TECHNISCHE UNIVERSITÄT MÜNCHEN
Lehrstuhl für Produktentwicklung

Solving Engineering Design Problems through a Combination of Generative Grammars and Simulations

Amir Hooshmand Shabanabadi

Vollständiger Abdruck der von der Fakultät für Maschinenwesen der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs

genehmigten Dissertation.

Vorsitzender:		Univ.-Prof. Dr. Ir. Daniel Rixen
Prüfer der Dissertation:	1.	Univ.-Prof. Dr.-Ing. Udo Lindemann
	2.	Prof. Matthew Ira Campbell, PhD / Oregon State University, Corvallis, OR, USA

Die Dissertation wurde am 16.01.2014 bei der Technischen Universität München eingereicht und durch die Fakultät für Maschinenwesen am 20.03.2014 angenommen.

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über
http://dnb.d-nb.de abrufbar.

ISBN 978-3-8439-1573-1

1. Auflage 2014

ABSTRACT

In this thesis a framework is proposed which considers the use and the applicable different knowledge levels at various abstractions within an automated design process. For an effective utilization of available design information and knowledge, the computational synthesis process is divided into three main phases: search, optimization and modification. The generative graph grammar for representing design knowledge is used but some aspects are applicable to other representations as well. The generality and flexibility of the proposed mechanism is demonstrated by automating the synthesis of three engineering design problems in two domains. For all cases the graph grammar interpreter, GraphSynth, is used to carry out graph transformations, which define different topologies for a problem. The proposed method combines generative design synthesis methods with conventional simulation models, leading to a significant reduction in the numerical operations in all three design problems.

The first engineering design case is topology and shape optimization of fluid channels. By utilizing a multiple representation approach, there is no need for a large grid of variables to represent the topology, which causes significant computational savings and allows the simulation model to be independent. After evaluating and optimizing the generated graphs, they are transformed into meaningful 3D shapes to be simulated in a CFD solver. The second design problem is structural layout optimization. Through applying the proposed framework, a design technique is produced to achieve optimal topologies and shapes for cable trusses considering various constraints such as stress, displacement, stability. Furthermore, manufacturing issues and material imperfections and limitations can be considered in the synthesis. The last design problem is to produce large irregular tensegrity structures. Unlike most of the form-finding methods, the approach does not require the description of the connectivity of the tensegrity structures to define the shape of the tensegrities. It uses graphs to represent the tensegrity structures, which allows a very fast generation of stable tensegrity solutions for a given design problem.

The effectiveness of the proposed method in all of the cases is checked by solving and comparing a variety of available test problems found in the literature. Furthermore by solving complex large scale three dimensional problems, the robustness of the method is tested. The results show that the approach not only creates the existing solutions for available test problems, it creates new structures that have never been seen before. The contribution achieved in this work provides a mechanism for designers to utilize design information and knowledge at all abstraction levels. Besides applying and testing the framework in other domains and design cases, future work may include the improvement of search strategies, which are used for exploring the design space to achieve faster results in larger design spaces.

ACKNOWLEDGEMENTS

This work was written during my tenure as both a scientific research assistant at the Institute of Product Development at the Technische Universität München (TUM) from November 2010 until January 2014 and as an exchange researcher at the Automated Design Lab at the University of Texas at Austin from September 2012 to July 2013.

I am greatly thankful for the mentorship provided by Prof. Matthew Campbell. Without his intense support of my research and confidence in my abilities to manifest my ideas, this thesis would not have been possible. His abundant insights and willingness to share knowledge helped me a lot in writing this doctoral thesis in a much shorter time than originally planned.

Many thanks to Prof. Kristina Shea for her supervision during the first 17 months of my research. Our research meetings were invaluable for increasing my insight in the field of CDS.

I would like to heartily thank Prof. Dr.-Ing. Udo Lindemann for the wonderful support of my work and for offering me a very productive and pleasant working environment at the Institute of Product Development.

I would also like to thank my sponsor institute "Institute for Advanced Study, Technische Universität München (IAS)". And many thanks to the IAS staff especially Ms. Stefanie Hofmann for her support and helpfulness.

Special thanks to all my colleagues at the Institute of Product Development. I learned so much through the opportunities you provided.

Most importantly, this work would have not been possible without the continuous understanding, love and support from my wife Nina Polous, who always encouraged me in my work and offered patience and comprehension in stressful times.

Garching, January 2014 Amir Hooshmand

The following publications are part of the work presented in this thesis:

Hooshmand, A., & Campbell, M. I., (2014). Layout Synthesis of Fluid Channels Using Generative Graph Grammars. AIEDAM, Computational Design Synthesis Special Issue, Vol. 28, No.3, 2014.

Hooshmand, A., & Campbell, M. I., (2013). Topology Optimization of Fluid Channels Uing Generative Graph Grammars. In 39th Design Automation Conference,. ASME.

Hooshmand, A., Schlaich, M., Belaus, L., & Campbell, M. I., (2013). CDS Platform - a Platform for Multi-Physics Computational Design Synthesis. In U. Lindemann, S. Venkataraman, Y. Kim, S. Lee, P. Papalambros, & W. Chen (Eds.), 19th International Conference on Engineering Design (ICED13), Design for Harmonies, Vol.9: Design Methods and Tools (pp. 099–108). Seoul, Korea: Design Society.

Hooshmand, A., Campbell, M. I., & Shea, K., (2012). Steps in Transforming Shapes Generated With Generative Design Into Simulation Models. In Volume 3: 38th Design Automation Conference, Parts A and B (p. 883-892). ASME.

Hooshmand, A., & Campbell, M. I., Truss Layout Optimization using Generative Design Synthesis Approach. Computers & Structures. (under review)

Hooshmand, A., & Campbell, M. I., Tensegrity Form-Finding Using Generative Design Synthesis Approach. International Journal of Solids and Structures. (under review)

CONTENTS

1. Introduction

The significant amount of computing power and memory of today's computers has enabled the development of new methods and algorithms for diverse areas of application. However, the amount of design automation in different fields varies drastically. Computer aided tools for designing integrated circuits covers the whole design process from synthesis to simulation, analysis, and optimization (Whitney, 1996). In mechanical engineering, Computer Aided Design (CAD) technologies are used mainly for analysis and representational purposes (Celani, 2002). These tools mainly concentrate on analysis and optimization of specific details of a proposed solution (Alber and Rudolph, 2004). They do not typically cover synthesis and leave the most critical part of the conceptual design, i.e. finding a solution, to human designers (Hoisl and Shea, 2011; Schotborgh et al., 2006). Design synthesis is an area of research, which is focused on developing methods and tools, to support the generation of solutions at the early phases of the design (Chakrabarti et al., 2011). One of the main problems of automated design synthesis is utilizing design knowledge and information during the synthesis process. In most of the developed synthesis methods in the literature, through an interactive visual approach for generating solutions, the user can easily prevent the creation of invalid designs, whereas it is not in an automatic process possible (among others see (Hoisl and Shea, 2011)). The problem lays in the incapability to capture and use design information and knowledge for guiding the process in the automatic generation not in the lack of information. Because in the interactive generation process, the designer uses information in the design to make the design decisions for creating the solutions, whereas the computer does not have access to such information for making decisions. Hence, developing adequate approaches to cope with the design information and knowledge of an evolving artifact is of vital importance.

In this thesis we propose a framework for capturing and using design knowledge in the synthesis processes using generative graph grammars. This framework has two parts. First, the design process is divided into three main phase; search, optimization and modification. Each phase may have multiple sub phases with different levels of abstraction. The reason for this division is that a design solution evolves during the design process, which means the abstraction level of the design changes progressively. Therefore for an efficient and effective utilization of design information and knowledge during the synthesis process, different mechanisms in different phases of the design are required to guide the design decision-making. In general in the search phase, the topology of the solutions is defined, whereas in the optimization phase the shape of the design solutions is determined. In the final modification phase the detailed-design is accomplished. The second part of the framework is providing a base for using appropriate type of information and knowledge in search, optimization and modification phases of the design. This base determines the applicability of available design information and knowledge in each phase.

By solving three engineering design problems in two different domains; fluid mechanics and structural mechanics, the proposed framework is validated. The design decision-making process is made more efficient by an appropriate leveraging of design information and knowledge. The design problems are extensively discussed in chapters 2, 3 and 4. The results show that dividing the synthesis process in three phases and utilizing design information and knowledge in these phases – however with different degrees of utilization – not only is a guarantee to generate many alternative valid and optimum solutions, it eliminates the generation of invalid design candidates. These invalid solutions normally constitute a big part of the design space in automatic synthesis approaches due to inappropriate design decision-makings.

After this introductory section, in sections 1 the division of the design process in three phases is discussed. Section 2 introduces different levels of knowledge and their utilization in the synthesis process. In section 3, capturing design information and knowledge in a representation is discussed. And section 4 concludes the first chapter.

1.1 Three Phases of the Synthesis Process

Design is the process of transforming information from one state to another (Hubka et al., 1988; Ognjanovic, 1999) and a decision to transform existing information to a new state is based on available information and knowledge (Hicks et al., 2002). Therefore a key issue in developing new intelligent design automation approaches and tools is handling the evolving information of the design process. To cope with this changing information and knowledge the design process is divided into three phases based on different abstraction levels of the design knowledge: search for concept generation, optimization for concept selection, and modification of design details (figure 1). Search is the first step of the design process, which is used to explore the design space in the most abstract level of the design. As illustrated in figure 1, this step may also be divided into multiple abstraction levels called sub-levels. In the search phase of the design, the valid solutions for the desired design problem are generated. For instance for a structural design problem, the valid solutions are different truss structure configurations with different components, which can hold the load without violating any criterion. The number of solutions depends normally upon various initial requirements and the boundary conditions of the problem. After exploring the design space and generating all feasible candidates in the search phase, the optimization phase begins.

Indeed, the search phase is responsible to explore the topological variation of the designs and the optimization phase defines the best shape for each candidate. The optimization phase may contain a sophisticated algorithm for shape and size optimization using an adequate evaluation method or a simple evaluation approach to sort the generated solutions in the search phase based on a performance criterion. For instance after defining the topology of a truss structure, the shape of the truss (spatial position of the joints) or the thickness of the components can be optimized using finite element methods or just simply evaluated based on a criterion such as maximum displacement at the load point. So, based on the results of this step, best candidates are selected for the final phase. The third step of the framework is the modification phase, in which secondary details are added to the design such as adding a chamfer. The effect of this step on the design is not as important as the first and second phases.

