conducted research and the author's vision about generative design, design space exploration and machine intelligence.

5.1 Contributions

The contributions that PEER brings to the field of conceptual structural design exploration are listed below. Primarily, the framework emphasizes on static equilibrium of all generated design candidates at all intermediate steps, interactive features that allow transformations and backtracking until satisfactory design candidates are generated and the dissemination into plug-and-play software named *Libra*.

1. Topological and geometric plurality as the main output

The generated design candidates are not resembling catalogued structural geometries and do have diverse topology. This attainment makes up the main output of the workflow and features the PEER framework as a form-finding engine of alternative topologies which are not known a priori. Rather, topology is defined during the decision-making process, allowing for alternative design pathways. The hypothesis is that the exploration of different topological configurations in static equilibrium yields unprecedented structural geometries.

2. Form-finding engine for diversity and plurality of spatial bar networks in static equilibrium

The generation of bar networks in equilibrium (strut-and-tie models) is a usual practice of early-stage conceptual structural design. Famous examples of strut-and-tie models are found in the design of reinforced concrete, where, if combined with stress fields and the lower-bound theorem of plastic theory, they provide safe equilibrium solutions [101]. While their generation extensively relies on optimization methods that yield a single solution, the established methods are mainly applicable to planar problems.

PEER framework has demonstrated the feasibility of generating spatial bar networks in static equilibrium in an incremental way and for various applications. The networks growth is always ensured within

explicitly defined geometric boundaries, convex or not. The final (complete) bar networks consist of axially loaded members in compression or tension, connected by nodes, and a set of external forces applied on those nodes only. Bar networks satisfy equilibrium constraints, as well as other optional constraints, at all intermediate steps of the transformative process. Though these bar networks are not deemed optimum, they are indicative equilibrium representations, potentially inspirational, for the early-(conceptual) design of structural forms.

The framework can guarantee plurality of bar networks with different topologies, but diversity is not always ensured. The latter is mostly related to the initial setup (design domain and configuration of external forces). Nevertheless, plurality alone successfully supports the hypothesis of provoking creativity and unveiling unconventional design candidates through PEER framework. Compared to [75], PEER allows the generation of spatial networks of bars in static equilibrium and has significantly revised the set of transformative rules. The latter grants higher level of control to the designer and allows him/her to steer the design exploration towards desired directions more easily.

3. Geometry and topology impartial transformations

The provision of multiple valid solutions questions the level of diversity and novelty each of these solutions reaches. The suggested transformation policy is neither type-specific nor resorts to common topology patterns – i.e., triangles. Exploration is driven by an incrementally applied policy defined by four high level rules. These rules collaboratively describe static equilibrium constraints. This single policy is enough to impartially (regarding topology related inputs) generate bar networks considering geometric constraints only, such as geometric boundaries or bounded bar lengths, if any, imposed by the designer. Consequently, the process is freed from:

- narrow exploration and lack of diversity imposed by a specific type of geometry (e.g., tower, pylon etc.).
- narrow exploration and lack of diversity imposed by a specific topology.
- lack of novelty imposed by the repetition of recursing modular units (e.g., triangles or pyramids), in need of retaining static equilibrium.

Concretely, the policy generates any type of topology for any given loads and therefore goes beyond conventional structural geometries.

4. Convergence / Divergence “on demand”

While the network topology is defined incrementally, the process converges/diverges “on demand”. This feature has a twofold meaning: (a) the designer has direct control over the impact of each transformation through a balanced ternary rule – i.e., -1 for *convergence*, 0 for *stagnation*, 1 for *divergence* - and (b) the *minimum* number of transformations until the process terminates is known in advance.

Unlike other form-finding processes, PEER operates in a bidirectional way. While, the goal is completeness of incomplete networks through convergence, *intentional divergence* for a finite number of steps is not excluded as a design input. This seems contradictory to most established form-finding processes, that solely operate in converging mode and designers have no control over it until the process ends. Due to this type of bidirectionality the framework allows for broad exploration of the design space.

The transition from an interim network to a complete one presupposes the replacement of interim forces by bars in tension and/or compression. Thus, the *minimum* number of transformations coincides with: (a) the number of initial interim forces in the network and (b) the minimum number of introduced nodes in the network. Nevertheless, these assumptions are not always true. Bounded bar lengths, for example, make such a prediction absurd. Because of length constraints, many of the desired transformations are

likely to become infeasible. Consequently, the process is respectively prolonged - i.e., more transformations introduce more nodes and more bar elements.

5. User-controlled, two-directional process

At every intermediate transformation step, the designer has the possibility to interfere with the transformative process. Either to change the rule, or its parameters, or to manually select the forthcoming parents to crossover and mutate. In other words, the designer incrementally and interactively explores design candidates by setting preferred design states as new starting points and by creating new design branches on top of them. Therefore, thanks to the retainment of interim equilibrium, there is no commitment on the design decisions. Namely, the designer can always backtrack to previous states that satisfy personal, functional or performance criteria without risking static equilibrium.

6. Post-processing-free results to impose equilibrium

The abovementioned interruption of the transformative process is only feasible thanks to the equilibrium-aware policy definition. Originally, this means that each generated network maintains (interim) static equilibrium at every intermediate step. Concretely, embedding static equilibrium into the policy definition, liberates the process from any post-processing to impose equilibrium unlike other form finding approaches.

7. Form exploration and performance optimization

Interactive Genetic Algorithms are not new in the field of design space exploration. Nor the presented workflow is deemed exploratory because of exploiting them. The real benefit of their fusion is twofold: (a) they amplify the exploration capacity of incremental policy-based design and (b) the generation and storage of evolving populations builds a large database of diverse designs.

The multimodal nature of multi-variable optimization offered by Genetic Algorithms makes the selection among large populations of candidates a challenging decision. Plotting all phenotypes simultaneously and showcasing their absolute performance with regards to various metrics makes qualitative and quantitative comparison easier [Fig. 59]. The interactive functionalities within the *DesignSpaceExplorer* component [section 4.4] of *Libra* as described in sections 4.3.4.2 and 4.3.4.3, adds value to the PEER framework and eventually formulates a user-friendly design toolkit for the generation and exploration of spatial bar networks in static equilibrium.

8. Libra toolkit

The PEER framework has been disseminated through the parametric toolkit of *Libra*, a plugin operating in framework Grasshopper platform, soon to be published in www.food4rhino.com. Its development allows designers to interactively explore the design space and filter out structurally valid design candidates, which can be further transformed or optimized. More specifically, *Libra* grants access to the PEER framework to external users beyond the author himself.

Extra attention has been paid to keep the workflow simple, compact and intuitive. In this direction, each step of the procedure is fully executed within one Grasshopper component [Fig. 31]. This compactness retains the size of the entire Grasshopper definition small. Furthermore, it prevents the creation of a spaghetti tangle that discourages other designers from further manipulations within the environment of Grasshopper. Regarding intuition, the process is didactic and provides feedback. When a set of input choices results in a non-feasible transformation, the designer is accordingly informed, making it a favorable educational tool to help architects understand how to design along with static equilibrium constraints. Another feature that enhances this understanding is the upon-request plotting of the chosen forces' feasibility domain in compliance with the desired entropy rate.

The PEER integration in the parametric platform of Grasshopper is benefited by the interactivity and the user-friendliness offered by the platform. Focusing on interactivity, the toolkit instantly visualizes any input modifications or past design-states retrieval request. Investing into user-friendliness, the concept of dynamic Grasshopper components has been adopted for those that define the policy rules and require supplementary parameters. Therefore, the list, the type and the number of input parameters on the left side, where input variables are plugged, is updated according to the value of the first input parameter [Fig. 34, Fig. 36, Fig. 37, Fig. 38]. Eventually, the size of the Grasshopper components remains small and indicates the mandatory provision of input values for all the provided slots.

The genetic properties, namely the DNA in biology terms, of each design candidate is stored as a list of design policies (a triple set of rules) that reflects the designer decisions. By encoding a bar network into a .txt file, its reproduction is facilitated upon identical conditions (design constraints and applied loads). It is interesting that the information to reproduce a bar network does not consist of a connectivity matrix and cartesian coordinates to describe the nodes, but rather rules that define policies.

The toolkit has been already tested by students in the course of an elective master class offered at EPFL's Master in Architecture [Fig. 50]. Their feedback has helped the author not only to improve the toolkit itself but also prove the potential of the supported design workflow itself. Concretely, it has successfully demonstrated the diverse topologies and geometries of design candidates a policy-based design approach can provide [Fig. 47, Fig. 49, Fig. 50].

Through the case studies it becomes evident that the current implementation presents limitations, which require further development and work, as will be described in sections 5.2 and 5.3.

5.2 Future work

While the assets and benefits of this dissertation have been highlighted, several issues require further development. The following sections describe short- and long-term additions and/or edits, as well as aspects for further investigation, that are considered. Further development is grouped in two fields: the methodology and the implementation, each described in different section. The former includes advancements that will bring new functionalities that increase the designer's control over the design process. The latter includes advancements that will improve the overall framework performance and will reconsider the role of the machine and the designer. The performance related edits will contribute to a faster responsive user experience.

5.2.1 Methodology related

1. Integrate local optimization objectives as explicit rules

The preferred node placement rule controls the location of the introduced new node *P* and is defined implicitly or explicitly. Precisely, most of the rules define geometric sub-domains where the node is introduced in compliance with the desired entropy rate. Most of the times these sub-domains describe additional constraints applied on the feasibility domain. For the rest of the node placement rules, the location is explicitly defined via cartesian or homogeneous coordinates.

Currently, none of the implemented node placement rules considers the network's performance over some kind of metrics. Those expressing sub-domains operate blindly and randomly introduce a new node, while the explicitly defined locations are not assessed as performative or not. However global optimization of the entire network is feasible when for example explicit node placement rules are used - i.e., express *P* location via homogeneous coordinates - along with genetic algorithms. In such cases,

the coordinates are described as genotypes and the performance over specific metrics becomes the objective to minimize / maximize. Typically, the youngest populations outperform the oldest ones [section 4.5.2].

An alternative approach to consider objective functions through rules selection in the near future follows. In the list of nodes placement rules, each will represent an optimization objective at a local level. In such a case, each transformation step corresponds to an optimization problem solved deterministically, rather than stochastically, and returns the optimum (at a local transformation level) coordinates of node *P*, among other output values. The feasibility domain, namely its solution space, describes the constraints and the node placement rule describes the objective. The drawback of such an approach is that the introduction of optimum-located nodes, at a transformation (local) level, does not guarantee optimum overall (global) performance for the network. However, the outcomes of such an implementation should be compared with those output from the current implementation.

2. Manipulate the pools of rules

Genetic algorithms (GA) have been integrated in the PEER framework to produce diverse, but still structurally relevant, networks more efficiently–i.e., lower static action, less intersecting bars etc.–than before, when brute-force and random values were applied to generate networks with PEER framework alone. The stochastic nature of metaheuristic algorithms augments the breadth of design candidates but also returns optimized networks for various metrics. Currently, GA genotypes result from the manipulation of supplementary rule parameters. The rules themselves are predefined and not shuffled among a pool, as part of stochastic search mechanisms. To increase results' heterogeneity and to avoid numerical repetitions of parameters along the transformative sequence of each generated population, each step is described by identical rules customized by different constitutive parameters respectively. For example, while the forces selection and node placement rules along a transformative sequence of 20 steps remain identical throughout, the *t_param* (supplementary parameter of an implicit node placement rule) is manipulated by the GA exploratory mechanism.

Originally, the choice of each of the three rules (*forces selection*, *node placement* and *force indeterminacies*) is made out of a pool of rules. Instead, in the future, the choice of the first two rules will be made exclusively in an automated way and never manually by the designer, in compliance with performance related objectives - i.e., complete network's static action. In this case, the exploratory mechanism operates on an abstract level. The sequence of rules, rather than their parameters, are subject to manipulations. Due of this shift, the definition of rules will become more explicit without supplementary parameters to describe them.

Consequently, the designer's control level over the process will be limited to minor decisions (the *entropy rate* and the *force indeterminacies rule* only) that are likely to turn the process into a black box. Once again, the outcomes of the abovementioned approach should be compared with those output from the current implementation.

3. Design for a stock of bar elements

The *node placement rule* directly defines the location of the new node *P*. Indirectly, it also affects the lengths of the introduced bars. The current implementation considers constraints that force the placement of the new node in geometric domains to ensure that the introduced bars remain within predefined bounds. Nevertheless, the bar lengths do not comply with any standardized lengths available in the market, manufacturing constraints or other reasoning.

Considering that the construction industry is responsible for a vast amount of global material use and greenhouse emission [102], the design process will be reversed with the aim of reusing structural

elements. Precisely, the bars will be taken from an available stock in compliance with the concept of reuse in structural design. Practically, the pool of *node placement rules* will be further enriched with a new rule that operates on stock-constrained design of structures. Therefore, the available bars in the stock will first replace the interim forces and the location of the *P* node will be determined afterwards, as a consequence of the bar lengths and the bars intersection – i.e., for all in-between topological configurations in Fig. 16. Exceptionally and for cases that the available stock does not provide bar lengths that allow the convergence of the interim bar network, new bars – length bounded within a domain or not – will be introduced. This addition will demonstrate how design space exploration can still lead to structures with less environmental impact through the combination of reused and new, when necessary, bar elements.

4. Manipulate the axial forces along the bars

The application studies presented in [91] showcase how the use of bar length bounds tame the transformative process; undesired network density and, occasionally, high nodal valences get decreased compared to preliminary studies. Like discussed before, as part of the future development, additional constraints and bounds can easily get integrated in PEER. Mechanics-related bounds to control the developed axial forces and constrain their magnitude will allow designing for specific materials and will filter out buckling-prone designs.

The force indeterminacies rule explicitly defines the force magnitude (stresses) developed in the new introduced bar when the static equilibrium equations system is indeterminate – i.e., for all cases of *stagnation* and *divergence*. Currently, the force magnitudes are arbitrarily defined. If the material to be used has been decided in advanced, specific rules can reflect certain objectives or needs related to the material properties. Examples are: (a) exploration for material usage minimization or (b) exploration with an available stock whose structural capacity per element is already known. Precisely, certain utilization values will constrain the force magnitudes. For example, for a CHS bar with 60.3 mm diameter and 3mm thickness, made of steel S235, utilization between 75% and 80% implies force magnitudes between 95kN and 102kN. When the material is unknown, rules will bound the force magnitudes within predefined domains in a similar way that bar lengths are bounded.

A set of such rules will designate the PEER, as well as the *Libra* plug-in, as a powerful tool to design, engineer and construct conceptual structural forms with remarkable control over multiple aspects of the process.

5. Cluster the representative phenotypes, not the genotypes

The part of the framework that handles evolutionary exploration though metaheuristics builds on top of the open-source implementation of *Biomorpher* [3]. To integrate it into *Libra*, necessary modifications were made. The *Biomorpher* computational core remained untouched, including the clustering process which is based on the input (numerical) parameters. Typically, design candidates constructed by numerically close parameters are expected to resemble each other. Thus, clustering them based on the numerical proximity of their input parameters is a robust and straight-forward process. *Biomorpher* operates on a K-means algorithm that shrinks the size of generated design candidates into a short list of 12 representative candidates. The design space compactness boosts the interactivity aspect of *Biomorpher* and the designer effortlessly selects the parents of the next generation out of a short list.

In policy-based design approaches, genotypes clustering, namely clustering based on input parameters, does not provide the expected results. It has been observed that slight numerical differences in the policy, or the rule parameters, result in diverse networks of unique topology. According to *Biomorpher* documentation, the clustering implementation is based on the genotypes and therefore, currently in *Libra*,

the dashboard of representatives represents a random fraction of the generated design candidates. To provide to the designer an original set of representatives, two scenarios are considered:

- changing the current K-means implementation of *Biomorpher* in a way that clustering is based on the output results, phenotype clustering. Indicative features that would be compared for the case of networks include: the number of bars, the number of nodes, the valance of each node, the length of bars etc.
- skipping the clustering and plotting the full list of phenotypes generated by the GA. In this case the designer will have the possibility to select the parents of the upcoming generation directly from the entire population. Nevertheless, this shift comes with its own drawbacks. Traditionally, in the history of interactive evolution clustering is highly recommended to facilitate selection among long lists of options [38]. Overtaking such a facilitating feature will make the parents selection a tedious and boring task that increases the designer's fatigue [31].

6. Impose type-specific transformations

It has been observed that recursion of specific topological patterns (i.e., triangulation) in the entire network of bars is not occurring frequently. This behavior has been included in the primary PEER objectives. However, it does not mean that the generation of fully triangulated networks for example is not feasible if the policy is accordingly set up. Similarly, the transformation of a trinomial of interim forces into a closed triangle is also a pattern that is not currently considered on purpose. It is believed that the selective, occasional, use of such transformations will increase the designer's control level and lead to efficient and fast-converging networks with more creative topologies than triangulation that ensures static equilibrium.

Other specific transformations that could impose the tension bars to be all either horizontal or vertical. Such a strict constrain would be extremely useful to inform the placement of rebars in reinforced concrete.

7. Fuse neighboring nodes

Many of the illustrated bar networks in the application studies [section 3.4] demonstrate an "introvert" behavior characterized by dense in terms of topological complexity - i.e., multiple nodes populated close to each other. Post-processing the bar networks will handle this behavior. Neighboring nodes within a user defined tolerance, but not only, will be possible to get fused to a new average position and static equilibrium will be imposed to the updated network. The adjacent bars that superimpose each other will be replaced with new bars with force magnitude equal to the resultant force. This addition aims at fixing unintentional behaviors of the process itself, as imposed by human choices. The integration of machine intelligence in the process, as described later, is likely to make this feature obsolete.

It is important to underline that the recommendation for post-processing is triggered by the desire to simplify the networks topological complexity rather than to impose static equilibrium. As it has already been mentioned, all networks - both the interim and the complete ones - satisfy static equilibrium all along.

8. Re-consider the loss of possible design candidates

The location of node *P* is defined by node placement rules as described in section 3.1.4.3. Currently, the most favorable node placement rule is the *User Selected Parameter*, which explicitly, via homogeneous coordinates, describes the *P* location. Although this rule has been proven very handy for most of the presented application studies it comes with its own drawbacks. As homogeneous coordinates exist in the [0,1] domain, their resolution (precision in number of decimal points) is of utmost importance. If the

feasibility domain represents a segment, the homogeneous coordinates describe a point on it. When the *feasibility domain* stretches in two- (closed polygon) or three-dimensions (volume), it is discretized into cells or voxels respectively. The discretization density follows the decided precision. Many possible positions are not considered/lost when the resolution is relatively low. Even though this is an aspect that requires the designer's attention, it cannot be considered of minor importance and should not be neglected during the settings setup. If not, other explicit ways to define the node *P* location should be considered.

5.2.2 Implementation related

1. Accelerate three-dimensional boolean operations

The definition of the *feasibility domain* is supported by the computation of the *entropy rate domain* and the *constructability domain* executed sequentially. This procedure must be executed prior to every transformation (exceptions apply; section 3.1.5.3) and it is likely to be repeated multiple times, until a non-empty intersection of the two domains is found. The involved algorithms are custom made and calculate the Isovist [90] in two and three dimensions. The abovementioned domains are represented by RhinoCommon objects; *Point3d* and *Curve* for monomials or binomials acting on planar *design domains*; *Point3d*[4], *Curve*[5] and *Brep*[6] for monomials, binomials or trinomials acting within spatial *design domains*. Looking at Fig. 22, it is evident that the computation of the *constructability domain* for *stagnation* and *divergence*, requires multiple, recursive boolean operations which are known to be computationally exhaustive. In order to limit them, the current implementation only considers one void per *design domain*. The consideration of length bounded bars already adds more exhaustive operations on top of the long list.

As understood, the *feasibility check* [Fig. 19, feasibility checks in rhombuses] itself significantly slows down the entire workflow and deviates from low response times expected in digital design exploration. First attempts to use parallel computing (by replacing *for* loops with *Parallel.For* method) have not performed fast enough. Replacing *Brep*[7] and *Curve*[8] classes by *Mesh*[9] and *PolylineCurve*[10] classes respectively, when possible, is seriously considered. Alternatively, other libraries than RhinoCommon, like *ShapeOp* (https://www.shapeop.org/), *libigl* (https://libigl.github.io/) or *CGAL* (https://www.cgal.org/) will be considered. Replacing the domains (*entropy rate* and *constructability*) generating algorithms by a new custom C++ library is part of the author's future plans too.

---

[4] *https://developer.rhino3d.com/api/RhinoCommon/html/T_Rhino_Geometry_Point3d.htm*

[5] *https://developer.rhino3d.com/api/RhinoCommon/html/T_Rhino_Geometry_Curve.htm*

[6] *https://developer.rhino3d.com/api/RhinoCommon/html/T_Rhino_Geometry_Brep.htm*

[7] *https://developer.rhino3d.com/api/RhinoCommon/html/T_Rhino_Geometry_Brep.htm*

[8] *https://developer.rhino3d.com/api/RhinoCommon/html/T_Rhino_Geometry_Curve.htm*

[9] *https://developer.rhino3d.com/api/RhinoCommon/html/T_Rhino_Geometry_Mesh.htm*

[10] *https://developer.rhino3d.com/api/RhinoCommon/html/T_Rhino_Geometry_PolylineCurve.htm*

2. Invest on three-dimensional application studies

The long computing time required for spatial design domains, as identified above, has prevented the generation of multiple application studies in three-dimensions. Precedent grammar-inspired approaches have been mainly limited to two dimensions. Even though only one spatial application has been demonstrated in this dissertation, PEER framework and its implementation already consider the third dimension. In the near future, after fixing some algorithmic bottlenecks, more spatial application cases will be disseminated to highlight the exploration capacity in three dimensions too.

3. Human-machine peer collaboration

Interactive evolutionary design exploration successfully enables a first level synergy between the human and the machine. Possibly without it many of the demonstrated results would not even be conceived. Overall, PEER allows designers to generate statically valid networks of bars without any knowledge of structural mechanics but the level of control over the process is debatable. Efficient design space exploration and accurate control over the generation of bar networks, rely on understanding the impact of design decisions made during the process. Human intelligence has limited capacity of predictability though.

PEER as generative process has similarities with chess playing and is significantly constrained if purely based on human logic. Every chess piece has its own unique way of moving. Similarly, a set of rule choices (a policy) transforms the networks in its own unique way and creates its own design branch. Certain movements in chess checkmate the opponent faster or slower. In PEER, convergence is accelerated, or decelerated, by certain choice sequences. Like in chess playing, the human mind foresees and processes ahead a limited number of movements, or policies, that bring the player closer to the victory, or convergence. In chess, the success rate of specific tactics competes with the uncertainty brought up by the opponent's defensive movements (dynamic constraints). In generative design, uncertainty comes from the design constraints, namely static equilibrium and design domain (static constraints) or delayed requests (dynamic constraints). Static or dynamic, these constraints have a huge impact on the decision-making of the right tactics or transformation policies respectively.

Future development will advance the efficiency of the current PEER framework, that is limited to human intelligence, through machine intelligence. The machine, promoted as an intellectual designer, will foresee and prevent undesired design situations, imposed by human-made decisions. Instead, the machine will suggest sequences of design decisions that allow for efficient exploration and sufficiently controlled design candidates. Its intervention though will still allow the (human) designer to steer the design candidates with regards to subjective criteria and preferences (i.e., aesthetics). Concretely, future development will investigate the integration of reinforcement learning into the developed framework and will determine if it grants additional control to the structural design candidates.

The results of this research will promote the machine as collaborative partner during the design process, that contributes with its own intelligence towards the final design concept. The machine-assisted design space explorer will dodge premature design fixation and will provoke creativity and diversity, whilst retaining high level of control over the process. Concretely, it will unveil highly controlled unconventional forms and will synthetize systems that provide a balanced answer to complex architectural /structural contexts with reduced environmental impacts and controlled production costs. Like this, certain objectives at a global level will be reached, i.e., non-intersecting bars etc. Such an advancement is expected to reveal PEER's full potential for the generation of highly controlled spatial bar networks in static equilibrium. Inside academia, direct applications to the teaching of structures in schools of civil engineering and architecture are foreseen. Outside of academia, interested architectural and engineering offices are to be benefited by the automated and high-level intelligent generation of conceptual structural

design candidates. Their workflows are expected to get simpler and shorter with novel and bespoke solutions at no additional cost.

5.3 Applications and opportunities; capabilities and incapabilities

1. Applications

PEER is intended to be used for exploring diverse schematic, early-stage topologies of static equilibrium representations and eventually facilitate conceptual structural design. Equilibrium representations or bar networks are well suited for describing force paths. Planar and spatial reticulated structures are the direct materialization of bar networks into structural systems. Nevertheless, bar networks could be further applied in a very wide range of planar and spatial structural systems of almost any building material (e./g steel, wood, concrete, glass, ceramics, polymers), including their combination into composites:

- statically determinate and indeterminate structures (e.g., roof trusses, building skeletons, stadiums, bridges), including non-triangulated structures (e.g., simply connected funicular geometries), hinged and moment-resisting structures (i.e., truss and frames), pre/post-stressed structures, self-stressed structures (e.g., tensegrities);
- strut-and-tie networks in continuum materials, satisfying plastic design assumptions (e.g., reinforced concrete plane elements), including thrust networks within compression-only structures (e.g., masonry arches or shells).