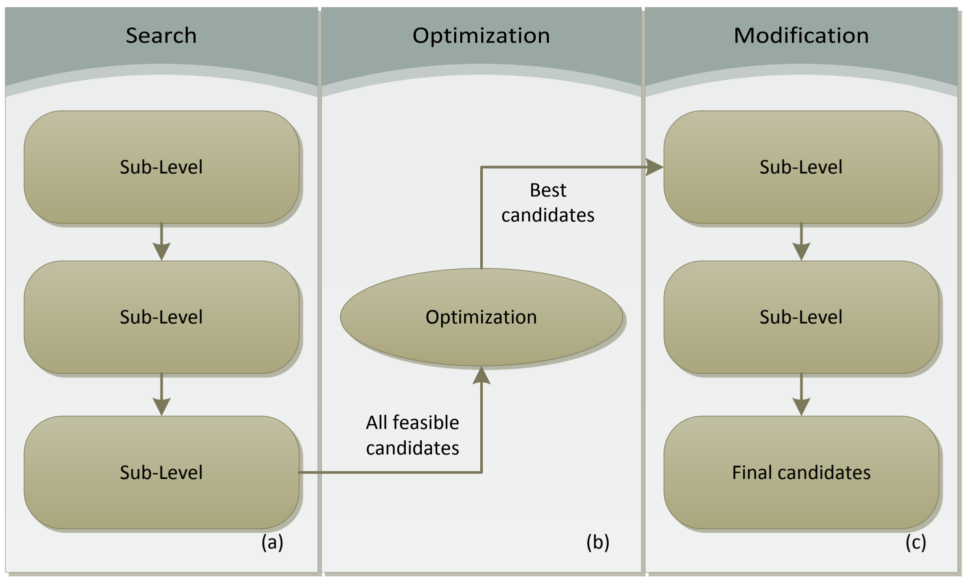


Figure 1-1: Three Phases of the synthesis.

The approach presented in this section, considers the abstraction level of the design decreasing progressively during three phases and corresponding sub-phases. However it is of vital importance to use appropriate knowledge to generate the solutions at each abstraction level to avoid many unnecessary detailed analyses and optimization of the concepts. Furthermore, early feasibility assessment and evaluation of abstract solutions is essential for restricting the search space to best concepts, which consequently can reduce the design time and the number of design analyses (Netten and Vingerhoeds, 1997). Without suitable restriction of the design space – through meaningful generation and consequent evaluation of the solutions – considerable additional effort is required to generate many invalid or poor performance solutions.

However, due to imprecise and incomplete design requirements and constraints at the early stages it is difficult to capture the design knowledge in order to generate valid solutions. Further, evaluating the generated solutions with available analysis tools and methods is not possible, because the input to most of these tools should be a fully defined design. The behavior of solutions cannot be predicted and reliably evaluated at the early abstraction levels, because strong component interactions – which are not mainly defined yet – complicates the design decision-making (Netten and Vingerhoeds, 1997). For instance in a spatial frame structure, the stresses and strains in a component mainly depend on other components and also the overall structure. Therefore to support these essential decision-making processes at the search phase a design information and knowledge should be used, which is flexible and are not depending upon many such specifications. In the next section the requirements for capturing and using design information and knowledge in various abstraction levels are defined.

1.2 Applicability of Knowledge Levels

The decision-making process of conceptual design is very complex, because at different abstraction levels – based on the level of specifications – different inference processes are required. These decision-making processes are based on available information and knowledge. But we should search for both available and applicable design information and knowledge at various abstraction levels. Hicks et al. (2002) define four levels for design knowledge with different applicability scopes (figure 2). As shown in figure 2, the highest levels of knowledge are general and generic knowledge, which are applicable to unfamiliar situations, whereas the specific knowledge is restricted to familiar situations and case knowledge is applicable only in specific situations. The vertical arrows in the figure 2 illustrate the applicability field of each level of knowledge. Indeed there is a correspondence between the abstraction level of the design problem and the familiarity situation of the problem. The more abstract a design, the more unfamiliar the situation, and the more detailed an artifact the more known (familiar) are the specifications.

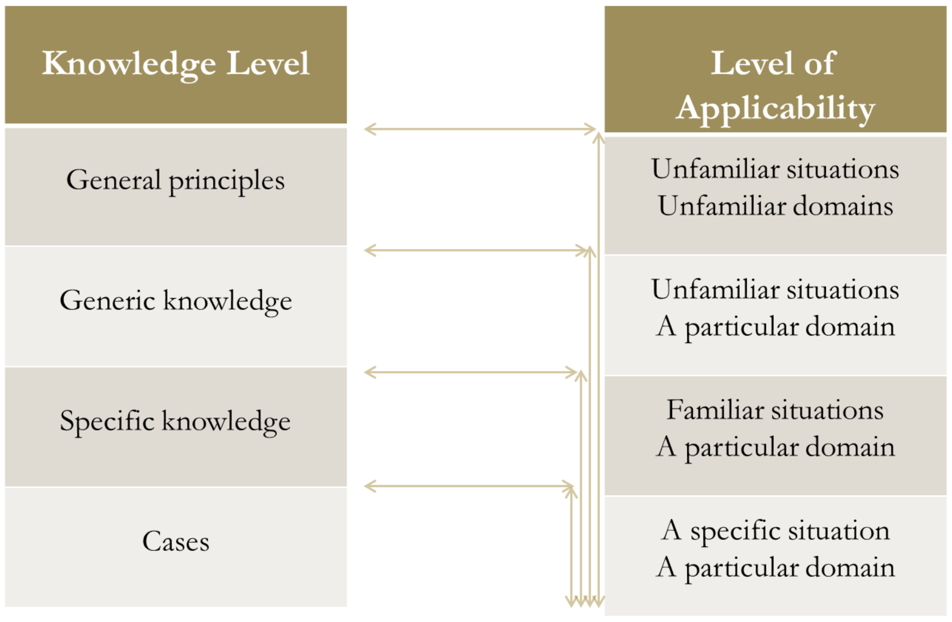


Figure 1-2: Knowledge levels and states of applicability (Hicks et al., 2002)

Figure 3 combines the proposed mechanism in the previous section with the Hicks et al. (2002) knowledge levels. It shows that in the search phase, which is the most abstract design level, because the solutions are not yet formed and the situation is unfamiliar, only generic and general knowledge are applicable. In the second phase, as the topology is defined and the design is concretized to a higher degree and the situation is familiar, specific knowledge is also applicable. And finally in the third phase of the design, all levels of knowledge are applicable.

Using generic or general levels knowledge at the early stages of design may be an essential solution for incomplete design knowledge. Because these abstract levels of knowledge are normally simple facts, which means their capturing and representation is easy. As an example, we consider the Newton's third law of motion. This law says; the same force that a body exerts on a second body will be exerted back upon it through the second body but in the opposite direction. Capturing and representing this law is very simple and for using it in an automatic approach, no quantitative value is required. This law can be used in unfamiliar situations and applied in unfamiliar domains; therefore it is a general type of knowledge. For instance it can be applied in the field of structural mechanics, fluid mechanics or in designing electro magnetics artifacts, because in all these domains we are handling forces. Through three case studies in chapters 2, 3 and 4, this formal approach for using design information and knowledge is further discussed and it is shown how knowledge about the design knowledge can increase the efficiency of the design synthesis process.

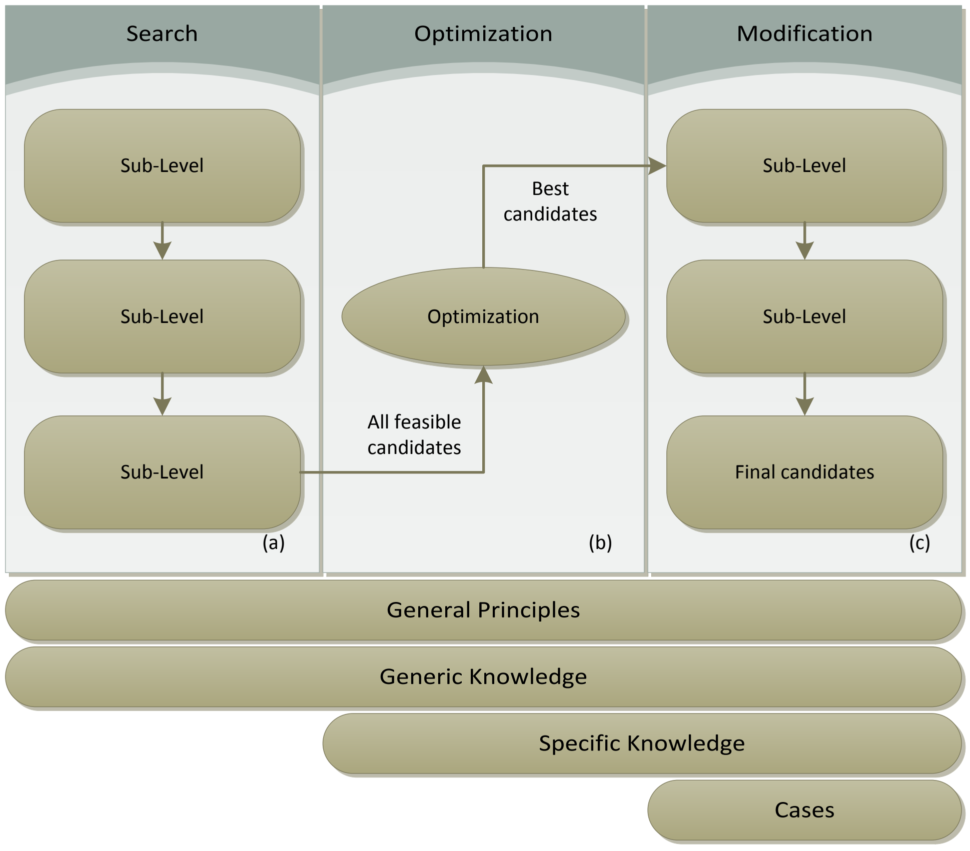


Figure 1-3: Using design information and knowledge in various abstraction levels.

1.3 Capturing Design Information and knowledge

To have a better understanding of mechanisms and procedures for capturing and using the design information and knowledge, formal definitions for information and knowledge are first discussed here.

1.3.1 Design information and knowledge

Data, information and knowledge terms have been defined in different fields of research and are reviewed within the context of engineering design by Court (Court, 1995). Based on his work and other researchers such as Marsh (Marsh, 1997), Hicks et al., (2002) conclude that knowledge has two aspects, knowledge processes and knowledge elements; "the knowledge process is the procedure(s) utilized by the individual to infer the knowledge element from information, other knowledge elements or a combination of each". Representing the knowledge processes in a formalism is in general a complicated task, because they are mainly considered as within-person activities (Hicks et al., 2002). Whereas, knowledge element representation is much easier, because the knowledge elements are in fact taken as information (Boston, 1998). In engineering sciences the knowledge processes are considered as scientific practices or procedures (Ehrlenspiel, 1997). Hicks et al. (2002), define information elements as the totality of one or some data elements and one or more context descriptors, where “the context descriptor(s) clarify the meaning of the data element, and are themselves one or a combination of data elements”, for instance mass of an object, or acceleration of an object. Some researcher classify information to different categories such as formal and informal or structured and unstructured, but in this study (due to the computerized nature of the information) the author considers information as formal and structured. To further clarify the subject the following simple example is given. Newton’s second law of motion says that, the sum of all forces on any object is equal to the mass of that object multiplied by its acceleration ($F = ma$). In this example m (mass) and a (acceleration) are two knowledge elements (information) and their multiplication is the knowledge process to create another knowledge element F (force).

1.3.2 Capturing information and knowledge

Capturing design information and knowledge generally means generating a representation for them. “Design is most appropriately characterized as a construction of representations” (Visser, 2006). A representation scheme is used to store, organize, process, and access information and knowledge elements. Furthermore it is used to capture the relationships between information and knowledge elements and provide a structure, through which later reasoning and reuse of information and knowledge will be possible. Indeed representation schemes catch the fundamental characteristics of a problem domain in their structure and make it available for a problem-solving procedure (Luger, 2008). Luger (2008) argues that two main criteria for a knowledge representation scheme are its expressiveness and efficiency, which may contradict in different cases. Although efficiency is an important criterion, its increase must not limit “the representation’s ability to capture essential problem-solving knowledge". There are three main categories for the knowledge representation systems: Rule-based, Model-based, and Case-based knowledge representation schemes (Chakrabarti et al., 2011; Helms, 2013). This thesis uses the first scheme but some aspects are applicable to these other representations as well.