Overall, PEER and its implementation provide an intuitive design tool to explore the design space in a way that analysis software does not allow for. 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 development. Hence, as a tool it is not applicable to hydraulic engineering (dam design, dredging engineering, coastal engineering) or geo-engineering (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.

Above all though, the one to be benefited the most by *Libra* is the human designer, who can explore the design space making sure that all of the design candidates are structurally valid.

2. Opportunities

The following graph illustrates the types of structures, within the author's interest, served, or expected to be serve in the near future, by PEER. The X-axis corresponds to the types of structures, sorted out with regards to their dimensionality as final products. PEER is currently focusing on truss systems (direct translation of static equilibrium representations to bar networks and standing structures). Y-axis corresponds to the means available, starting from hand-drawing until collaborative systems. Specifically:

- hand drawing relates to the use of ruler and pencil
- computerized drawing relates to the use of computers in a laborious and dull way that ensures nothing more than speed
- computational drawing relates to the use of computers and parametric algorithms
- integrated interactive drawing relates to the use of computers in an automated way and the practical involvement of humans in the process

- human and machine collaborative drawing relates to the outcome of a joint effort coming from two equal collaborative peers

Z-axis corresponds to geometric approaches that consider static equilibrium and thus can be used for conceptual structural design exploration. Specifically:

- graphic statics comprises the famous analytical approach that builds on geometry rather than policies. Based on the dual relationship between two diagrams it has been repetitively chosen by master builders such as Robert Maillart and Rafael Guastavino among others to explore and finally design highly efficient structures. Nowadays its contribution in the field of structural design has evolved from a purely analytical tool to a highly explorative one.

Policy-based structural design relates to incremental approaches inspired by grammars and is ruled by policies. PEER is such an example and is expressed in the following 3 aspects:

- unconstrained relates to policies that consider static equilibrium only; everything else is defined randomly
- constrained relates to policies that consider additional constraints besides static equilibrium
- evolutionary/intelligent relates to policies that describe certain objectives besides static equilibrium consideration.

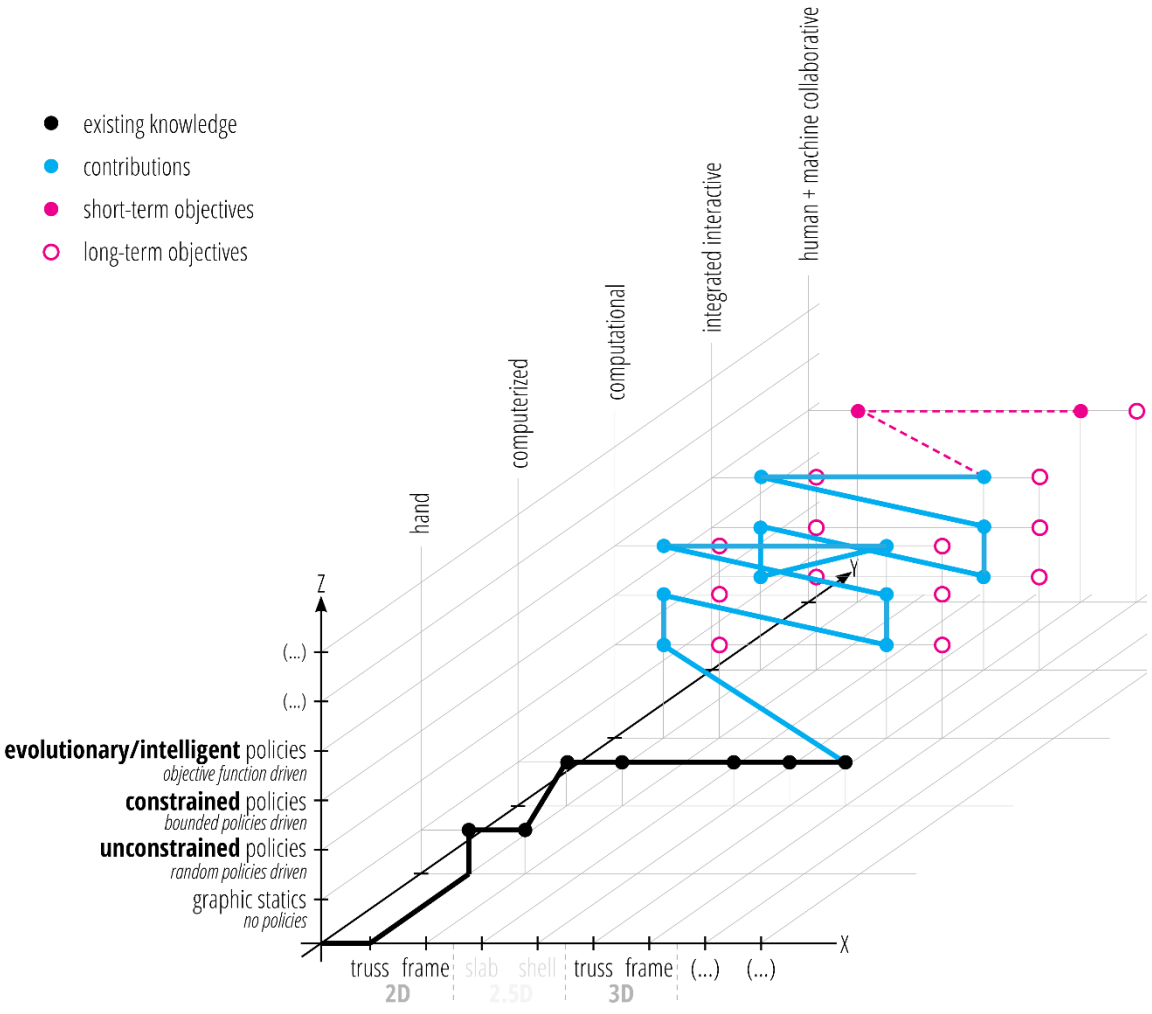


Fig. 61: PEER capabilities, milestones and future goals

The black trail acknowledges the existing knowledge in the field of graphic statics and thus clearly indicates that this dissertation does not provide any contribution in the field. The cyan trail highlights how the PEER approaches facilitate the design of certain types of structures, and with which means. The magenta trail indicates the short-term objectives for future development with an emphasis on the human machine collaboration. The long-term objectives include the framework extension for the design of frames, illustrated with empty magenta circles [Fig. 61]. Thus far, frame structures have not been considered. It is expected though that the existing PEER theory is easily extendable, simply by introducing interim moments along with interim forces.

5.4 Conclusions

The use of geometry in finding the equilibrium of forces has a history as early as 1586 [103]. PEER relies on a transformative policy that integrates geometric constraints to ensure static equilibrium too. The policy definition expresses design intentions in a descriptive way through rules. The introduced nodes, connect in various topological configurations with interim bar networks in static equilibrium, transform them in compliance with the desired entropy rate and incrementally grow them larger until the pool of interim forces is empty. Each transformation step is highly dependent on the precedent steps and the node-by-node process shapes accordingly the solution space –geometric domain - of the upcoming transformation step. The *step-specific geometric domains* along with *policy-imposed design intentions* per transformation step result in highly combinatorial design pathways and facilitate broad and thorough design space exploration. Design space exploration is also facilitated by two more features; *the possibility to backtrack and browse through previous valid states* and *the fusion with interactive metaheuristic algorithms*. All four aspects have supported the exploration and generation of diverse, from a topological and geometrical point of view, bar networks in static equilibrium and established the Policy-Based Exploration of Equilibrium Representations (PEER). Last but not least, this dissertation proves that like in graphic statics, with the help of geometry designers can confirm static equilibrium. In PEER, equilibrium constraints are solved algebraically but geometry describes the feasibility domains where new nodes are incrementally introduced while ensuring static equilibrium.

Grammar-inspired generative design in the form of shape grammars bloomed at MIT and stimulated diversity and creativity more than 50 years ago. Since then, dozens of researchers charmed by grammars' beauty have contributed to their semantics and developed various interpreters to feature them as design approaches. The overwhelming growth of interest in the field of shape grammars the last decade resulted in the founding of research labs fully dedicated to shape grammars (*Shape Computing Lab* at Georgia Institute of Technology, *Design & Computation Group* at University of Lisbon) and others, partly affiliated to grammars (*Design Automation Laboratory* at National University of Singapore, *Genesis Lab* at Delft University of Technology). International conferences, such as eCAADe and CAADRIA, for the course of many years now have been hosting separate sessions on shape grammars and the latest relevant findings are disseminated. Nevertheless, most of these projects has been and have remained theoretical and only few of them have been implemented for real design applications (urban or structural). Due to recent technological advancements and the computers ubiquity in design professions, research in the field of design space exploration with digital means has been observed. Shape grammars for design space exploration has never managed to escape the realm of academia and practitioners hesitate to integrate it into their design workflows. In my opinion, the main reason is the lack of high-level control and the belief that design cannot be ruled by randomness. The development of appropriate framework will hopefully shift this way of thinking and finally give shape grammars the chance to unveil its potential in the design of the built environment.

The presented research and the development of PEER are inspired by this emergent spurt in both fields. The possibility to incrementally generate and browse through infinite design candidates that satisfy static equilibrium without any prior knowledge in mechanics is tremendous. Although it already offers a complete workflow its development is not complete. The ultimate hope of this research is to leave its footprint as a legitimate generative, conceptual, structural design workflow both in academia and in practice. Its further development towards fully controlled manipulations will ease this achievement. Last but not least, the potential to effectively achieve a seamless peer collaboration between the human and the machine would be a great future milestone and grammar-inspired generative design offers unique opportunities in that regard.

References

[1] A. Kilian, Design Explorations through Bidirectional Modeling of Constraints, (2006). http://www.designexplorer.net/download/Kilian-phd-arch-2006.pdf.

[2] Boden Margaret, Dimensions of Creativity.pdf, A Bradford book, 1994.

[3] J. Harding, C. Brandt-Olsen, Biomorpher: Interactive evolution for parametric design, Int. J. Archit. Comput. 16 (2018) 144–163. https://doi.org/10.1177/1478077118778579.

[4] J. Norman, R. De'ath, J. Carr, G. Knowles, O. Broadbent, R. Harpin, I. Lloyd, Conceptual design of buildings, IStructE Ltd, 2020.

[5] B.C. Paulson, Designing To Reduce Construction Costs, ASCE J Constr Div. 102 (1976) 587–592. https://doi.org/10.1061/jcceaz.0000639.

[6] I.C. Parmee, Evolutionary and Adaptive Computing in Engineering Design, Springer, London, UK, 2001.

[7] J. Burry, M. Burry, The New Mathematics of Architecture, Thames & Hudson Ltd, London, UK, 2010.

[8] K. Shekhawat, N. Upasani, S. Bisht, R.N. Jain, A tool for computer-generated dimensioned floorplans based on given adjacencies, Autom. Constr. 127 (2021) 103718. https://doi.org/10.1016/j.autcon.2021.103718.

[9] T. Kotnik, Digital Architectural Design as Exploration of Computable Functions, Int. J. Archit. Comput. 8 (2010) 1–16. https://doi.org/10.1260/1478-0771.8.1.1.

[10] Simon H.A., The structure of ill structured problems, Artificial Intelligence (1973) 181-201.

[11] H.W.J. Rittel, M.M. Webber, Rittel, Horst W. J., Dilemmas in a General Theory of Planning , Policy Sciences, 4:2 (1973:June) p.155, Policy Sci. 4 (1973) 155–169.

[12] A.T. Purcell, J.S. Gero, Design and Other Types of Fixation or Is Fixation Always Incompatible with Innovation ?, Des. Stud. 17 (1996) 363–383.

[13] C. Mueller, J. Ochsendorf, Combining structural performance and designer preferences in evolutionary design space exploration, Autom. Constr. 52 (2015) 70–82. https://doi.org/10.1016/j.autcon.2015.02.011.

[14] K. Deb, Multi-Objective Optimization using Evolutionary Algorithms, Wiley, 2001.

[15] R. Kicinger, T. Arciszewski, K. De Jong, Evolutionary computation and structural design: A survey of the state-of-the-art, Comput. Struct. 83 (2005) 1943–1978. https://doi.org/10.1016/j.compstruc.2005.03.002.

[16] G. Box, Evolutionary Operation: A Method for Increasing Industrial Productivity, J. R. Stat. Soc. 6 (1957).

[17] J. Holland, Adaptation in Natural and Artificial Systems, 1975.

[18] D. Goldberg, Genetic Algorithms in search, optimization and machine learning, Addison-Wesley Publishing COmpany, Inc., 1989.

[19] J. Koza, Genetic Programming; On the programming of computers by means of natural selection, A Bradford book, 1992.

[20] I. Rechenberg, The Evolution Strategy. A Mathematical Model of Darwinian Evolution, in: E. Frehland (Ed.), Synerg. --- From Microsc. to Macrosc. Order, Springer Berlin Heidelberg, Berlin, Heidelberg, 1984: pp. 122–132.

[21] D.B. Fogel, An introduction to simulated evolutionary optimization, Evol. Comput. Foss. Rec. 5 (1994) 3–14. https://doi.org/10.1109/9780470544600.ch1.

[22] Y.C. Toklu, G. Bekdas, S.M. Nigdeli, Metaheuristics for Structural Design and Analysis, Wiley, 2021.

[23] P. Janssen, J. Frazer, T. Ming-Xi, Evolutionary design systems and generative processes, Appl. Intell. 16 (2002) 119–128. https://doi.org/10.1023/A:1013618703385.

[24] D. Goldberg, The Design of Innovation: Lessons from and for Competent Genetic Algorithms, Springer, 2002.

[25] D. Goldberg, M.P. Samtani, Engineering Optimization Via Genetic Algorithm., in: 1986: pp. 471–482.

[26] S. Adriaenssens, P. Block, D. Veenendaal, C. WIlliams, Shell Structures for Architecture: Form Finding and Optimization, Taylor & Francis - Routledge, London, UK, 2014. https://doi.org/10.4324/9781315849270.

[27] M. Turrin, P. Von Buelow, R. Stouffs, Design explorations of performance driven geometry in architectural design using parametric modeling and genetic algorithms, Adv. Eng. Informatics. 25 (2011) 656–675. https://doi.org/10.1016/j.aei.2011.07.009.

[28] M. Turrin, P. Von Buelow, A. Kilian, R. Stouffs, Performative skins for passive climatic comfort: A parametric design process, Autom. Constr. 22 (2012) 36–50. https://doi.org/10.1016/j.autcon.2011.08.001.

[29] P. von Buelow, Paragen: Performative exploration of generative systems, J. Int. Assoc. Shell Spat. Struct. 53 (2012) 271–284.

[30] R. Dawkins, The blind watchmaker, 30th Anniv, Penguin Books Ltd, London, UK, 2006.

[31] H. Takagi, Interactive evolutionary computation: fusion of the capabilities of EC optimization and human evaluation, Proc. IEEE. 89 (2001) 1275–1296. https://doi.org/10.1109/5.949485.

[32] W. Banzhaf, Interactive evolution in "Handbook of Evolutionary Computation" (1997) by Thomas B, in: Handb. Evol. Comput., 1997: pp. 1–6. https://doi.org/10.1887/0750308958.

[33] C.R. Bonham, I.C. Parmee, An Investigation of Exploration and Exploitation Within Cluster Oriented Genetic Algorithms (COGAs), Proc. 1st Genet. Evol. Comput. Conf., GECCO'99. (1999) 1491–1497.

[34] C.R. Bonham, I.C. Parmee, Developments of the cluster oriented genetic algorithm (COGA), Eng. Optim. 36 (2004) 249–279. https://doi.org/10.1080/03052150410001650160.

[35] T. Terano, Y. Ishino, Data Analyses Using Simulated Breeding and Inductive Learning Methods Takao TERANO and Yoko ISHINO Graduate School of Systems Management , The University of Tsukuba , Tokyo address : 3-29-1 Otsuka , Bunkyo-ku , Tokyo 112 , Japan, (1995).

[36] B. Addis, Building: 3000 years of design engineering and construction, Phaidon, London, New York, 2007.

[37] M. O'Neill, J. Mcdermott, J.M. Swafford, J. Byrne, E. Hemberg, A. Brabazon, Evolutionary design using grammatical evolution and shape grammars : designing a shelter Elizabeth Shotton , Ciaran McNally Martin Hemberg, 3 (2010) 1–23.

[38] J. Byrne, M. Fenton, E. Hemberg, J. McDermott, M. O'Neill, E. Shotton, C. Nally, Combining structural analysis and multi-objective criteria for evolutionary architectural design, Lect. Notes Comput. Sci. (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics). 6625 LNCS (2011) 204–213. https://doi.org/10.1007/978-3-642-20520-0_21.

[39] D.R. Brophy, Comparing the attributes, activities, and performance of divergent, convergent, and combination thinkers, Creat. Res. J. 13 (2001) 439–455. https://doi.org/10.1207/s15326934crj1334_20.

[40] J. Kelly, P.Y. Papalambros, C.M. Seifert, Interactive genetic algorithms for use as creativity enhancement tools, AAAI Spring Symp. - Tech. Rep. SS-08-03 (2008) 34–39.

[41] R. Vierlinger, Multi Objective Design Interface, 2013. https://doi.org/10.13140/RG.2.1.3401.0324.

[42] C. Mueller, Computational Exploration of the Structural Design Space, 2014. http://dspace.mit.edu/handle/1721.1/91293.

[43] J. Harding, Evolving Parametric Models using Genetic Programming, in: ECAADe 2016, 2016: pp. 423–432.

[44] K. Martini, Harmony Search Method for Multimodal Size, Shape, and Topology Optimization of Structural Frameworks, J. Struct. Eng. 137 (2011) 1332–1339. https://doi.org/10.1061/(asce)st.1943-541x.0000378.

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

[46] L. Lachauer, P. Block, Interactive equilibrium modelling, Int. J. Sp. Struct. 29 (2014) 25–38. https://doi.org/10.1260/0266-3511.29.1.25.

[47] T. van Mele, L. Lachauer, M. Rippmann, P. Block, Geometry-based understanding of structures, J. Int. Assoc. Shell Spat. Struct. 53 (2012) 285–295.

[48] C. Fivet, D. Zastavni, A fully geometric approach for interactive constraint-based structural equilibrium design, CAD Comput. Aided Des. 61 (2015) 42–57. https://doi.org/10.1016/j.cad.2014.04.001.

[49] C. Fivet, Constraint Based Graphic Statics, PhD thesis, KU Leuven, 2013.

[50] C. Hartz, A. Mazurek, M. Miki, T. Zegard, T. Mitchell, W. Baker, The application of 2D and 3D graphic statics in design, J. Int. Assoc. Shell Spat. Struct. 59 (2018) 235–242. https://doi.org/10.20898/j.iass.2018.198.032.

[51] P.O. Ohlbrock, P. D'Acunto, A Computer-Aided Approach to Equilibrium Design Based on Graphic Statics and Combinatorial Variations, CAD Comput. Aided Des. 121 (2020) 102802. https://doi.org/10.1016/j.cad.2019.102802.

[52] K. Saldana Ochoa, P.O. Ohlbrock, P. D'Acunto, V. Moosavi, Beyond typologies, beyond optimization: Exploring novel structural forms at the interface of human and machine intelligence, Int. J. Archit. Comput. (2020). https://doi.org/10.1177/1478077120943062.

[53] M. P. Bendsøe, O. Sigmund, Topology Optimization, Springer, Berlin, Heidelberg (2004) https://doi.org/10.1007/978-3-662-05086-6

[54] Y.M. Xie, G.P. Steven, Evolutionary structural optimization, London , Springer (1997).

[55] X. Huang, Y.M. Xie, Evolutionary Topology Optimization of Continuum Structures: Methods and Applications, Chichester, West Sussex, UK: Wiley (2010).

[56] M. P. Bendsøe, N. Kikuchi, Generating optimal topologies in structural design using a homogenization method, Computer Methods in Applied Mechanics and Engineering, vol.71, Issue 2, pp 197-224 (1988), https://doi.org/10.1016/0045-7825(88)90086-2.

[57] P. Park, M. Gilbert, A. Tyas, O. Popovic-Larsen Potential use of structural layout optimization at the conceptual design stage, Int J Archit Comput. vol.10 (1): 13–32, (2012).

[58] L. He, Li Q., Gilbert M., Shepherd, C. Rankine, T. Pritchard, R. Vincenzo, Optimization -driven conceptual design of truss structures in a parametric modellingenvironment, Structures, vol 37: 469-482 (2022). https://doi.org/10.1016/j.istruc.2021.12.048.

[59] N. Chomsky, Syntatic structures, Martino Fine Books (1957).

[60] G. Stiny, J. Gips, Shape grammars and the generative specification of painting and sculpture, Inf. Process. 71 Proc. IFIP Congr. 1971. Vol. 2. 71 (1972) 1460–1465.

[61] A. McKay, S. Chase, K. Shea, H.H. Chau, Spatial grammar implementation: From theory to useable software, Artif. Intell. Eng. Des. Anal. Manuf. AIEDAM. 26 (2012) 143–159. https://doi.org/10.1017/S0890060412000042.

[62] P. Pauwels, T. Strobbe, S. Eloy, R. De Meyer, Shape grammars for architectural design: The need for reframing, Commun. Comput. Inf. Sci. 527 (2015) 507–526. https://doi.org/10.1007/978-3-662-47386-3_28.

[63] J.P. Duarte, A discursive grammar for customizing mass housing: The case of Siza's houses at Malagueira, Autom. Constr. 14 (2005) 265–275. https://doi.org/10.1016/j.autcon.2004.07.013.

[64] A. Economou, M. Swarts, Performing Palladio, Int. J. Archit. Comput. 4 (2006) 47–61. https://doi.org/10.1260/147807706778658865.

[65] A. Chakrabarti, K. Shea, R. Stone, J. Cagan, M. Campbell, N.V. Hernandez, K.L. Wood, Computer-based design synthesis research: An overview, J. Comput. Inf. Sci. Eng. 11 (2011). https://doi.org/10.1115/1.3593409.

[66] G. Stiny, Shape: Talking about Seeing and Doing. Cambridge, MA: MIT Press (2006).

[67] J.N. Beirão, J.P. Duarte, R. Stouffs, Creating Specific Grammars with Generic Grammars: Towards Flexible Urban Design, Nexus Netw. J. 13 (2011) 73–111. https://doi.org/10.1007/s00004-011-0059-3.

[68] R. Stouffs, P. Janssen, A Rule-Based Generative Analysis Approach for Urban Planning, in: J.-H. Lee (Ed.), Morphol. Anal. Cult. DNA Tools Decod. Cult. Forms, Springer Singapore, Singapore, 2017: pp. 125–136. https://doi.org/10.1007/978-981-10-2329-3_10.

[69] G. Stiny, W. Mitchell, The Palladian grammar, Environ. Plan. B Plan. Des. 5 (1978) 5–18. https://doi.org/10.1068/b050005.

[70] S. Carta, Self-Organizing Floor Plans, Harvard Data Sci. Rev. (2021) 1–39. https://doi.org/10.1162/99608f92.e5f9a0c7.

[71] M. Ruiz-Montiel, J. Boned, J. Gavilanes, E. Jiménez, L. Mandow, J.L. Pérez-De-La-Cruz, Design with shape grammars and reinforcement learning, Adv. Eng. Informatics. 27 (2013) 230–245. https://doi.org/10.1016/j.aei.2012.12.004.

[72] H. Koning, J. Eizenberg, The language of the Prairie: Frank Loyd Wright's Prairie houses, Environ. Plan. B Plan. Des. 8 (1981) 295–323. https://doi.org/10.1068/b080295.

[73] W. Mitchell, Functional Grammars: An introduction, Real. Virtual Real. Proc. ACADIA ’91. (1991) 167–176.

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

[75] J. Lee, C. Mueller, C. Fivet, Automatic generation of diverse equilibrium structures through shape grammars and graphic statics, Int. J. Sp. Struct. 31 (2016) 147–164. https://doi.org/10.1177/0266351116660798.

[76] F. Cascone, D. Faiella, V. Tomei, E. Mele, A structural grammar approach for the generative design of diagrid-like structures, Buildings. 11 (2021) 1–21. https://doi.org/10.3390/buildings11030090.

[77] K. Shea, J. Cagan, S.J. Fenves, A shape annealing approach to optimal truss design with dynamic grouping of members, J. Mech. Des. Trans. ASME. 119 (1997) 388–394. https://doi.org/10.1115/1.2826360.

[78] K. Shea, J. Cagan, Languages and semantics of grammatical discrete structures, Artif. Intell. Eng. Des. Anal. Manuf. AIEDAM. 13 (1999) 241–251. https://doi.org/10.1017/S0890060499134012.

[79] K. Shea, R. Aish, M. Gourtovaia, Towards integrated performance-driven generative design tools, Automation in Construction, vol 14, pp. 253-264 (2005).

[80] M. O’Neill, J. Mc-Dermott, J.M. Swafford, J. Byrne, E. Hemberg, A. Brabazon, E. Shotton, C. McNally, M. Hemberg Evolutionary Design using Grammatical Evolution and Shape Grammars: Designing a Shelter’, Int. J. Design Engineering, Vol. 3, No. 1, pp. 4–24 (2010)

[81] K. Terzidis, Algorithmic architecture, Routledge, 2006.