Rule-based knowledge representation methods capture the design knowledge in IF-THEN rules. In a procedural way these rules transform an initial state to an altered situation. Continuous application of the rules transforms the initial design into a wide range of new designs. The expert knowledge is often encoded in the rules as heuristics for solving the problem. Chakrabarti et al. (2011) have also reviewed the advances in using these representation schemes in computational design synthesis research in the last decade. In this study a graph grammar approach – which is a rule based representation method – is used to represent the design knowledge, therefore other representation schemes are not further discussed.

1.3.3 Generative grammars

Rules are conditional statements, which consist of a condition and a consequence. The early expert systems used rules to formalize knowledge; such as MYCIN for diagnosing bacterial infections (Shortliffe et al., 1975) and DENDRAL for analyzing mass spectrographic data (Lindsay et al., 1993). Production systems are also a more specific kind of rule-based representation systems, which consists of a set of rules, in which the expert knowledge for problem-solving is encoded, a memory, which contains the current state of the data structure and can be represented as a string, a shape or a graph, and an inference engine to control the rule execution (Helms, 2013; Luger, 2008). To execute the action of a rule, its conditions must match the contents of the current state, which consequently changes the state of the memory. The execution is continued until there are no more rule conditions matched with the memory. One of the most important characteristics of the rules is that they "are a natural and intuitive way for representing heuristic knowledge" (Dym and Levitt, 1991) which is important in domains such as engineering design that rely on heuristics for solving their problems. As the rules are principally conditional sentences which are used in natural languages, human experts finds it easier to formulate their problem solving knowledge in rule-based formulations (Beierle and Kern-Isberner, 2008). However, grasping the reasoning logic for solving problems, which are complex and the rules are intertwined, can be very hard for human users (Rude, 1998). Therefore knowledge extraction and formulation from experts into rules can be a very tedious task. Grammars capture large design spaces in a single formalism, and hence can increase the design freedom (Alber and Rudolph, 2004). Generative grammars in general and graph grammars specifically have been used in many different domains to capture the design knowledge of complex engineering design rules (Antonsson and Cagan, 2001; Chakrabarti et al., 2011). They have been used in diverse areas such as general routing problems (Drumheller, 2002), network flow and structural topology optimization (Shea and Cagan, 1999).

1.3.4 Graph grammars

A graph grammar is a formal method to represent elements and their relationships in the design space (Cagan, 2001). Like natural languages, graph grammars are based on a vocabulary and a set of grammatical rules. By choosing a graph grammar for representing the design knowledge; the representation is fixed to graphs and the vocabulary elements are defined as nodes and edges. The initial design for a graph grammar is a seed graph, which is defined based on initial design requirements. Starting from this initial graph, grammars generate alternative design solutions based on a set of pre-defined rules (Chase, 2002). To define the grammar rules, rule validity conditions are typically encoded in the left-hand side (LHS) of a rule and rule modifications (changes in case of the rule recognition) are represented in right-hand side (RHS) of a rule as two graphs. Recognizing a rule, means that the graph in the LHS of a rule can be matched to a sub-graph in the working graph and its application means that this sub-graph is replaced with the graph in the RHS of the rule. After applying the rule a new graph is generated. For graphs, a graph grammar interpreter is required to apply a set of transformative operations on a seed graph. For this study, GraphSynth is used to accomplish graph transformations. GraphSynth is a unique research software for creating, editing, displaying, and manipulating generative grammars. This framework stores graphs, rules and rulesets under XML file format. It allows for the automatic search of creative, optimal or targeted solutions. GraphSynth is an open source framework built on Microsoft Visual Studio .NET. Additionally, it is able to perform various graph transformations such as the double-pushout method and free-arc embedding; these two together cover nearly all types of required graph transformations (Kurtoglu et al., 2010). One of the most important characteristics of the GraphSynth is its extensibility; through additional compiled on-the-fly functions nearly any capability can be added to the rules and rulesets.

1.4 Conclusions

In order to effectively utilize design information and knowledge, the design process is divided into three main phase: search, optimization and modification. Each phase may have multiple abstraction levels. To support the essential decision-making processes at each phase, the requirements for capturing and using design information and knowledge in various abstraction levels are defined. An adequate framework for capturing and using design knowledge is discussed based on various classes of knowledge levels proposed by Hicks et al. (2002). These include: general knowledge which is the most abstract level of knowledge, generic knowledge, specific knowledge, and case knowledge, which is the most concrete level of knowledge. These classifications are necessary to generate a framework that defines the applicability limits for each knowledge level. The work identifies that for more abstract levels (search phase) of design, higher level knowledge, such as general or generic knowledge are required and applicable. And for the optimization and modification phases other knowledge levels are applicable too. Following these formal definitions, the work introduces various knowledge representation schemes especially rule-based knowledge representation approaches. In chapter 2, 3 and 4 the proposed strategy in this introduction chapter is applied for three engineering design problems. The results of these design problems show, how the application of the proposed approach reduces the design space to a manageable size, in which only valid solutions are generated.

Ongoing research based on the proposed approach should focus on improving the search strategies for exploring the design space. Implementation of this step is important in order to achieve faster results in larger design spaces. Another important field of research is automatic capturing of generic and general levels of knowledge in the grammar rules and creating a data base of these captured rules. This will help designers to understand the real design problem at all abstraction levels easier. The framework is flexible enough and independent of the problem domain and type, therefore the approach can be used for other domains and other design problems.

2. A Novel Optimization Method of Fluid Channels Using Domain Knowledge

The aim of this chapter is to show the abilities of generative design systems, in achieving topology and shape optimization of fluid channels. By utilizing a multiple representation approach, there is no need for a large grid of variables to represent the topology, which causes significant computational savings and allows the simulation model to be independent. After evaluating and optimizing the generated graphs, they are transformed into meaningful 3D shapes to be simulated in a CFD solver. The effectiveness of the proposed method is checked by solving and comparing a variety of available test problems found in the literature. Furthermore by solving complex large scale problems (3-Dimensional), the robustness and effectiveness of the method is tested.

Keywords: Computational Design Synthesis, Design Automation, Graph Grammar, Computational Fluid Dynamics, Optimization.

2.1 Introduction

One of the most popular computational design synthesis approaches in engineering design involves shape and topology optimization methods, which is based on using finite element methods (FEM) for the analysis, and various gradient-based optimization techniques (Bendsøe and Sigmund, 2003). Topology optimization is a mathematical approach that models a fixed number of decision variables (cells or grids), and optimizes its objective function (e.g. part stiffness) for a given set of boundary conditions and loads. Numerical optimization methods have shown their efficiency in aiding the synthesis of engineering artifacts by generating many novel solutions (Bendsøe and Sigmund, 2003). Using topology optimization methods in solving channel fluid layouts has received a large amount of attention in recent years and various parameterizations have been suggested to solve Stokes (Guest and Prévost, 2006a) as well as Navier–Stokes problems (Evgrafov, 2006) with different Reynolds numbers (Duan et al., 2008; Gersborg-Hansen et al., 2005; Olesen et al., 2006; Zhou and Li, 2008). However, even very recent results by different scientists in the field (Challis and Guest, 2009; Jang et al., 2010; Liu et al., 2010) show that problems are mainly limited in complexity (number and direction of inlets and outlets), flow equation, and number of fluid types (if combination of fluids is not allowed). They are mainly 2D problems and the time to complete the process is exceptionally high, especially for solving complex 3D problems. Considering these limitations and the fact that demanding industrial problems are more complex, reveals a gap in capabilities and computational power of current methods. These limitations are mainly due to limited representation power. The synthesis process and design rules are dependent and integrated into the simulation model; the simulation model is often fixed for a given set of loads and boundary conditions. The simulation model is based on time-consuming numerical approaches such as matrix inversion and many iterations are required, which leads to slow convergence.

The aim of this chapter is to introduce a new perspective and show the abilities of generative design systems, such as graph grammars, in achieving design synthesis and optimization of fluid channels. The novelty of the proposed method is in its combination of generative design synthesis methods with conventional simulation models, leading to a significant reduction in the numerical costs. This method uses a graph grammar interpreter to generate different topological solutions for the fluid channel problem. Through exhaustive search of the design space all valid candidates are generated and evaluated. Based on the evaluation results, the candidates are sorted. Two optimization algorithms are then used to optimize all (or the top *n* based on computation limits) candidates of the sorted list. These optimization algorithms change the radius of fluid channels and the position of intermediate junctures, to minimize head loss. The candidates are again stored in a second list based on the new objective function values. Finally they are transformed into meaningful 3D shapes to be simulated in an adequate CFD solver. The nodes and arcs of the generated graph represent Constructive Solid Geometry (CSG) shapes.

The graph grammars rules work with graph elements to generate a new topological state, as a result the search and generation process is very fast. However, it is vitally important to embed enough information in the graph grammar rules in order to create precise 3D shapes. To increase the computational effectiveness of the generation process, the design process is carried out in different steps. To enter each step, the candidate solution must meet specific requirements such as maximum allowed compression of the fluid; otherwise it is filtered out. After passing the requirements of three such filters, which evaluate the validity of candidates in different stages of the synthesis process, information is added to the candidate solution.

By utilizing a multiple representation approach for the topology optimization of channels, our algorithm avoids many problems associated with other approaches in setting up the fluid equations. There is no need for a parameterization scheme because representing the topology is independent of the simulation model. It causes significant computational savings, because the CFD analyses and remeshing at each iteration is no longer required, which is a prohibitive in previous efforts (Zhou and Li, 2008). By using multiple representations in our method, dimension (e.g. 2 or 3) has almost no effect on the computation efforts in finding flow channel topologies which show the numerical efficiency of the proposed approach. Furthermore the need for postprocessing of the results is eliminated and a more accurate control over designing of solution topologies is provided. However after finding candidate design solutions, the transformation and CFD analysis of 3D results are computationally more costly. As the representation and simulation models are fully separated from each other, one can use the same rules for problems with completely different boundary conditions, fluid types, fluid directions and loads.

The proposed method produces results in agreement with previously solved power dissipation minimization problems for Stokes flow (Borrvall and Petersson, 2003; Challis and Guest, 2009; Guest and Prévost, 2006b; Liu et al., 2010). The effectiveness of the proposed method is checked not only by solving a variety of available test problems and comparing them with those found in the literature, the results of different complex problems with arbitrary flow directions in inlets and outlets shows the capabilities of the method in solving very complex large scale 3D problems. This chapter is organized as follows. Section 2 describes a background about topology optimization methods, generative design synthesis systems and our graph grammar approach. Section 3 provides details of the proposed approach in this chapter. Section 4 presents results and discusses the implications of results; the focus of this section is to present significant benefits of proposed methodology over previously used approaches. And finally, Section 5 concludes the study and suggests further research projects to extend the presented work.