[82] L. Moretti, Ricerca Matematica in Architettura e Urbanisticâ, Moebius IV, Republished Federico Bucci Marco Mulazzani. 2000. Luigi Moretti Work. Writings, New York Princet. Archit. Press. (1971) 30–53.

[83] R. Aish, R. Woodbury, Multi-level interaction in parametric design, Lect. Notes Comput. Sci. 3638 (2005) 151–162. https://doi.org/10.1007/11536482_13.

[84] R. Woodbury, Elements of parametric design, 1st editio, Routledge, 2010.

[85] L. Zimmermann, T. Chen, K. Shea, 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. AIEDAM. 32 (2018) 189–199. https://doi.org/10.1017/S0890060417000324.

[86] S. Huerta, Structural design in the work of gaudí, Archit. Sci. Rev. 49 (2006) 324–339. https://doi.org/10.3763/asre.2006.4943.

[87] D. Zastavni, The structural design of Maillart’s Chiasso Shed (1924): A graphic procedure, Struct. Eng. Int. J. Int. Assoc. Bridg. Struct. Eng. 18 (2008) 247–252. https://doi.org/10.2749/101686608785096496.

[88] E. Allen, W. Zalewski, Form and forces: Designing efficient, expressive structures, John Wiley & Sons, Inc., Hoboken, New Jersey, 2009.

[89] J. Conzett, M. Mostafavi, Structure as space: Engineering and architecture in the works of Jurg Conzett, AA Publications, 2006.

[90] M.L. Benedikt, Computational models of space: Isovists and isovist fields, Comput. Graph. Image Process. 11 (1979) 49–72. https://doi.org/10.1016/0146-664X(79)90076-5.

[91] I. Mirtsopoulos, C. Fivet, Grammar-based generation of bar networks in static equilibrium with bounded bar lengths, Proceedings of the International Association of Shell and Spatial Structures IASS 2020-21, 23-27 August 2021, Surrey, UK (2021).

[92] W.F. Baker, L.L. Berghini, A. Mazurek, J. Carion, A. Beghini, Maxwell's reciprocal diagrams and discrete Michell frames, Struct Multidisc Optim, 48: 267-277, (2013) .

[93] I. Mirtsopoulos, C. Fivet, Exploratory conceptual structural design through policy-based generation of equilibrium, CAD Comput. Aided Des. (2022) (under review).

[94] C. Cubukcuoglu, B. Ekici, M.F. Tasgetiren, S. Sariyildiz, Optimus: Self-adaptive differential evolution with ensemble of mutation strategies for grasshopper algorithmic modeling, Algorithms. 12 (2019). https://doi.org/10.3390/a12070141.

[95] M. Makki, M. Showkatbakhsh, A. Tabony, M. Weinstock, Evolutionary algorithms for generating urban morphology: Variations and multiple objectives, Int. J. Archit. Comput. 17 (2019) 5–35. https://doi.org/10.1177/1478077118777236.

[96] N. Brown, J. Ochsendorf, C. Mueller, J. de Oliveira, Early-stage integration of architectural and structural performance in a parametric multi-objective design tool, Struct. Archit. - Proc. 3rd Int. Conf. Struct. Archit. ICSA 2016. (2016) 1103–1111. https://doi.org/10.1201/b20891-152.

[97] S. Tseranidis, N. Brown, C. Mueller, Data-driven approximation algorithms for rapid performance evaluation and optimization of civil structures, Autom. Constr. 72 (2016) 279–293. https://doi.org/10.1016/j.autcon.2016.02.002.

[98] R.A. Danhaive, C. Mueller, Combining parametric modeling and interactive optimization for high- performance and creative structural design Combining parametric modeling and interactive optimization for high-performance and creative structural design, in: IASS, 2015.

[99] W.M. Spears, The Role of Mutation and Recombination in Evolutionary Algorithms, A Disertation Submitt. Partial Fulfillment Requir. Degree Od Dr. Philosofhy Georg. Mason Univ. (1998) 258.

[100] M. Koechlin, Pylône de 300 mètres de hauteur pour la ville de Paris 1889, (1884). https://doi.org/10.7891/e-manuscripta-5218.

[101] A. Muttoni, M.F. Ruiz, F. Niketic, Design versus assessment of concrete structures using stress fields and strut-and-tie models, ACI Struct. J. 112 (2015) 605–615. https://doi.org/10.14359/51687710.

[102] J J Orr, M. Cooke, T.J. Ibell, C. Smith, N. Watson, Design for zero, IStructE Ltd, 2021. https://doi.org/10.5694/j.1326-5377.1914.tb22300.x.

[103] M. Akbarzadeh, 3D Graphical Statics Using Reciprocal Polyhedral Diagrams, ETH Zurich, 2016. https://doi.org/10.3929/ethz-a-010782581.

Appendix A Rules

Appendix A.1 Entropy rate

-1	Convergence
0	Stagnation
1	Divergence

Appendix A.2 Force(s) selection rule

-1	CustomUserSelection
0	RandomAny
1	RandomMonomial
2	RandomBinomialAny
3	RandomBinomialInbetween
4	RandomBinomialPeripheral
5	RandomTrinomialAny
6	RandomTrinomialInbetween
7	RandomTrinomialCentral
8	RandomTrinomialPeripheral
9	ProximityToUserSelectedPoint
10	MinMagnitudeMonomial
11	MinMagnitudeBinomialAny
12	MinMagnitudeBinomialInbetween
13	MinMagnitudeBinomialPeripheral
14	MinMagnitudeTrinomialAny
15	MinMagnitudeTrinomialInbetween
16	MinMagnitudeTrinomialCentral
17	MinMagnitudeTrinomialPeripheral
18	MaxMagnitudeMonomial
19	MaxMagnitudeBinomialAny
20	MaxMagnitudeBinomialInbetween
21	MaxMagnitudeBinomialPeripheral
22	MaxMagnitudeTrinomialAny
23	MaxMagnitudeTrinomialInbetween
24	MaxMagnitudeTrinomialCentral
25	MaxMagnitudeTrinomialPeripheral
26	OldestMonomial
27	OldestBinomialAny
28	OldestBinomialInbetween
29	OldestBinomialPeripheral
30	OldestTrinomialAny
31	OldestTrinomialInbetween
32	OldestTrinomialCentral
33	OldestTrinomialPeripheral
34	NewestMonomial
35	NewestBinomialAny
36	NewestBinomialInbetween

37 NewestBinomialPeripheral
38 NewestTrinomialAny
39 NewestTrinomialInbetween
40 NewestTrinomialCentral
41 NewestTrinomialPeripheral
42 ClosestBinomialAny
43 ClosestBinomialInbetween
44 ClosestBinomialPeripheral
45 ClosestTrinomialAny
46 ClosestTrinomialInbetween
47 ClosestTrinomialCentral
48 ClosestTrinomialPeripheral
49 FurthestBinomialAny
50 FurthestBinomialInbetween
51 FurthestBinomialPeripheral
52 FurthestTrinomialAny
53 FurthestTrinomialInbetween
54 FurthestTrinomialCentral
55 FurthestTrinomialPeripheral
56 ClosestToDomainCentroidMonomial
57 ClosestToDomainCentroidBinomialAny
58 ClosestToDomainCentroidBinomialInbetween
59 ClosestToDomainCentroidBinomialPeripheral
60 ClosestToDomainCentroidTrinomialAny
61 ClosestToDomainCentroidTrinomialInbetween
62 ClosestToDomainCentroidTrinomialCentral
63 ClosestToDomainCentroidTrinomialPeripheral
64 FurthestToDomainCentroidMonomial
65 FurthestToDomainCentroidBinomialAny
66 FurthestToDomainCentroidBinomialInbetween
67 FurthestToDomainCentroidBinomialPeripheral
68 FurthestToDomainCentroidTrinomialAny
69 FurthestToDomainCentroidTrinomialInbetween
70 FurthestToDomainCentroidTrinomialCentral
71 FurthestToDomainCentroidTrinomialPeripheral
72 ClosestToPtCloudMonomial
73 ClosestToPtCloudBinomialAny
74 ClosestToPtCloudBinomialInbetween
75 ClosestToPtCloudBinomialPeripheral
76 ClosestToPtCloudTrinomialAny
77 ClosestToPtCloudTrinomialInbetween
78 ClosestToPtCloudTrinomialCentral
79 ClosestToPtCloudTrinomialPeripheral
80 FurthestToPtCloudMonomial
81 FurthestToPtCloudBinomialAny
82 FurthestToPtCloudBinomialInbetween
83 FurthestToPtCloudBinomialPeripheral
84 FurthestToPtCloudTrinomialAny

85 FurthestToPtCloudTrinomialInbetween
86 FurthestToPtCloudTrinomialCentral
87 FurthestToPtCloudTrinomialPeripheral
88 ClosestToBoundaryMonomial
89 ClosestToBoundaryBinomialAny
90 ClosestToBoundaryBinomialInbetween
91 ClosestToBoundaryBinomialPeripheral
92 ClosestToBoundaryTrinomialAny
93 ClosestToBoundaryTrinomialInbetween
94 ClosestToBoundaryTrinomialCentral
95 ClosestToBoundaryTrinomialPeripheral
96 FurthestToBoundaryMonomial
97 FurthestToBoundaryBinomialAny
98 FurthestToBoundaryBinomialInbetween
99 FurthestToBoundaryBinomialPeripheral
100 FurthestToBoundaryTrinomialAny
101 FurthestToBoundaryTrinomialInbetween
102 FurthestToBoundaryTrinomialCentral
103 FurthestToBoundaryTrinomialPeripheral
104 LeftmostMonomial_X_Axis
105 LeftmostBinomialAny_X_Axis
106 LeftmostBinomialInbetween_X_Axis
107 LeftmostBinomialPeripheral_X_Axis
108 LeftmostTrinomialAny_X_Axis
109 LeftmostTrinomialInbetween_X_Axis
110 LeftmostTrinomialCentral_X_Axis
111 LeftmostTrinomialPeripheral_X_Axis
112 RightmostMonomial_X_Axis
113 RightmostBinomialAny_X_Axis
114 RightmostBinomialInbetween_X_Axis
115 RightmostBinomialPeripheral_X_Axis
116 RightmostTrinomialAny_X_Axis
117 RightmostTrinomialInbetween_X_Axis
118 RightmostTrinomialCentral_X_Axis
119 RightmostTrinomialPeripheral_X_Axis
120 BackmostMonomial_Y_Axis
121 BackmostBinomialAny_Y_Axis
122 BackmostBinomialInbetween_Y_Axis
123 BackmostBinomialPeripheral_Y_Axis
124 BackmostTrinomialAny_Y_Axis
125 BackmostTrinomialInbetween_Y_Axis
126 BackmostTrinomialCentral_Y_Axis
127 BackmostTrinomialPeripheral_Y_Axis
128 FrontmostMonomial_Y_Axis
129 FrontmostBinomialAny_Y_Axis
130 FrontmostBinomialInbetween_Y_Axis
131 FrontmostBinomialPeripheral_Y_Axis
132 FrontmostTrinomialAny_Y_Axis

133 FrontmostTrinomialInbetween_Y_Axis
134 FrontmostTrinomialCentral_Y_Axis
135 FrontmostTrinomialPeripheral_Y_Axis
136 BottommostMonomial_Z_Axis
137 BottommostBinomialAny_Z_Axis
138 BottommostBinomialInbetween_Z_Axis
139 BottommostBinomialPeripheral_Z_Axis
140 BottommostTrinomialAny_Z_Axis
141 BottommostTrinomialInbetween_Z_Axis
142 BottommostTrinomialCentral_Z_Axis
143 BottommostTrinomialPeripheral_Z_Axis
144 TopmostMonomial_Z_Axis
145 TopmostBinomialAny_Z_Axis
146 TopmostBinomialInbetween_Z_Axis
147 TopmostBinomialPeripheral_Z_Axis
148 TopmostTrinomialAny_Z_Axis
149 TopmostTrinomialInbetween_Z_Axis
150 TopmostTrinomialCentral_Z_Axis
151 TopmostTrinomialPeripheral_Z_Axis
152 LeftBackmostMonomial_XY_Plane
153 LeftBackmostBinomialAny_XY_Plane
154 LeftBackmostBinomialInbetween_XY_Plane
155 LeftBackmostBinomialPeripheral_XY_Plane
156 LeftBackmostTrinomialAny_XY_Plane
157 LeftBackmostTrinomialInbetween_XY_Plane
158 LeftBackmostTrinomialCentral_XY_Plane
159 LeftBackmostTrinomialPeripheral_XY_Plane
160 LeftFrontmostMonomial_XY_Plane
161 LeftFrontmostBinomialAny_XY_Plane
162 LeftFrontmostBinomialInbetween_XY_Plane
163 LeftFrontmostBinomialPeripheral_XY_Plane
164 LeftFrontmostTrinomialAny_XY_Plane
165 LeftFrontmostTrinomialInbetween_XY_Plane
166 LeftFrontmostTrinomialCentral_XY_Plane
167 LeftFrontmostTrinomialPeripheral_XY_Plane
168 RightBackmostMonomial_XY_Plane
169 RightBackmostBinomialAny_XY_Plane
170 RightBackmostBinomialInbetween_XY_Plane
171 RightBackmostBinomialPeripheral_XY_Plane
172 RightBackmostTrinomialAny_XY_Plane
173 RightBackmostTrinomialInbetween_XY_Plane
174 RightBackmostTrinomialCentral_XY_Plane
175 RightBackmostTrinomialPeripheral_XY_Plane
176 RightFrontmostMonomial_XY_Plane
177 RightFrontmostBinomialAny_XY_Plane
178 RightFrontmostBinomialInbetween_XY_Plane
179 RightFrontmostBinomialPeripheral_XY_Plane
180 RightFrontmostTrinomialAny_XY_Plane

181 RightFrontmostTrinomialInbetween_XY_Plane
182 RightFrontmostTrinomialCentral_XY_Plane
183 RightFrontmostTrinomialePeripheral_XY_Plane
184 LeftBottommostMonomial_XZ_Plane
185 LeftBottommostBinomialAny_XZ_Plane
186 LeftBottommostBinomialInbetween_XZ_Plane
187 LeftBottommostBinomialPeripheral_XZ_Plane
188 LeftBottommostTrinomialAny_XZ_Plane
189 LeftBottommostTrinomialInbetween_XZ_Plane
190 LeftBottommostTrinomialCentral_XZ_Plane
191 LeftBottommostTrinomialPeripheral_XZ_Plane
192 LeftTopmostMonomial_XZ_Plane
193 LeftTopmostBinomialAny_XZ_Plane
194 LeftTopmostBinomialInbetween_XZ_Plane
195 LeftTopmostBinomialPeripheral_XZ_Plane
196 LeftTopmostTrinomialAny_XZ_Plane
197 LeftTopmostTrinomialInbetween_XZ_Plane
198 LeftTopmostTrinomialCentral_XZ_Plane
199 LeftTopmostTrinomialPeripheral_XZ_Plane
200 RightBottommostMonomial_XZ_Plane
201 RightBottommostBinomialAny_XZ_Plane
202 RightBottommostBinomialInbetween_XZ_Plane
203 RightBottommostBinomialPeripheral_XZ_Plane
204 RightBottommostTrinomialAny_XZ_Plane
205 RightBottommostTrinomialInbetween_XZ_Plane
206 RightBottommostTrinomialCentral_XZ_Plane
207 RightBottommostTrinomialPeripheral_XZ_Plane
208 RightTopmostMonomial_XZ_Plane
209 RightTopmostBinomialAny_XZ_Plane
210 RightTopmostBinomialInbetween_XZ_Plane
211 RightTopmostBinomialPeripheral_XZ_Plane
212 RightTopmostTrinomialAny_XZ_Plane
213 RightTopmostTrinomialInbetween_XZ_Plane
214 RightTopmostTrinomialCentral_XZ_Plane
215 RightTopmostTrinomialPeripheral_XZ_Plane
216 BackBottommostMonomial_YZ_Plane
217 BackBottommostBinomialAny_YZ_Plane
218 BackBottommostBinomialInbetween_YZ_Plane
219 BackBottommostBinomialPeripheral_YZ_Plane
220 BackBottommostTrinomialAny_YZ_Plane
221 BackBottommostTrinomialInbetween_YZ_Plane
222 BackBottommostTrinomialCentral_YZ_Plane
223 BackBottommostTrinomialPeripheral_YZ_Plane
224 BackTopmostMonomial_YZ_Plane
225 BackTopmostBinomialAny_YZ_Plane
226 BackTopmostBinomialInbetween_YZ_Plane
227 BackTopmostBinomialPeripheral_YZ_Plane
228 BackTopmostTrinomialAny_YZ_Plane

229 BackTopmostTrinomialInbetween_YZ_Plane
230 BackTopmostTrinomialCentral_YZ_Plane
231 BackTopmostTrinomialPeripheral_YZ_Plane
232 FrontBottommostMonomial_YZ_Plane
233 FrontBottommostBinomialAny_YZ_Plane
234 FrontBottommostBinomialInbetween_YZ_Plane
235 FrontBottommostBinomialPeripheral_YZ_Plane
236 FrontBottommostTrinomialAny_YZ_Plane
237 FrontBottommostTrinomialInbetween_YZ_Plane
238 FrontBottommostTrinomialCentral_YZ_Plane
239 FrontBottommostTrinomialPeripheral_YZ_Plane
240 FrontTopmostMonomial_YZ_Plane
241 FrontTopmostBinomialAny_YZ_Plane
242 FrontTopmostBinomialInbetween_YZ_Plane
243 FrontTopmostBinomialPeripheral_YZ_Plane
244 FrontTopmostTrinomialAny_YZ_Plane
245 FrontTopmostTrinomialInbetween_YZ_Plane
246 FrontTopmostTrinomialCentral_YZ_Plane
247 FrontTopmostTrinomialPeripheral_YZ_Plane
248 LeftBackBottommostMonomial
249 LeftBackBottommostBinomialAny
250 LeftBackBottommostBinomialInbetween
251 LeftBackBottommostBinomialPeripheral
252 LeftBackBottommostTrinomialAny
253 LeftBackBottommostTrinomialInbetween
254 LeftBackBottommostTrinomialCentral
255 LeftBackBottommostTrinomialPeripheral
256 LeftFrontBottommostMonomial
257 LeftFrontBottommostBinomialAny
258 LeftFrontBottommostBinomialInbetween
259 LeftFrontBottommostBinomialPeripheral
260 LeftFrontBottommostTrinomialAny
261 LeftFrontBottommostTrinomialInbetween
262 LeftFrontBottommostTrinomialCentral
263 LeftFrontBottommostTrinomialPeripheral
264 RightBackBottommostMonomial
265 RightBackBottommostBinomialAny
266 RightBackBottommostBinomialInbetween
267 RightBackBottommostBinomialPeripheral
268 RightBackBottommostTrinomialAny
269 RightBackBottommostTrinomialInbetween
270 RightBackBottommostTrinomialCentral
271 RightBackBottommostTrinomialPeripheral
272 RightFrontBottommostMonomial
273 RightFrontBottommostBinomialAny
274 RightFrontBottommostBinomialInbetween
275 RightFrontBottommostBinomialPeripheral
276 RightFrontBottommostTrinomialAny

277	RightFrontBottommostTrinomialInbetween
278	RightFrontBottommostTrinomialCentral
279	RightFrontBottommostTrinomialPeripheral

Appendix A.3 Node placement rule

0	Random
1	UserSelectedParam
2	UserSelectedPt
3	ConstrainedBetweenXbounds_LeftToRight
4	ConstrainedBetweenXbounds_RightToLeft
5	ConstrainedBetweenYBounds_BackToFront
6	ConstrainedBetweenYBounds_FrontToBack
7	ConstrainedBetweenZbounds_BottomToTop
8	ConstrainedBetweenZbounds_TopToBottom
9	ConstrainedBetweenXYbounds_LeftBackToRightFront
10	ConstrainedBetweenXYbounds_LeftFrontToRightBack
11	ConstrainedBetweenXYbounds_RightBackToLeftFront
12	ConstrainedBetweenXYbounds_RightFrontToLeftBack
13	ConstrainedBetweenXZbounds_LeftBottomToRightTop
14	ConstrainedBetweenXZbounds_LeftTopToRightBottom
15	ConstrainedBetweenXZbounds_RightBottomToLeftTop
16	ConstrainedBetweenXZbounds_RightTopToLeftBottom
17	ConstrainedBetweenYZbounds_BackBottomToFrontTop
18	ConstrainedBetweenYZbounds_BackTopToFrontBottom
19	ConstrainedBetweenYZbounds_FrontBottomToBackTop
20	ConstrainedBetweenYZbounds_FrontTopToBackBottom
21	ConstrainedBetweenXYZbounds_LeftBackBottomToRightFrontTop
22	ConstrainedBetweenXYZbounds_LeftFrontBottomToRightBackTop
23	ConstrainedBetweenXYZbounds_RightBackBottomToLeftFrontTop
24	ConstrainedBetweenXYZbounds_RightFrontBottomToLeftBackTop
25	ConstrainedBetweenXYZbounds_LeftBackTopToRightFrontBottom
26	ConstrainedBetweenXYZbounds_LeftFrontTopToRightBackBottom
27	ConstrainedBetweenXYZbounds_RightBackTopToLeftFrontBottom
28	ConstrainedBetweenXYZbounds_RightFrontTopToLeftBackBottom

Appendix A.4 Force indeterminacies

0	Random
1	UserDefinedMagnitude
2	UserDefinedUtilization

Appendix B Illustrated behavior / settings

Appendix B.1 Entropy rate

INPUT PARAMETERS AND THEIR VALUES - FIG. 24

desired entropy rate	**-1**	**0**	**1**
force(s) selection rule	106	106	106
flip forces	false	false	false
node placement rule	1	1	1
t_param	0.611	0.611	0.611
constrained bar lengths	false	false	false
force indeterminacies rule	0	0	0
seed number	544	544	544
absolute value	5	5	5

Appendix B.2 Force selection

INPUT PARAMETERS AND THEIR VALUES - FIG. 25

desired entropy rate	-1	-1	-1	-1	-1	-1
force(s) selection rule	106	138	114	3	3	3
flip forces	false	false	false	false	false	false
seed number	-	-	-	340	781	223
node placement rule	1	1	1	1	1	1
t_param	0.764	0.764	0.764	0.764	0.764	0.764
constrained bar lengths	false	false	false	false	false	false
force indeterminacies rule	0	0	0	0	0	0
seed number	544	544	544	544	544	544
absolute value	5	5	5	5	5	5

INPUT PARAMETERS AND THEIR VALUES - FIG. 26

desired entropy rate	-1	-1	-1	-1	-1	-1
force(s) selection rule	28	36	43	50	12	20
flip forces	false	false	false	false	false	false
seed number	-	-	-	-	-	-
node placement rule	1	1	1	1	1	1
t_param	0.764	0.764	0.764	0.764	0.764	0.764
constrained bar lengths	false	false	false	false	false	false
force indeterminacies rule	0	0	0	0	0	0
seed number	544	544	544	544	544	544
absolute value	5	5	5	5	5	5

Appendix B.3 Node placement

INPUT PARAMETERS AND THEIR VALUES - FIG. 27				
desired entropy rate	-1	-1	-1	-1
force(s) selection rule	106	106	106	106
flip forces	false	false	false	false
node placement rule	0	0	1	1
seed number	689	590	-	-
t_param	-	-	0.945	0.285
constrained bar lengths	false	false	false	false
force indeterminacies rule	0	0	0	0
seed number	783	783	783	783
absolute value	5	5	5	5

Appendix B.4 Force indeterminacies

INPUT PARAMETERS AND THEIR VALUES - FIG. 28						
desired entropy rate	-1	-1	-1	-1	-1	-1
force(s) selection rule	106	106	106	106	106	106
flip forces	false	false	false	False	false	false
node placement rule	1	1	1	1	1	1
t_param	0.764	0.764	0.764	0.764	0.764	0.764
constrained bar lengths	false	false	false	False	false	false
force indeterminacies rule	0	0	0	0	0	0
seed number	783	692	783	692	-	-
absolute value	10	10	20	20	-	-
N1	-	-	-	-	-7.230	1.796
N2	-	-	-	-	3.155	1.199
N3	-	-	-	-	-1.960	-1.960

Appendix B.5 Design domain

INPUT PARAMETERS AND THEIR VALUES - FIG. 29				
desired entropy rate	-1	-1	-1	-1
force(s) selection rule	106	106	106	106
flip forces	false	false	false	false
node placement rule	1	1	1	1
t_param	0.764	0.764	0.764	0.764
constrained bar lengths	false	false	false	false
force indeterminacies rule	0	0	0	0
seed number	783	783	783	783
absolute value	5	5	5	5

Appendix B.6 Constrains addition

INPUT PARAMETERS AND THEIR VALUES - FIG. 30				
desired entropy rate	-1	-1	-1	-1
force(s) selection rule	106	106	106	106
flip forces	false	false	false	false
node placement rule	1	1	1	1
t_param	0.764	0.764	0.764	0.764
constrained bar lengths	true	true	true	true
minimum length	15	10	10	5
maximum length	20	20	15	20
force indeterminacies rule	0	0	0	0
seed number	225	225	225	225
absolute value	5	5	5	5