2.2 Topology optimization

For more than two decades, engineering designers have used shape and topology optimization methods for a wide range of structural design problems. These optimization methods are now being used successfully by other areas such as electro-magnetics, MEMS and fluids as well (Bendsøe and Sigmund, 2003; Eschenauer and Olhoff, 2001). Borrvall and Petersson (2003) were the first to use topology optimization for solving Stokes flow fluid problems. Optimization of fluid channels is an essential topic in designing microfluidic devices (Andreasen et al., 2009; Vangelooven et al., 2010). It has application in diverse areas such as designing pipe bends for minimum head loss, diffusers, valves, interior air flow of vehicles, and engine intake ports. The goal is mainly to find an optimal topology for the fluid subdomains along with an optimal shape of channels that minimizes the power dissipated by the fluid (Liu et al., 2010). In order to use Stokes equations, as opposed to the full Navier-Stokes equations, the fluid flow is mainly assumed to be incompressible, steady, and slow. Topology optimization has been applied to solve Stokes flow problems on large scales (Aage et al., 2007), to design maximum permeability of material microstructures (Guest and Prévost, 2007), and in optimizing multifunctional materials; microstructures with maximum stiffness and fluid permeability (Guest and Prévost, 2006b). Details of using different approaches such as the level-set method or material distribution to increase the computational efficiency and the chance to find the global minimum can be found in the recent contribution of Challis and Guest (Challis and Guest, 2009). They describe methods which can avoid convergence of the algorithm to local minima (Aage et al., 2007; Borrvall and Petersson, 2003; Guest and Prévost, 2006b) and aim to overcome limitations of other models such as (Zhou and Li, 2008) with costly computational power for remeshing the whole domain. The chronological progress of results in the literature reveals significant improvements concerning minimizing required time and computation power, achieving global minima, smoothing the boundaries and using various Reynolds values for the flow.

2.3 Approach

The overall approach to the shape and topology optimization of fluid channels using generative graph grammars is depicted in the Fig. 1. The whole process can be divided into three phases; topology generation, transformation, and CFD evaluation. The topology generation phase also consists of three steps; search, optimization and detailed shape design. The separation of the topology generation from the evaluation phase enables the creation of topologies without considering the constitutive fluid equation or other issues related to the fluid representation. The topology generation phase uses the graph grammar interpreter to apply graph transformations and generating topologies. Three control parameters are used in the detailed shape design phase to define the final shape of the channel layout. The first control parameter is used to give a rough shape to the path of a channel in the layout. The second and third parameters are used to define the curvature at inlets and outlets respectively. For different problems, with different boundary conditions and fluid types, some experiments are required to tune these control parameters. In the transformation phase the generated topologies, which are represented as graphs are converted to 3D shapes. Finally in the evaluation phase OpenFOAM CFD solver (OpenCFD Ltd (ESI Group), 2013) and snappyHexMesh preprocessor are used to evaluate the 3D shapes regarding fluid dynamic criteria to select the top candidates. In the following sub-sections all three phases of the design are described in detail.

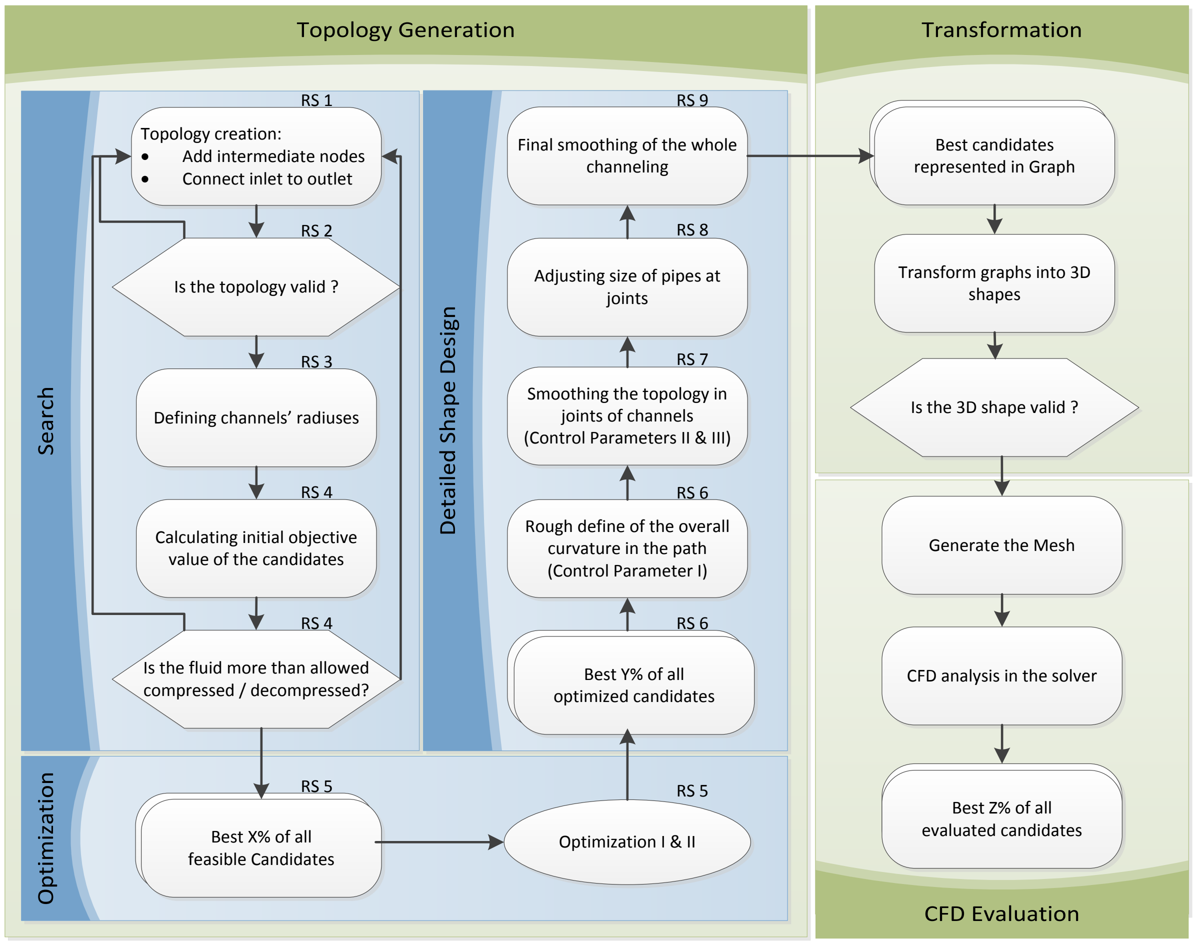

Topology Generation
Transformation
Search
Detailed Shape Design
Optimization
CFD Evaluation
RS 1
Topology creation:
• Add intermediate nodes
• Connect inlet to outlet
RS 2
Is the topology valid ?
RS 3
Defining channels' radiuses
RS 4
Calculating initial objective value of the candidates
RS 4
Is the fluid more than allowed compressed / decompressed?
RS 5
Best X% of all feasible Candidates
Optimization I & II
RS 5
RS 6
Best Y% of all optimized candidates
RS 6
Rough define of the overall curvature in the path (Control Parameter I)
RS 7
Smoothing the topology in joints of channels (Control Parameters II & III)
RS 8
Adjusting size of pipes at joints
RS 9
Final smoothing of the whole channeling
Best candidates represented in Graph
Transform graphs into 3D shapes
Is the 3D shape valid ?
Generate the Mesh
CFD analysis in the solver
Best Z% of all evaluated candidates

Figure 2-1: Approach for shape and topology optimization of fluid channels

2.3.1 Topology generation

The graph grammar interpreter receives a seed graph as input and delivers all valid topologies that can be generated for that graph. The generation (graph transformation) is carried out through twenty rules which are distributed into nine rulesets. A ruleset is a set of rules that transforms the design from one level of maturity to the next level. Rulesets are used as a means to compartmentalize different phases of the generation process. The first ruleset is responsible for generating all possible candidates – both valid and invalid – and expands the tree of the solution space to its maximum breadth (Fig. 2). Rulesets 2 to 9 change the shape (spatial position of graph elements and size of channels) of a candidate from a rough topology to a final solution. Depending upon the size of a candidate (number of channels), the number of rule applications on a candidate in each ruleset varies between one and many hundreds. Rulesets 3, 4 and 5 are assigned to calculate optimum size of channels and spatial position of channel bifurcations. While rulesets 6 to 9 use the three control parameters to give adequate curvature to channels. As can be seen in the Fig. 2, all invalid candidate designs are eliminated periodically to prevent time wasted in subsequent stages. In order to accomplish this, three trigger rules are used to check the completeness and validity of a design before transitioning to the next ruleset. All duplicate designs are also detected and filtered out in the ruleset 5. The whole approach is developed in such a way that incremental information, which is required in the next step, is added to the design. For instance in the topology generation phase, the topology is represented with graph elements node and arc, therefore the transformation operations are done much faster than if using 3D shapes.

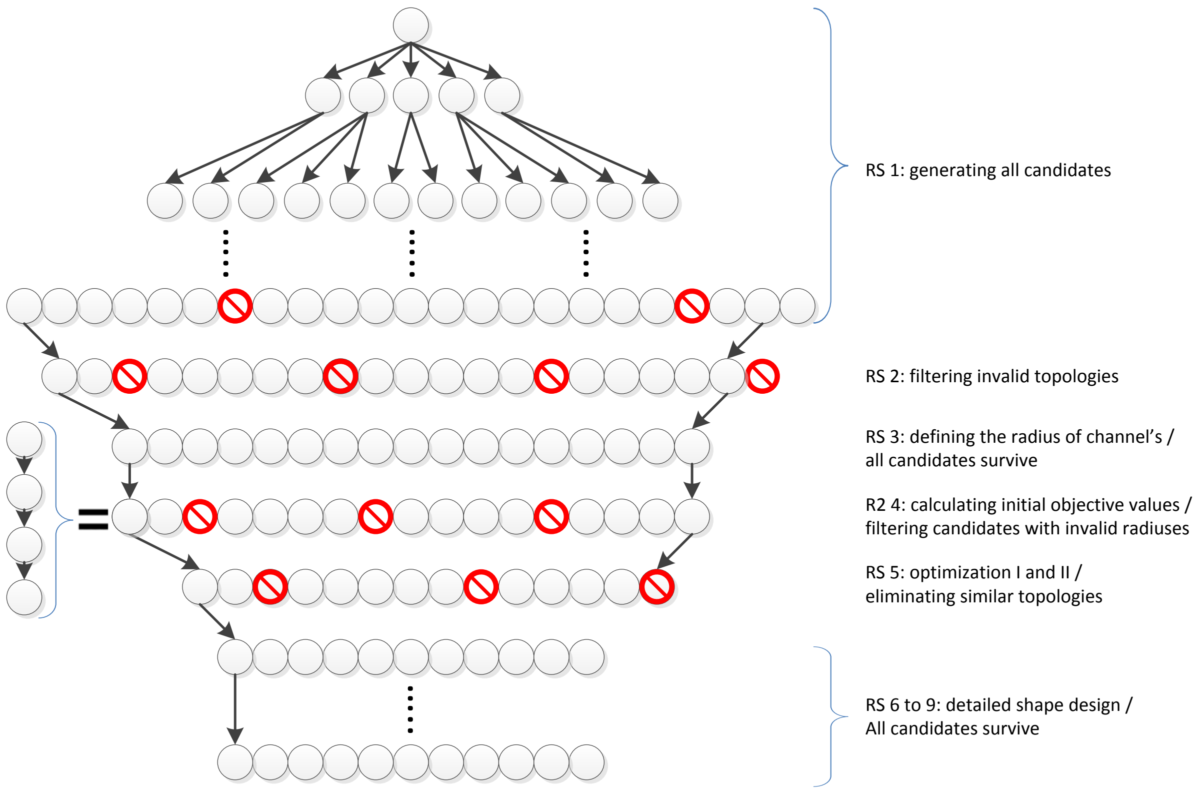

RS 1: generating all candidates
RS 2: filtering invalid topologies
RS 3: defining the radius of channel's / all candidates survive
R2 4: calculating initial objective values / filtering candidates with invalid radiuses
RS 5: optimization I and II / eliminating similar topologies
RS 6 to 9: detailed shape design / All candidates survive

Figure 2-2: Tree search for creating fluid channels

In Fig. 3 all twenty grammar rules with a short description of each are illustrated. The rules are created in a general way, so that for different types of fluid channel problems the same rules can be used. The only change which is required, if the simulation model (such as fluid type) is different, is adjusting the *control parameters*. The left picture in the Rule column is the left hand side of a rule (LHS) and the right picture is the RHS of the rule. The graph grammar interpreter converts that part of the seed graph which is matched to the LHS into the graph segment depicted in the RHS. Three rulesets (3, 4 and 8) change attributes of graph elements (for example add the radius to a channel section); therefore their LHS and RHS are not depicted. Figures 1 to 3 represent the approach from three different angles with different levels of detail.