Appendix C Application studies

INPUT PARAMETERS AND THEIR VALUES – FIG. 14

step	force(s) selection rule	node placement rule	force indeterminacies rule	seed number	desired entropy rate	applied entropy rate
1	138	2	0.26, 0.39, 0.46	-	-1	0
2	-1	2	-0.52, 0.39, 0.46	-	0	0
3	-1	2	0.32, 0.39, 0.46	-	0	0
4	-1	2	-0.08, 0.39, 0.46	-	0	0
5	-1	2	-0.08, 0.39, 0.46	-	0	0
6	-1	2	0.07, 0.39, 0.46	-	-1	-1
7	-1	2	-0.85, -0.80, 0.56	-	1	1
8	-1	2	-0.95, -0.84, 0.56	-	-1	-1
9	-1	2	-0.95, -084, 0.56	-	-1	-1
10	-1	2	-0.95, -0.84, 0.56	-	0	0
11	-1	2	-0.72, -0.84, 0.56	-	0	0
12	-1	2	-0.72, -0.84, 0.56	-	-1	-1
13	-1	2	-0.72, -0.84, 0.56	-	-1	-1

INPUT PARAMETERS AND THEIR VALUES – FIG. 42

step	force(s) selection rule	node placement rule	applied entropy rate
1	-1	2	1
2	-1	2	1
3	-1	2	0
4	-1	2	-1
5	-1	2	-1
6	-1	2	-1
7	-1	2	-1
8	-1	2	-1
9	-1	2	1
10	-1	2	-1
11	-1	2	-1
12	-1	2	1
13	-1	2	-1
14	-1	2	1
15	-1	2	-1
16	-1	2	-1

INPUT PARAMETERS AND THEIR VALUES – FIG. 43

step	force(s) selection rule	node placement rule	applied entropy rate
1	-1	2	1
2	-1	2	1
3	-1	2	-1
4	-1	2	-1
5	-1	2	-1
6	-1	2	1
7	-1	2	-1
8	-1	2	1
9	-1	2	-1
10	-1	2	-1
11	-1	2	1
12	-1	2	-1
13	-1	2	-1
14	-1	2	-1
15	-1	2	-1
16	-1	2	-1

INPUT PARAMETERS AND THEIR VALUES – FIG. 44

step	force(s) selection rule	node placement rule	force indetermi-nacies rule	seed number	desired entropy rate	applied entropy rate
1	106	3	0	49	0	0
2	106	3	0	0	-1	-1
3	106	3	0	85	0	0
4	106	3	0	0	-1	-1
5	106	3	0	98	0	0
6	106	3	0	0	-1	-1
7	106	3	0	68	0	0
8	106	3	0	119	0	0
9	106	3	0	119	0	0
10	106	3	0	0	-1	-1
11	106	3	0	285	0	0
12	106	3	0	0	-1	-1
13	106	3	0	620	0	0
14	106	3	0	0	-1	-1
15	106	3	0	295	0	0
16	106	3	0	0	-1	-1
17	106	3	0	0	-1	-1

INPUT PARAMETERS AND THEIR VALUES – FIG. 45

step	force(s) selection rule	node placement rule	force indeterminacies rule	seed number	desired entropy rate	applied entropy rate
1	186	0	0	728	0	0
2	186	0	0	0	-1	-1
3	202	0	0	728	0	0
4	202	0	0	0	-1	-1
5	186	0	0	883	0	0
6	202	0	0	883	0	0
7	186	0	0	114	0	0
8	202	0	0	114	0	0
9	186	0	0	679	0	0
10	202	0	0	679	0	0
11	186	0	0	119	0	0
12	202	0	0	119	0	0
13	186	0	0	23	0	0
14	202	0	0	23	0	0
15	186	0	0	241	0	0
16	202	0	0	241	0	0
17	186	0	0	441	0	0
18	202	0	0	441	0	0
19	106	0	0	121	0	0
20	114	0	0	121	0	0
21	106	0	0	0	-1	-1
22	114	0	0	0	-1	-1
23	106	0	0	80	0	0
24	114	0	0	80	0	0
25	106	0	0	0	-1	-1
26	114	0	0	0	-1	-1
27	106	0	0	206	0	0
28	114	0	0	206	0	0
29	106	0	0	0	-1	-1
30	114	0	0	0	-1	-1
31	114	0	0	0	-1	-1

INPUT PARAMETERS AND THEIR VALUES – FIG. 46

step	force(s) selection rule	node placement rule	force indeterminacies rule	seed number	desired entropy rate	applied entropy rate
1	109	3	0	60	0	0
2	109	3	0	0	-1	-1
3	117	3	0	40	0	0
4	117	3	0	0	-1	-1
5	-1	0	0	495	0	0
6	106	0	0	901	0	0
7	106	0	0	730	0	0
8	-1	0	0	495	0	0
9	114	0	0	1000	0	0
10	114	0	0	679	0	0
11	61	0	0	30	0	0
12	117	0	0	622	0	0
13	106	0	0	272	0	0
14	106	0	0	0	-1	-1
15	114	0	0	603	0	0
16	114	0	0	881	0	0
17	114	0	0	596	0	0
18	106	0	0	638	0	0
19	106	0	0	0	-1	-1
20	106	0	0	0	-1	-1
21	106	0	0	0	-1	-1

INPUT PARAMETERS AND THEIR VALUES – FIG. 47I &FIG. 48

step	force(s) selection rule	node placement rule	force indeterminacies rule	seed number	desired entropy rate	applied entropy rate
1	112	3	0	90	1	1
2	114	3	0	65	0	0
3	114	3	0	507	0	0
4	114	3	0	56	0	0
5	114	3	0	353	0	0
6	114	3	0	23	0	0
7	114	3	0	170	0	0
8	114	3	0	100	0	0
9	43	3	0	0	-1	-1
10	43	3	0	0	-1	-1
11	43	3	0	0	-1	-1

INPUT PARAMETERS AND THEIR VALUES – FIG. 47II

step	force(s) selection rule	node placement rule	force indeterminacies rule	seed number	desired entropy rate	applied entropy rate
1	112	0	0	342	1	1
2	114	3	0	395	0	0
3	114	3	0	395	0	0
4	114	3	0	197	0	0
5	114	3	0	473	0	0
6	114	3	0	496	0	0
7	114	3	0	449	0	0
8	114	3	0	119	0	0
9	114	3	0	119	0	0
10	114	3	0	205	0	0
11	-1	3	0	119	-1	-1
12	-1	3	0	119	-1	-1
13	114	3	0	0	-1	-1

INPUT PARAMETERS AND THEIR VALUES – FIG. 47III

step	force(s) selection rule	node placement rule	force indeterminacies rule	seed number	desired entropy rate	applied entropy rate
1	112	0	0	1	1	1
2	114	3	0	47	0	0
3	114	3	0	47	0	0
4	114	3	0	55	0	0
5	114	3	0	98	0	0
6	114	3	0	111	0	0
7	114	3	0	132	0	0
8	-1	3	0	132	-1	-1
9	-1	3	0	132	-1	-1
10	114	3	0	0	-1	-1

INPUT PARAMETERS AND THEIR VALUES – FIG. 49I

step	force(s) selection rule	node placement rule	seed number	desired entropy rate	applied entropy rate
1	-1	1	198	0	0
2	-1	1	198	0	0
3	-1	1	198	0	0
4	-1	1	246	0	0
5	-1	1	246	0	0
6	-1	1	246	0	0
7	-1	1	376160596	-1	-1
8	-1	1	376160596	-1	-1
9	-1	1	1146688524	-1	-1
10	-1	0	1146688524	-1	-1
11	-1	0	1146688524	-1	-1
12	0	0	795316633	-1	-1
13	-1	1	479	1	1
14	-1	1	348	0	0
15	-1	1	324	0	0
16	-1	0	1027200916	-1	-1
17	-1	0	1027200916	-1	-1
18	0	0	1797728844	-1	-1

INPUT PARAMETERS AND THEIR VALUES – FIG. 49II

step	force(s) selection rule	node placement rule	seed number	desired entropy rate	applied entropy rate
1	-1	1	164	0	0
2	-1	0	1327166544	-1	-1
3	-1	1	164	0	0
4	-1	0	1139894790	-1	-1
5	-1	1	246	0	0
6	-1	0	1439563219	-1	-1
7	-1	1	99	0	0
8	-1	0	900919574	-1	-1
9	-1	1	430	0	0
10	-1	0	362275929	-1	-1
11	-1	1	24	0	0
12	-1	0	1783844177	-1	-1
13	-1	1	299	1	1
14	-1	1	4	0	0
15	-1	1	689	0	0
16	-1	0	1708969098	-1	-1
17	-1	0	1708969098	-1	-1
18	0	0	1357597207	-1	-1

INPUT PARAMETERS AND THEIR VALUES – FIG. 49III

step	force(s) selection rule	node placement rule	seed number	desired entropy rate	applied entropy rate
1	-1	1	164	0	0
2	-1	0	818953562	-1	-1
3	-1	1	164	0	0
4	-1	0	1402209736	-1	-1
5	-1	1	246	0	0
6	-1	0	863566091	-1	-1
7	-1	1	99	0	0
8	-1	0	2121034202	-1	-1
9	-1	1	430	0	0
10	-1	0	556806729	-1	-1
11	-1	1	89	0	0
12	-1	0	18163084	-1	-1
13	-1	1	113	1	1
14	-1	1	214	0	0
15	-1	0	1858887369	-1	-1
16	-1	1	689	0	0
17	-1	0	1320243724	-1	-1
18	0	0	968871833	-1	-1

INPUT PARAMETERS AND THEIR VALUES – FIG. 49IV

step	force(s) selection rule	node placement rule	seed number	desired entropy rate	applied entropy rate
1	-1	1	273	0	0
2	-1	0	1013484362	-1	-1
3	-1	1	273	0	0
4	-1	0	474840717	-1	-1
5	-1	1	259	0	0
6	-1	0	1732308828	-1	-1
7	-1	1	99	0	0
8	-1	0	1006393429	-1	-1
9	-1	1	430	0	0
10	-1	0	467749784	-1	-1
11	-1	1	89	0	0
12	-1	0	2076589786	-1	-1
13	-1	1	331	1	1
14	-1	1	722	0	0
15	-1	0	160990422	-1	-1
16	-1	1	389	0	0
17	-1	1	440	0	0
18	-1	0	692543434	-1	-1
19	0	0	692543434	-1	-1

INPUT PARAMETERS AND THEIR VALUES – FIG. 50I

step	force(s) selection rule	node placement rule	force indeterminacies rule	seed number	desired entropy rate	applied entropy rate
1	106	3	0	443	0	0
2	106	3	0	300	-1	-1
3	106	3	0	919	0	0
4	106	3	0	416	-1	-1
5	106	3	0	492	0	0
6	106	3	0	219	-1	-1
7	106	3	0	376	0	0
8	106	3	0	953	0	0
9	106	3	0	579	0	0
10	106	3	0	689	-1	-1
11	106	3	0	689	-1	-1
12	106	3	0	689	-1	-1
13	114	3	0	539	0	0
14	114	3	0	473	0	0
15	66	3	0	680	-1	-1
16	66	3	0	680	-1	-1
17	66	3	0	0	-1	-1

INPUT PARAMETERS AND THEIR VALUES – FIG. 50II

step	force(s) selection rule	node placement rule	force indeterminacies rule	seed number	desired entropy rate	applied entropy rate
1-2	106	3	0	746	0	-1
						-1
3	106	3	0	0	-1	-1
4	114	3	0	529	-1	-1
5-6	114	3	0	631	-1	-1
						-1
7-8	137	3	0	250	0	0
						0
9-10	137	3	0	488	-1	0
						-1
11	137	3	0	488	-1	-1

INPUT PARAMETERS AND THEIR VALUES – FIG. 50III

step	force(s) selection rule	node placement rule	force indeterminacies rule	seed number	desired entropy rate	applied entropy rate
1	106	7	0	3	0	0
2-8	106	7	0	3	-1	-1
						-1
						-1
						-1
						0
						-1
						-1
9-12	106	7	0	12	-1	0
						0
						-1
						-1

INPUT PARAMETERS AND THEIR VALUES – FIG. 50IV

step	force(s) selection rule	node placement rule	force indeterminacies rule	seed number	desired entropy rate	applied entropy rate
1	106	3	0	664	0	0
2	106	3	0	703	0	0
3	106	3	0	703	-1	-1
4	106	3	0	719	0	0
5	106	3	0	537	0	0
6	36	3	0	422	-1	-1
7	106	3	0	560	0	0
8	36	3	0	556	0	0
9	106	3	0	671	-1	-1
10	106	3	0	628	0	0
11	106	3	0	216	0	0
12	106	3	0	299	0	0
13	106	3	0	429	0	0
14	106	3	0	429	-1	-1
15	106	3	0	429	-1	-1
16	106	3	0	494	0	0
17	114	3	0	825	0	0
18	114	3	0	266	0	0
19	114	3	0	708	-1	-1
20	106	3	0	414	0	0
21	106	3	0	414	-1	-1