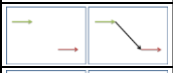
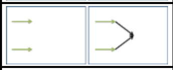
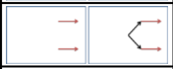
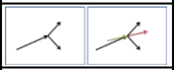
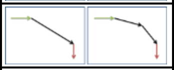
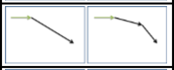
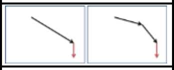
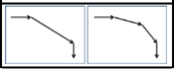
Ruleset		Rule	Description
1	1	Skeleton	Creates the skeleton of the node polygons
	2		Connect inlet to outlet (Maximum arcs to / from are limited)
	3		Insert intermediate inlet for two inlets (Maximum number is limited)
	4		Insert intermediate outlet for two outlets (Maximum number is limited)
	5	*Trigger rule 1*	*Minimum requirements are met?*
2	6	*Trigger rule 2*	*Is the topology valid ? (e.g. Inlets without outgoing or outlets without incoming arcs)*
3	7	Attribute	Calculate the radius of channels and adds it as an attribute to the arcs
	8	Attribute	Calculate radius of joints of flow and the value as an attribute to the nodes
4	9	Attribute	Initializing the size of complex channels
	10	Attribute	Initial calculating the first objective function
4	11	Attribute	Initial calculating the secodn objective function
	12	*Trigger rule 3*	*Is the fluid compression or decompression more than allowed?*
5	13	Optimization I	Optimize the size of channels
	14	Optimization II	Define position of intermediate inlets or outlets based on flow directions
	15		Define direction of flow in intermediate inlets or outlets
6	16		Roughly defines the curvature of each channel between an inlet and outlet
7	17		Fine smoothing of the topology at inlets
	18		Fine smoothing of the topology at outlets
8	19	Attribute	Adjust size of the channels at joints and adds it as an attribute to the arcs
9	20		Final smoothing of the whole channels

Figure 2-3: Grammar rules

Seed graph

A seed graph defines the scope and boundary conditions of the problem to be solved. In this case, it consists of some arcs and nodes which are labeled as inlet or outlet with different directions in 3D and different radii. Inlet 1 in Fig. 3 is represented by arc a0, which connects two nodes and its radius is 25. Fig. 3 illustrates a sample seed graph with three inlets and three outlets. The green (lighter) arrows in the shapes are the inlets and the red (darker) arrows are the outlets. The radius of the inlets and outlets can also be uniquely defined. The goal of grammar rules is to transform this seed graph to a graph that represents a meaningful channel layout.

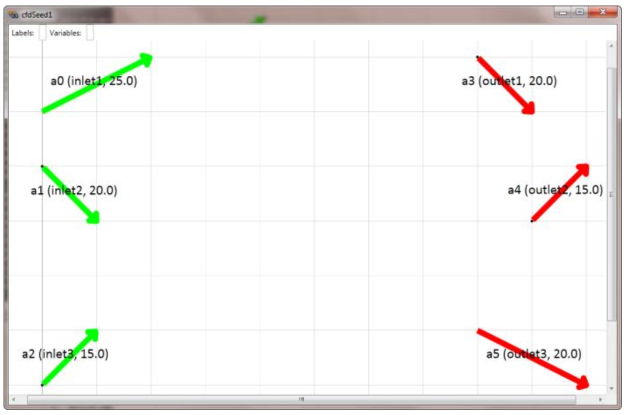


Figure 2-4: A seed graph with three inlets and outlets

Search

As illustrated in Fig. 1 the first step of the topology generation phase is an exhaustive depth first search (DFS) algorithm that creates all valid topological candidates for a given problem using the first four rulesets. The first four rules of the ruleset 1 are responsible for generating a topology. Aside from the depicted rule conditions in Fig. 3 (like connecting inlet to outlet, inserting intermediate inlet or outlet), many other additional functions are compiled into the rules to define detailed matching conditions as well as rule action. For instance, for rule 2, two functions aid in the recognition process; the first one prevents adding arcs that intersect other existing arcs in the design space and the second constraint function considers the maximum allowed spatial distance between an inlet and outlet. It prevents connecting inlets and outlets that are very far from each other in case of multiple connection possibilities. Rules 3 and 4 add an intermediate node for two inlets or outlets. For applying these rules, the distance between the inlets or the outlets and the direction flows at the inlets or outlets are considered. Rule 1 is used when facing channel layout problems with one or two inlet and many outlets or vice versa. It uses the position of inlets and outlets – as vertices of a simple convex or concave polygon – and creates the topological straight skeleton of the polygon. The Computational Geometry Algorithms Library (CGAL, 2013) has been used to find the straight skeletons. Aichholzer et al., (1996) introduced the concept of using the straight skeletons to represent simple polygons. Geometric skeletons like Medial Axis and the straight skeleton have been used in many applications such as Contour interpolation (Barequet et al., 2004), automatic shape synthesis and path planning (Eftekharian and Ilieş, 2012). The reason for this investigation is that – like rules 3 and 4 in our approach – it introduces intermediate juncture points between original ports to find the shortest possible spanning network. Fig. 5 shows a seed graph with one inlet and two outlets (a) and the topology which has been suggested through applying the first rule (b). This topology can also be reached by applying rules 3 and 2 respectively. Indeed, rules 2 to 4 can also generate results that rule 1 suggests. But rule 1 – especially when facing channel layouts with only one inlet or only one outlet – can give a near optimum channel topology through its single rule application.

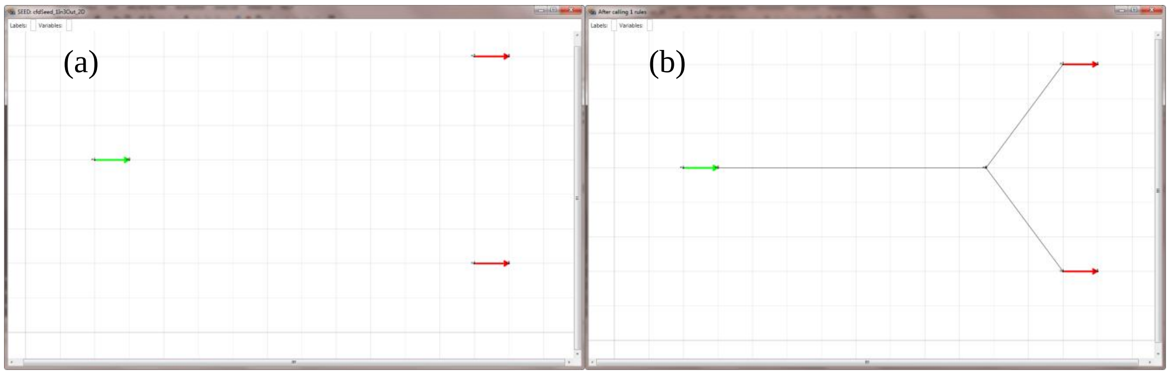


Figure 2-5: Using straight skeleton to find the channel layout

Rule 5 is called if the design has reached some degree of completeness. It is used to transition from using ruleset 1 into 2 after some degree of maturity is reached in the graph. It guarantees the creation of all valid candidates. For instance, without rule 5, Fig. 5 (b) won't be considered as a valid solution for the problem illustrated in Fig. 5 (a). Because there are still two rules which can be applied on the solution (connecting the inlet to either of the outlets through rule 2).

After a candidate transitions out of the first ruleset, the second ruleset checks the topological validity: 1) are all inlets have at least one outgoing arc? 2) do all outlets have at least one incoming arc? 3) and are intermediate inlets and outlets connected adequately to other graph segments? Many candidates are filtered out at this stage, which prevents many unnecessary simulations of invalid designs. Fig. 6 shows two candidates. Considering only topological criteria, the candidate at left is invalid and the right one is valid.

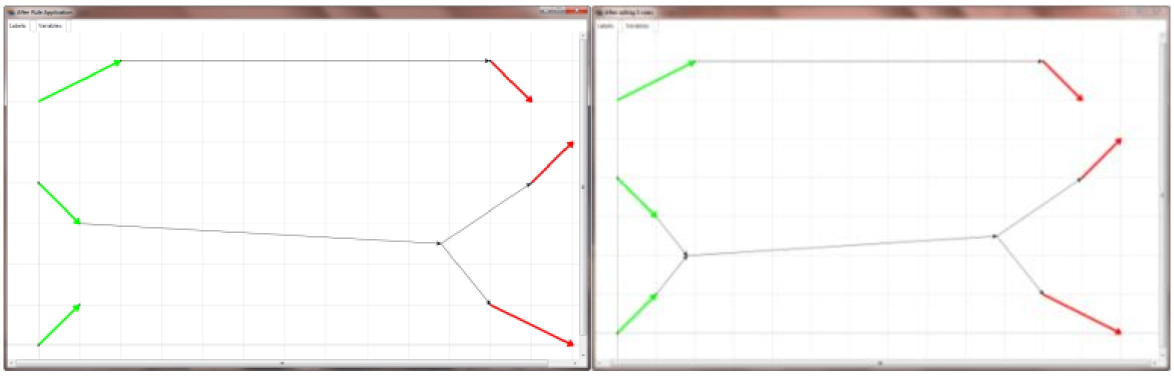


Figure 2-6: Two candidate topologies

As can be seen in Fig. 6 (b), the valid candidate has six arcs (channels). These arcs connect the inlets to the outlets but they are still not fully specified as they lack 3D dimensional information. The next step of the design process (ruleset 3) is to define the initial sizes of the channels. The size of a channel can be very tricky; in some cases knowing the inlet and outlet radius is enough to define the start and end radius of a channel like the arc that connects inlet 1 to outlet 1 in Fig. 6 (in Fig. 4 arcs are numbered).