INPUT PARAMETERS AND THEIR VALUES – FIG. 50V

step	force(s) selection rule	node placement rule	force indeterminacies rule	seed number	desired entropy rate	applied entropy rate
1-3	106	3	0	500	-1	-1
						-1
						-1
4-6	114	3	0	500	-1	-1
						-1
						-1
7-8	138	7	0	26	0	0
						0
9-11	138	7	0	339	-1	0
						-1
						-1

INPUT PARAMETERS AND THEIR VALUES – FIG. 50VI

step	force(s) selection rule	node placement rule	force indeterminacies rule	seed number	desired entropy rate	applied entropy rate
1	106	3	0	139	0	0
2	-1	3	0	159	-1	-1
3	-1	3	0	159	-1	-1
4	-1	3	0	159	-1	-1
5	-1	3	0	80	0	0
6	-1	3	0	80	-1	-1
7	-1	3	0	114	0	0
8	-1	3	0	239	0	0
9	-1	3	0	78	0	0
10	-1	3	0	78	-1	-1
11	-1	3	0	78	-1	-1
12	-1	3	0	78	-1	-1
13	-1	3	0	78	-1	-1
14	-1	3	0	78	-1	-1
15	-1	3	0	78	0	-1

INPUT PARAMETERS AND THEIR VALUES – FIG. 50VII

step	force(s) selection rule	node placement rule	force indeterminacies rule	seed number	desired entropy rate	applied entropy rate
1	-1	0	0	704	-1	-1
2	-1	0	0	285	-1	-1
3	-1	0	0	285	-1	-1
4	-1	0	0	285	-1	-1
5	-1	0	0	285	-1	-1
6	-1	0	0	285	-1	-1
7	-1	3	0	382	0	0
8	-1	3	0	382	-1	-1
9	-1	3	0	382	-1	-1
10	106	0	0	217	-1	-1

INPUT PARAMETERS AND THEIR VALUES – FIG. 50VIII

step	force(s) selection rule	node placement rule	force indeterminacies rule	seed number	desired entropy rate	applied entropy rate
1	106	3	0	54	0	0
2	106	3	0	87	-1	-1
3	106	3	0	71	-1	-1
4	106	3	0	79	0	0
5	106	3	0	1	-1	-1
6	106	3	0	14	0	0
7	106	3	0	24	-1	-1
8	106	3	0	75	-1	-1
9	106	3	0	120	-1	-1
10	106	3	0	46	0	0
11	106	3	0	140	-1	-1
12	106	3	0	64	-1	-1
13	106	3	0	39	-1	-1

INPUT PARAMETERS AND THEIR VALUES – FIG. 50IX

step	force(s) selection rule	node placement rule	force indeterminacies rule	seed number	desired entropy rate	applied entropy rate
1	106	7	0	358	0	0
2-4	106	7	0	1	-1	-1
						-1
						-1
5-11	114	7	0	529	-1	-1
						-1
						-1
						0
						0
						-1
						-1

INPUT PARAMETERS AND THEIR VALUES – FIG. 50X

step	force(s) selection rule	node placement rule	force indeterminacies rule	seed number	desired entropy rate	applied entropy rate
1-11	106	3	0	201	-1	-1
						-1
						-1
						-1
						-1
						-1
						0
						0
						0
						-1
						-1

INPUT PARAMETERS AND THEIR VALUES – FIG. 50XI

step	force(s) selection rule	node placement rule	force indetermi-nacies rule	seed number	desired entropy rate	applied entropy rate
1	106	3	0	137	0	0
2	106	3	0	100	-1	-1
3	106	3	0	119	-1	-1
4	106	3	0	0	0	0
5	106	3	0	122	-1	-1
6	106	3	0	56	0	0
7	106	3	0	183	-1	-1
8	106	3	0	81	-1	-1
9	106	3	0	140	-1	-1
10	146	3	0	58	-1	-1
11	146	3	0	100	0	0
12	146	3	0	67	-1	-1
13	146	3	0	67	-1	-1

INPUT PARAMETERS AND THEIR VALUES – FIG. 50XII

step	force(s) selection rule	node placement rule	force indetermi-nacies rule	seed number	desired entropy rate	applied entropy rate
1	106	3	0	157	0	0
2	106	3	0	157	-1	-1
3	106	3	0	134	0	0
4	106	3	0	44	-1	-1
5	106	3	0	17	0	0
6	146	3	0	28	-1	-1
7	106	3	0	31	0	0
8	106	3	0	15	0	0
9	106	3	0	67	0	0
10	146	3	0	56	0	0
11	106	3	0	56	-1	-1
12	106	3	0	46	0	0
13	106	3	0	46	-1	-1
14	146	3	0	25	0	0
15	106	3	0	59	0	0
16	106	3	0	159	-1	-1
17	114	3	0	164	0	0
18	106	3	0	149	-1	-1
19	106	3	0	149	0	-1

INPUT PARAMETERS AND THEIR VALUES – FIG. 50XIII

step	force(s) selection rule	node placement rule	force indeterminacies rule	seed number	desired entropy rate	applied entropy rate
1	106	3	0	0	-1	-1
2	114	3	0	0	-1	-1
3	106	7	0	101	0	0
4	106	7	0	131	0	0
5	106	7	0	0	-1	-1
6	106	3	0	0	-1	-1
7	114	3	0	0	-1	-1
8	114	3	0	0	-1	-1
9	106	3	0	0	-1	-1
10	106	3	0	0	-1	-1

INPUT PARAMETERS AND THEIR VALUES – FIG. 50XIV

step	force(s) selection rule	node placement rule	force indeterminacies rule	seed number	desired entropy rate	applied entropy rate
1-10	106	0	0	0	-1	-1
						-1
						-1
						-1
						-1
						-1
						0
						0
						-1
						-1

INPUT PARAMETERS AND THEIR VALUES – FIG. 50XV

step	force(s) selection rule	node placement rule	force indeterminacies rule	seed number	desired entropy rate	applied entropy rate
1	106	3	0	157	0	0
2	106	3	0	157	-1	-1
3	106	3	0	157	-1	-1
4	106	3	0	157	-1	-1
5	106	3	0	157	0	0
6	106	3	0	157	0	0
7	106	3	0	157	-1	-1
8	106	3	0	157	-1	-1
9	106	3	0	157	-1	-1
10	106	3	0	157	-1	-1
11	106	3	0	157	1	1
12	106	3	0	157	-1	-1
13	106	3	0	157	-1	-1
14	106	3	0	157	0	-1

INPUT PARAMETERS AND THEIR VALUES – FIG. 50XVI

step	force(s) selection rule	node placement rule	force indeterminacies rule	seed number	desired entropy rate	applied entropy rate
1-10	106	3	0	381	-1	-1
						-1
						-1
						-1
						-1
						-1
						0
						0
						-1
						-1

INPUT PARAMETERS AND THEIR VALUES – FIG. 50XVII

step	force(s) selection rule	node placement rule	force indeterminacies rule	seed number	desired entropy rate	applied entropy rate
1-11	106	7	0	635	-1	-1
						-1
						-1
						-1
						-1
						-1
						-1
						0
						0
						-1
						-1

INPUT PARAMETERS AND THEIR VALUES – FIG. 50XVIII

step	force(s) selection rule	node placement rule	force indeterminacies rule	seed number	desired entropy rate	applied entropy rate
1	106	7	0	89	0	0
2	106	7	0	68	-1	-1
3	106	7	0	0	0	0
4	106	7	0	55	-1	-1
5	106	7	0	130	-1	-1
6	106	7	0	62	0	0
7	106	7	0	94	0	0
8	106	7	0	66	-1	-1
9	106	7	0	117	-1	-1
10	106	7	0	0	-1	-1
11	114	5	0	58	0	0
12	114	5	0	25	-1	-1
13	114	5	0	37	-1	-1

Curriculum vitae

Ioannis Mirtsopoulos

Born in October 23rd, 1988 in Serres, Greece

Route des Arsenaux 9, 1700, Fribourg, Switzerland
johnmirts@hotmail.com
www.linkedin.com/in/ioannismirtsopoulos/

Academic record

01•2018-01•2022	**École polytechnique fédérale de Lausanne (EPFL) • Lausanne • Switzerland** Doctor of Sciences in Civil Engineering (Docteur ès Sciences) Doctoral School in Civil and Environmental Engineering (EDCE) Structural Xploration Lab (SXL) Title thesis: *Policy-based Exploration of Equilibrium Representations (PEER): A computational framework for conceptual structural design.* Supervisor: Prof. Corentin Fivet
09•2015-09•2016	**Swiss Federal Institute of Technology (ETH) • Zurich • Switzerland** Master of Advanced Studies in Digital Fabrication (MAS DFAB) Title thesis: *Robotic fabrication of Catalan vaulting* Supervisors: Prof. Fabio Gramazio, Prof. Matthias Kohler (Chair of Architecture and Digital Fabrication)
09•2013-07•2015	**Delft University of Technology (TUDeflt) • Delft • The Netherlands** Master of Science in Building Technology (MSc, Ir.) Title thesis: *Additive manufacturing of algorithmically form generated nodes for free form space frames* Supervisors: Prof. Andrew Borgart, Prof. Paul de Ruiter, Prof. Joris Smits (Chair of Structural Design & Mechanics and Chair of Design Informatics)
10•2007-07•2013	**Aristotle University of Thessaloniki (AUTh) • Thessaloniki • Greece** Diploma in Architectural Engineering (Dipl. Ing, MEng equivalent) Title thesis: *Form and structural design in digital approach of* architecture Supervisor: Prof. Maria Voyatzaki
10•2006-09•2007	**University of Thessaly (UTh) • Volos • Greece** studies in Architectural Engineering

Research experience

01•2018-01•2022	**Doctoral assistant** at **Structural Xploration Lab (SXL)** • École polytechnique fédérale de Lausanne (EPFL) • Lausanne • Switzerland
11•2017-01•2018	**Scientific assistant** at **Structural Xploration Lab (SXL)** • École polytechnique fédérale de Lausanne (EPFL) • Lausanne • Switzerland
10•2016-06•2017	**Research assistant** at **Block Research Group (BRG)** • Swiss Federal Institute of Technology (ETH) • Zurich • Switzerland

Teaching

Fall 2018, 2020	**Teaching assistant** • École polytechnique fédérale de Lausanne (EPFL) • Lausanne • Switzerland Interactive conceptual design of structural forms, led by Prof. Fivet • *Master Level*
Spring 2018	**Teaching assistant** • École polytechnique fédérale de Lausanne (EPFL) • Lausanne • Switzerland Geometry, Calculus, Linear Algebra and Differential Equations • *Bachelor Level*

Awards

2020	**EDCE EPFL PhD Mobility Award** • Visiting research mobility grant Massachusetts Institute of Technology (MIT)
2019	**eCAADe PhD grant** • Porto • Portugal

Professional membership

2020 - ongoing	International Association of Structures and Architecture (IASA) • member
2014 - ongoing	International Association of Shell and Spatial Structures (IASS) • member
2013 - ongoing	Technical Chamber of Greece (TEE-TCG) • chartered engineer

Software

Scripting	C#, Python, WPF, C++, Visual Basic, KUKA KRL, MAXScript, MEL
Digital design	McNeel Rhinoceros • Grasshopper Autodesk 3ds Max Design • Maya • AutoCAD Adobe Indesign • Illustrator • Photoshop
Documentation	MS Office

Languages

Greek	Native
English	C2
Spanish	B2
French	A2
German	A2

List of related publications

Journal articles

Mirtsopoulos I., Fivet C., Exploratory conceptual structural design through policy-based generation of equilibrium representations, *Computer-Aided Design* (2022) (under review).

Mirtsopoulos I., Fivet C., Design space exploration through force-based grammar rule, *archi-DOCT*, vol. 8 (3) (2020).

Conference papers

Mirtsopoulos I., Fivet C., Exploration of static equilibrium representations; policies and genetic algorithms, *Proceedings of the 5th International Conference on Structures and Architecture – ICSA* 2022, 6-8 July 2022, Aalborg, Denmark (accepted for publication).

Mirtsopoulos I., Fivet C., Grammar-based generation of bar networks in static equilibrium with bounded bar lengths, *Proceedings of the International Association of Shell and Spatial Structures IASS 2020-21*, 23-27 August 2021, Surrey, UK

Mirtsopoulos I., Fivet C., Grammar-based generation of trusses within non-convex domains, *Proceedings of the International fib Symposium on Conceptual Design of Structures 2019*, 26-27 September 2019, Madrid, Spain