Ruleset 4 is the final step in the search process. Rule 9 checks if all channels are initialized or not. In very complicated channel problems the rules 7 and 8 in ruleset 3 might not be able to calculate the size of channels, therefore rule 9 initializes them. Rules 10 and 11 are created to evaluate the candidates initially. They are evaluated based on the total length of the channels, amount of changes in the flow direction, and changes in the start and end radii of channels. These will be discussed in detail in the next sub section. Based on these initial evaluations, all candidates are sorted in a list to be further processed in the next step. This ruleset contains also the last important filter (trigger rule) for the validity check of candidates. It compares start and end radii (sizes) of channels. If the ratio is more or less than a desired one, the candidate will be rejected.

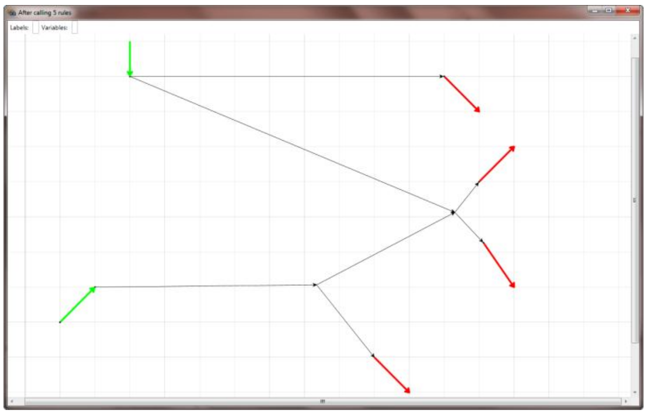


Figure 2-7: A candidate topology

For complicated fluid layout problems, in which many channel branches are joining together in a juncture, more complex computations and even an optimization algorithm is required to define the start and end size of channels. This is also carried out in the rule 13 of the ruleset 5. The objective function for the optimization is simply minimum difference between the start and end radii of each channel which does not require any CFD analysis; therefore the optimum channel sizing can be found very quickly. Another significant challenge in the developed approach is considering the direction of flow for each node. Direction affects the position of intermediate nodes. Therefore an optimization algorithm is developed (as a result of applying rule 14 of the ruleset 5) to find the optimum position of intermediate nodes. These two rules are not real grammar rules because they have no LHS matching; however they are implemented in the fifth ruleset to prevent unnecessary export and import operations. In the next sub section both optimizations are explained in detail.

Optimization

After storing all results of the exhaustive search in a sorted list, the best *n* candidates will be further optimized in the second step of the topology generation phase. Due to the smoothness and unimodality of the design spaces as well as the use of efficient optimization algorithms, we have been able to accommodate optimizing all candidates. More complex problems (problems with more than six ports) may require limiting them. However it is possible to set *n* to 500 without taking a prohibitive amount of time. For both optimizations, the Fletcher-Reeves gradient algorithm is used.

The primary goal of these optimizations is to minimize total head loss of the layout and – if required – prevent compression/decompression or velocity changes of fluid in channels. Head losses in closed channels and pipes include mostly three types of losses; head loss due to a) decelerating or accelerating of flow (Yamaguchi, 2008), b) friction between fluid and channel wall (friction loss), and c) pipe entrances, transition points, exits, and valves (minor losses) (Fay, 1994). The task of the first optimization is to reduce the first type of head loss, while reducing the friction losses and minor structural losses is assigned to the second optimization.

According to (Yamaguchi, 2008), the first type of the head loss between two segments of a channel is proportional to the changes in the cross sectional area of those segments. However, decelerating flow causes more head loss than accelerating flow. The first objective function is the total difference between: a) start and end area of each channel divided by its length, and b) incoming and outgoing channel areas to an intermediate joint. This affects also the compression or decompression of fluids in compressible fluids or the amount of velocity changes in non-compressible fluids (considering the principle of mass conservation). This optimization routine (rule 13) optimizes the size of channels in complicated layout problems where many channels intersect at a juncture. The objective is to minimize compression or decompression of the fluid (in compressible fluids) or large changes in the velocity of the fluid so as to decrease the pressure loss of the channel. The cross-sectional area of all incoming channels to a joint should be also similar to the area of all outgoing channels. For instance, in Fig. 6, the connecting channel of intermediate joint between inlets 2 and 3 to the intermediate joint between outlets 2 and 3 must have a start and an end radius of 25. The channel that connects inlet 1 to outlet 1 causes compression (head loss) of the fluid and thus increases in the velocity, because the start radius of the channel is 25 whereas its end radius is 20. Unless compression or decompression or velocity changes are desired, we will heed the heuristic to minimize the difference between the start and end radiuses.

There are various formulations for calculating the head loss due to friction. Darcy-Weisbach flow equation is one of the most useful head loss equations for closed flow conduits (Fay, 1994). It is proportional to the length of the channel (*L*), inverse of the diameter (*D*), density of the fluid (ρ) and the square of the average velocity (*V*) of the fluid in the channel (Eq. 1).

$$\mathrm{h}_f \propto \frac{L}{D}\,\frac{1}{2}\,\rho V^2 \tag{1}$$

Minor losses may also result from fittings that disturb the normal flow of the fluid in the channel. There are two basic methods to calculate the head loss due to changes in the flow direction: a) “equivalent length”, and b) “K factor”. Both of them are based on empirical test values for elbows, tees, valves and other various fittings. But there is a simple principle behind the results: the more the changes in the direction of the flow, the more the head loss. Indeed the fluid tries to move always in the same direction due to its momentum; hence by changing the direction of the channel, the flow will be pressed to the outer edge of the curvature, which causes more friction and consequently more head loss. Thus, the second objective function is the total: a) length of all channels in a graph, and b) the angle between incoming and outgoing flows in each joint. The higher these values, the more head loss of a layout. This optimization within rule 14 optimizes the position of the intermediate nodes, which were added initially by rules 1, 3 and 4.

Eq. 2 describes the first objective function. X1 (first radius of a channel segment) and X2 (second radius of a channel segment) are the variables of the function.

$$\mathrm{f(x)} = \sum\nolimits_{\mathrm{a=1}}^{\mathrm{A}} (R_a^1 - R_a^2) + \sum_{\mathrm{n=1}}^{\mathrm{N}} \left(\sum\nolimits_{\mathrm{i=1}}^{\mathrm{I}} R_i^{in} - \sum_{\mathrm{j=1}}^{\mathrm{J}} R_j^{out} \right)_{\mathrm{n}} \tag{2}$$

$\mathrm{X1}\colon R_a^1 = \text{first radius of the channel segment a}$

$\mathrm{X2}\colon R_a^2 = \text{second radius of the channel segment a}$

$\mathrm{A} = \text{number of arcs (channel segments)} \, in\ the\ layout$

$\mathrm{N} = \text{number of intersection nodes } (points\ ) \ in\ the\ layout$

$\mathrm{I} = \text{number of arcs (channel segments) } that\ enter\ to\ intersection\ point\ n$

$\text{J} = \text{number of arcs (channel segments)}\ that\ exit\ from\ intersection\ point\ n$

$R_i^{in} = \text{incoming radius of}\ channel\ \text{segment}\ i\ to\ the\ intersection\ n$

$R_j^{out} = \text{outgoing radius of}\ channel\ \text{segment}\ i\ to\ the\ intersection\ n$

Eq. 3 describes the second objective function. X1, X2 and X3 (spatial position of the intersection points) are the variables of the function.

$$\text{f(x)} = \sum_{\text{a}=1}^{\text{A}}(\text{L}_a) + \text{L}_{factor}\sum_{\text{n}=1}^{\text{N}}\sum_{\text{k}=1}^{\text{K}}(\text{Ave}_n^{dir} - \text{F}_n^k) \qquad (3)$$

$\text{X1: X}_n = \text{X position of intersection point n}$

$\text{X2: Y}_n = \text{Y position of intersection point n}$

$\text{X3: Z}_n = \text{Z position of intersection point n}$

$\text{A} = \text{number of arcs (channel segments)}\ in\ a\ layout$

$\text{N} = \text{number of intersection nodes}\ (intersecting\ points\ )\ in\ a\ layout$

$\text{K} = \text{number of all arcs (channel segments)}\ that\ enter\ to\ or\ exit\ from\ intersection\ n$

$\text{L}_a = \text{length of}\ channel\ \text{segment}\ a\ in\ the\ layout$

$\text{L}_{factor} = \text{conversion factor to find the equivalent length}$

$\text{Ave}_n^{dir} = \text{average direction of flow at intersection n}\ (\theta)$

$\text{F}_n^k = \text{direction of flow at channel segment k which enters to or exits from intersection}\ n\ (\theta)$

Through these two objective functions all parameters that affect the head loss of a layout, such as channel diameter, channel length, and position of joints are optimized. The velocity of the fluid is optimized indirectly through diameter (considering the principle of mass conservation). The velocity has also been considered in the second optimization through a weighting factor. If the initial velocity is too high, the weighting factor of the flow direction (third type of head loss) is increased to prevent sharp angles between incoming and outgoing flows. If the velocity is too slow, the length of the channels will be more important in defining the objective function. Other parameters such as Darcy friction factor, density, and gravity are not considered in these objective functions. This makes both optimizations independent from any fluid type.

Fig. 8 shows the effect of direction of flow at inlets and outlets on the position of intermediate nodes (objective function weighting factors are remained constant). Fig. 9 shows a candidate which has been suggested with the skeleton rule (a) and the result after the second optimization is shown in Fig. 9(b) (weighting factor of the third type of head loss due to slow velocity is very small). In this case the result is similar to the Steiner tree problem. The Steiner tree is the shortest tree that spans a given set of ports (Hwang et al., 1992). When the angle between flows is not considered at all, a Steiner tree should be the expected result from the optimization.

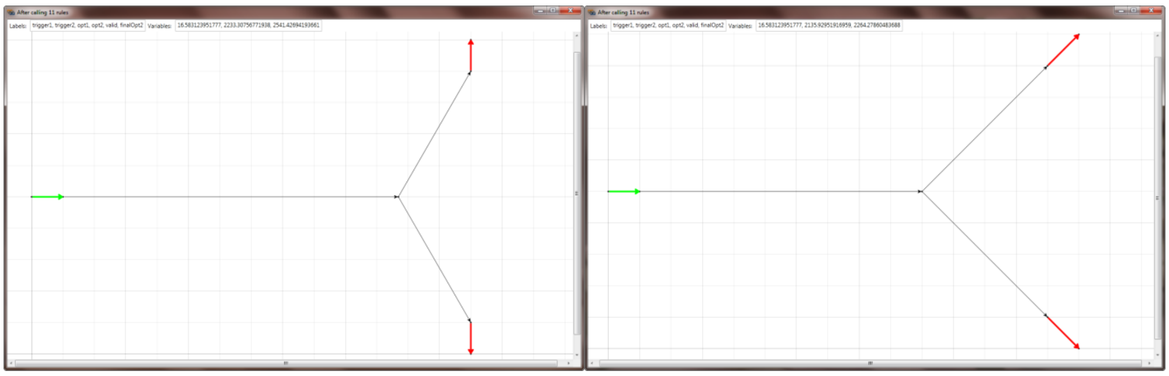

After calling 11 rules
Labels: trigger1, trigger2, opt1, opt2, valid, finalOpt2
Variables: 16.583123951777, 2233.30756771938, 2541.42694193661
After calling 11 rules
Labels: trigger1, trigger2, opt1, opt2, valid, finalOpt2
Variables: 16.583123951777, 2135.92951916959, 2264.27860483688

Figure 2-8: Effect of flow direction upon optimization ii results

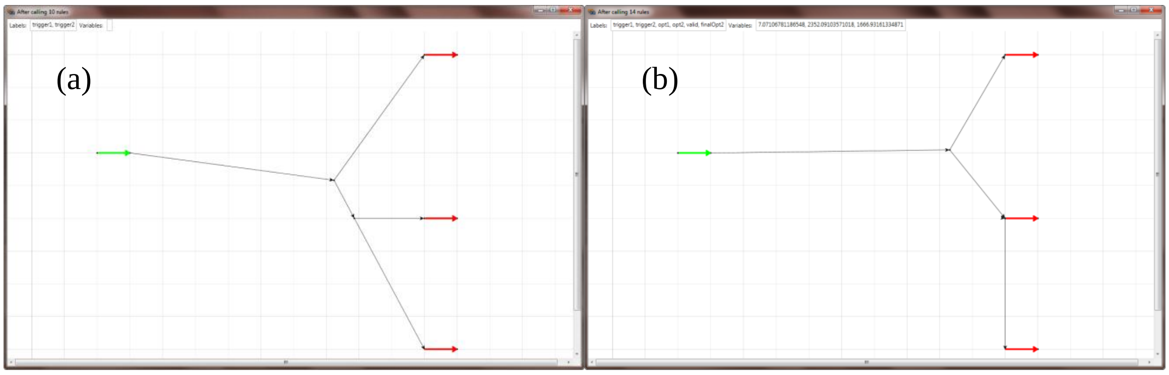

After calling 10 rules
Labels: trigger1, trigger2
Variables:
(a)
After calling 14 rules
Labels: trigger1, trigger2, opt1, opt2, valid, finalOpt2
Variables: 7.07106781186548, 2352.09103571018, 1666.93161334871
(b)

Figure 2-9: Not considering the flow direction gives the Steiner tree

After optimizing all candidates; the results are stored on a new list sorted by the first and second objective function values. A weighting factor is used to sum the objective values. Finally, rule 15 – last rule of the ruleset 5 – calculates the direction of flow at intermediate nodes, which is necessary for the next step; detailed shape design. This direction is calculated based on the direction of all incoming and outgoing flows to a juncture.

Detailed shape design

In the last step of the topology generation process, the shapes of the candidates are designed in detail. Rulesets 6, 7 and 9 are used to apply the shape changes. Ruleset 6 defines the overall curvature of a channel. Fig. 10 shows two designs with different overall curvature.

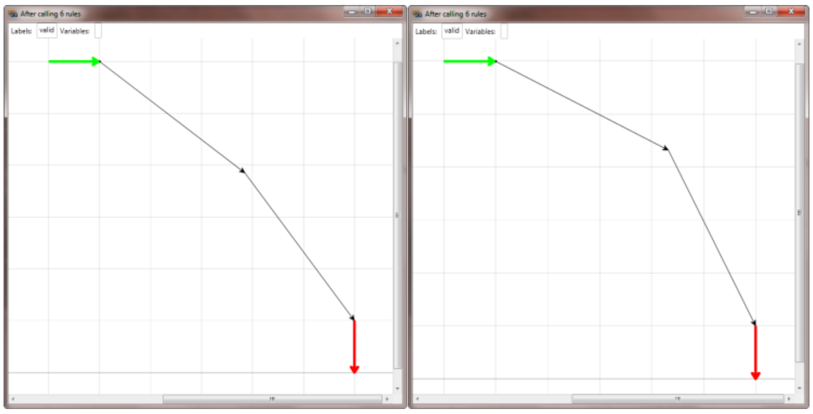


Figure 2-10: Defining the overall curvature

Rulesets 7 and 9 perform the final smoothing of the channels. Fig. 11 shows two channels which has been transformed through these two rulesets. These three rulesets (6, 7 and 9) use three control parameters. The first parameter defines the main curvature of the channels in ruleset 6. In Fig. 10, the effect of changing this parameter upon the curvature of the channel is shown. The minimum value for this parameter is zero which means no curvature. The second and third parameters are used in the ruleset 7 to define the curvature at inlets and outlets (Fig. 11). They are used to define the sharpness of the changes in the flow path at inlets and outlets. Defining the control parameters is depending upon many factors such as fluid type, fluid equation, and temperature. The designer should consider these factors when setting up the synthesis process for designing a new channel layout problem.

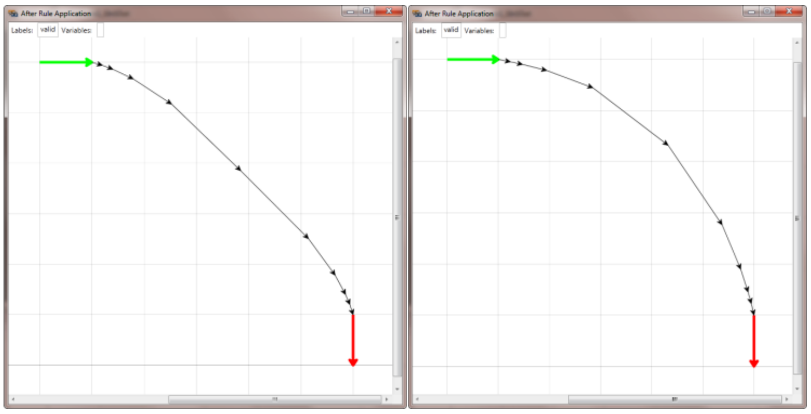


Figure 2-11: Smoothing the channel path

Rules 15 to 18 and 20 convert a simple topology like Fig. 6 to one like Fig. 12. This transformation has two aims; minimizing the head loss through adequate curving of the passageway and gradual changing of the channel radius. To each channel segment (arc) two start and end radii are assigned. The sum total of all these small differences of arcs in a channel is equal to the difference between start and end radius of that specific channel.

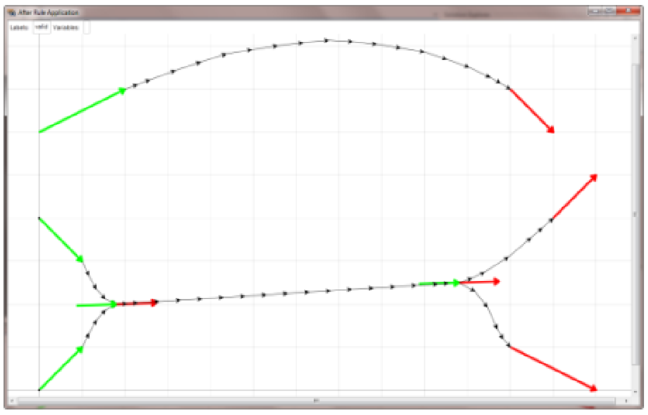


Figure 2-12: A candidate design after smoothing the shape

Finally ruleset 8 facilitates the connection of channels with a specific radius to intersections and joints which normally have different radii. This rule changes the radius of a channel segment, which connects to a joint, to the radius of the joint. For instance, if two channels in the Fig. 12 which join together have a radius of 20mm, the joint’s start radius should be around 28mm (considering the principle of mass conservation). Therefore those channel segments which connect the channels to the joint should have a start radius of 20mm and end radius of 28mm.

2.3.2 Transformation

After creating all possible topologies in the first phase, they are transformed to three dimensional shapes through a converter which uses the Parasolid geometric kernel (Siemens PLM Software Inc., 2013). The transformer converts nodes into spheres, and arcs into cones or cylinders. If the start and end radii of a channel is different, a cone is used for the transformation, and otherwise a cylinder is used. To increase the smoothness of the shapes, it is possible to reduce the minimum length of arcs in order to prevent sharp angles at joints and in different nodes. The shapes are saved as an STL file. Fig. 13 shows the candidate topology of Fig. 4 which has been converted to a 3D shape. For this specific design, the conversion took less than half a second. The converter saves all boundary conditions (inlet and outlet cross sections and the body of channels) separately. Fig. 13 has three inlets, three outlets and the addition of the body makes seven STL files. This separation of files is merely to facilitate translating the boundary conditions and the design for generating the finite element mesh and evaluating in a CFD solver. The inlet and outlet arcs (green and red arrows) are also converted to 3D shapes. These cylindrical boundary conditions stabilize the flow turbulence at the inlets.

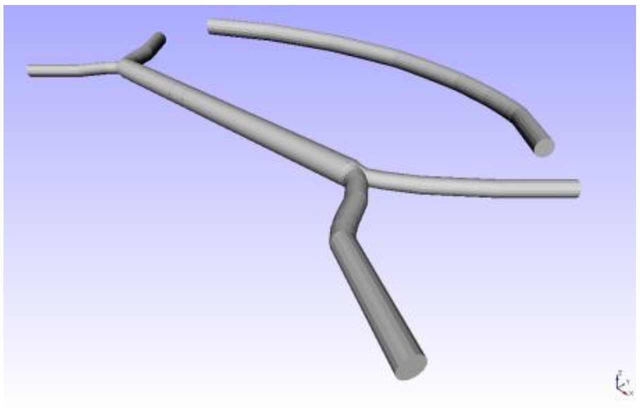


Figure 2-13: A converted topology into 3D shape

2.3.3 CFD Evaluation

For evaluating the performance of candidates, CFD simulation is accomplished. The last step of the design synthesis process is computationally the most expensive; however minimum number of candidates remains in this step. Through this simulation, the candidates with minimum head loss at outlets or any other desired criterion are recognized. For CFD simulation, OpenFOAM software is used. OpenFOAM is an open source CFD software that can be used for solving a variety of problems in engineering from complex fluid flows involving chemical reactions, turbulence and heat transfer, to solid dynamics and electromagnetics (OpenCFD Ltd (ESI Group), 2013). OpenFOAM includes tools for meshing – notably SnappyHexMesh – a parallelized mesher for complex CAD geometries. SnappyHexMesh generates 3D hexahedra meshes from a triangulated surface geometry in

STL format. In addition, it implicates more specific features, such as moving meshes, sliding grid, two-phase flow (Lagrange, VOF, Euler-Euler) and fluid-structure interaction (OpenCFD Ltd (ESI Group), 2013). OpenFOAM includes over 80 solver applications that simulate specific problems in engineering mechanics and over 170 utility applications that perform pre- and post-processing tasks, e.g. meshing, and data visualization (OpenCFD Ltd (ESI Group), 2013). After evaluating all candidates with the OpenFOAM solver, the best candidates will be selected as final solutions. The feedback from the user of this last step of the design is also necessary to tune the control parameters and also the weighting factors of the objective functions. Automating this step of the approach is still under development.

2.4 Results and Discussions

In this section, a few benchmark examples, which have been solved by other researchers, are discussed. This gives an insight upon the similarities and differences between the methods. The second part of this section is devoted to exploring the approach through some more sophisticated examples.

2.4.1 Benchmark Examples

There are three typical benchmark problems in the field of topology optimization of fluid channels which have been discussed by other scientists. Borrvall and Petersson, (2003) defined these problems in using topology optimization methods for channel layout design in 2003. Guest and Prevost (Guest and Prévost, 2006a), Challis and Guest, (2009), and Jang et al., (2010) are others who resolved all or some of these benchmark examples. Figure 14 represents these three test problems (Borrvall and Petersson, 2003).

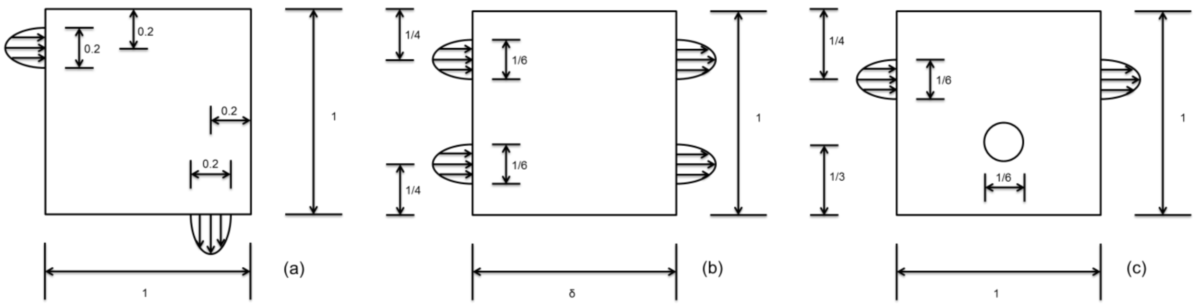


Figure 2-14: Design domain for the pipe bend example (a), design domain for the double pipe example (b), and design domain with a force term (c) (Borrvall and Petersson, 2003)

In Fig. 14 (b), the size of the design domain is variable. The design objective of these problems is to minimize the dissipated power in the fluid, subject to a fluid volume constraint (Borrvall and Petersson, 2003). Minimizing this objective reduces drag or pressure drop, which is vital in applications that require minimum head loss, such as bio-fluid mechanics, microfluidics and many other industrial processes. Time is an important secondary objective for these benchmark examples. Challis and Guest, (2009) give the precise time required for solving the examples with different approaches such as material distribution and level set method.

Achieved results of Borrvall and Petersson (2003) (Figures 7, 11 and 13 of the study) have been approved by other scientists (Challis and Guest, 2009; Guest and Prévost, 2006a; Jang et al., 2010), however with slightly different optimal objective values but significant changes in the required computational power and time. The registered time by Challis and Guest (2009) who have used a level set topology optimization method, is considered for comparison with results achieved with the developed method in this study. With a single core of a 2.0 GHz dual core AMD Opteron processor, 5 minutes is required for a two-dimensional pipe bend problem on a 100×100 element mesh and 44 minutes for a 200×200 element mesh (Challis and Guest, 2009). The results of the double pipe example for δ=1 on a 144×144 element mesh and for δ=1.5 on a 216×144 element mesh are 14 and 29 minutes respectively (Challis and Guest, 2009). These values increase dramatically when facing 3D problems. Fig. 10 of Challis and Guest (2009) shows the optimized 3D pipe bend on a mesh with 50×50×20 elements, which requires 3.35 hours. This shows that for real problems which are better modeled in 3D, the time is an important issue.

Due to significant differences between the developed multi-representations approach in this study and the aforementioned topology optimization methods, results of each representation are separately discussed. Fig. 15 shows all three representations of the developed approach for the first test problem: graph representation (a), 3D shape (b), and simulation model (c). For better visualization of the graph, the minimum length of an arc is increased; by increasing the number of arcs (decreasing minimum arc size) the path could be smoother. In the following paragraphs, the reasons behind all three representations are discussed.

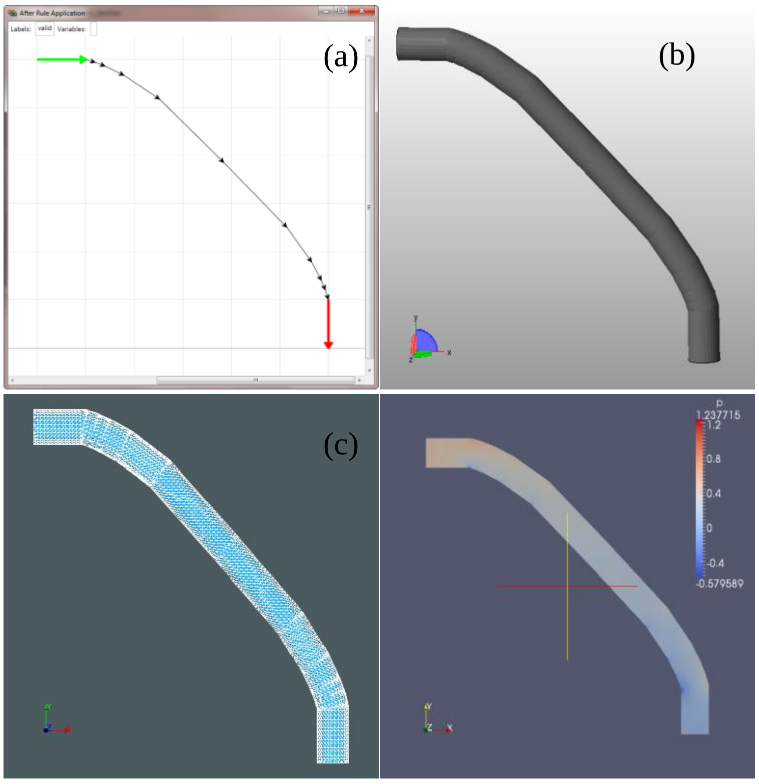


Figure 2-15: Three representations of a problem

The first representation (the graph) is used to create, edit, display, and manipulate the shape and topology of channels as described in detail in section 3. In this first level of representation, no trace of simulation model parameters such as fluid equation, Reynolds number, compressibility or non-compressibility, are used. However the designer could use the same rules with alternate control parameters so that optimal graphs would be created for different fluids and boundary conditions. Recall that changing the topology and shape of the channels can be accomplished in a fraction of a second.

Fig. 16 shows a resulting graph representation for each of the benchmark examples. These solutions have the same topology as those represented in (Borrvall and Petersson, 2003; Challis and Guest, 2009; Guest and Prévost, 2006a; Jang et al., 2010). For the double pipe example (Fig. 14 b), a few other topologies are suggested with this approach; the candidates with inferior performances are filtered out after optimizations I and II.

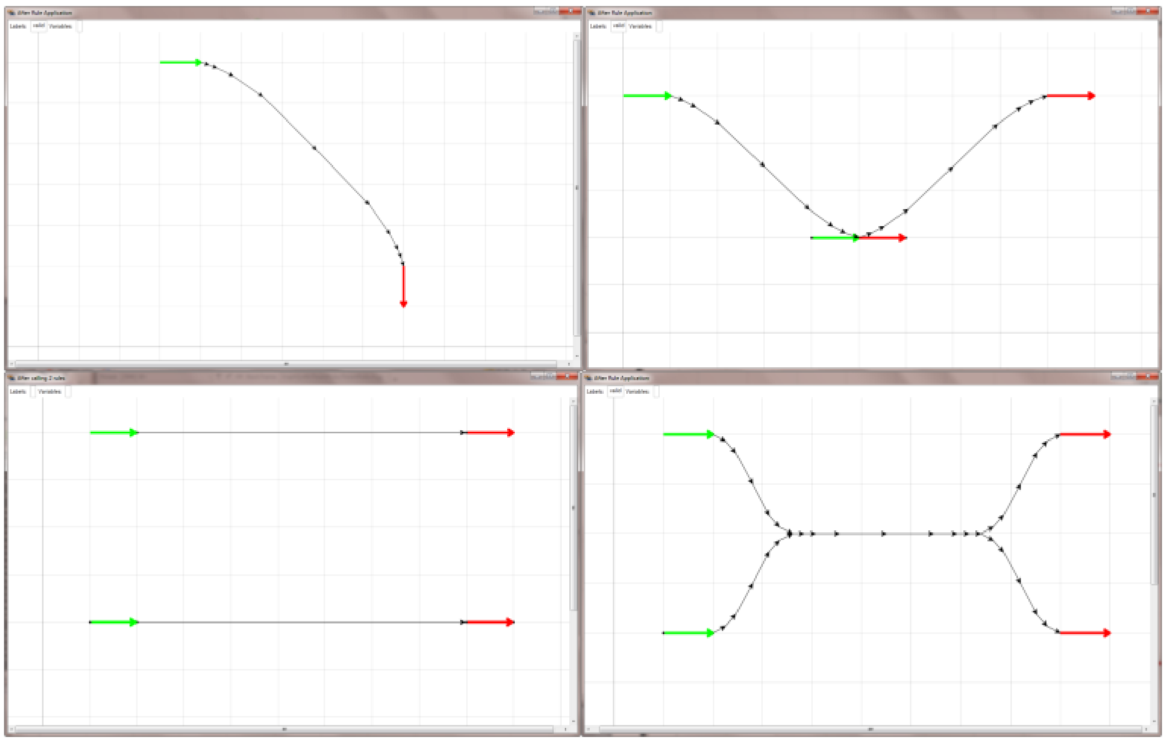


Figure 2-16: Topological representation of benchmark examples

The second representation is a 3D shape representation of the graph, where nodes have x, y, and z coordinates. It contains more information than the graph, but still not enough for the evaluation. Its second important task is to be used in other downstream applications without any postprocessing, which is normally required for grid based or level set topology optimization methods.

The third representation includes information about the fluid model, boundary conditions, loads, and the mesh. This information is used to evaluate the quality of generated topologies. A closer study of the benchmark examples reveals that, they have no or very little topological complexity. For instance, in case of first and third examples (Fig. 14a and b), there is only one topological variant, so the generation is done in very little time and the evaluation is also very fast (less than a minute). In case of the double pipe; there are less than ten different valid topologies possible. The first stage of the design (topology generation) requires less than a second to create the candidates. The transformation requires about three seconds and the evaluation phase requires about 60 seconds for meshing and evaluating each candidate. Again the most time-consuming part of the design is the evaluation. This shows that for small and middle size layout problems with moderate number of candidates (less than 1000) the topology optimization task is reduced to a straightforward single evaluation.

A single core of a virtual machine, installed on a computer with an Intel(R) Xeon(R) processor, is assigned to solve the benchmark examples. As the only 3D solution of these benchmark examples in the literature is for the pipe bend problem, it has been chosen for the comparison. However, the second and third examples can be compared like the first example. The creation of the topology in Fig. 15 and its conversion to a 3D shape needs less than 0.2 second. The evaluation was more time consuming as it required about 30 seconds for generating a Tetrahedron mesh with 27726 elements and evaluating it in OpenFOAM solver. Altogether 30.2 seconds time was required to reach the solution shown in Fig. 15. It is not adequate to compare this time with 3.35 hours for a 3D pipe bend in Fig. 10 of Challis and Guest (Challis and Guest, 2009), because to set the control parameters some trial and error are required. However, if one wants to optimize the three parameters, as only 30 seconds for each evaluation is required, the optimum shape could be obtained fairly quickly. The developed approach is able to handle very large scale 3D problems with arbitrary flow directions, high Reynolds number and different fluid types in the same layout design.

2.4.2 Layout design of a flow distributor

Fig. 17 shows the seed graph of a simple flow distributor with one inlet and five outlets. Distributors are used when uniform distribution of fluid is required (Liu and Li, 2013). This requires a similar head loss of the flow at all outlets, and not necessarily aimed at minimizing total head loss.

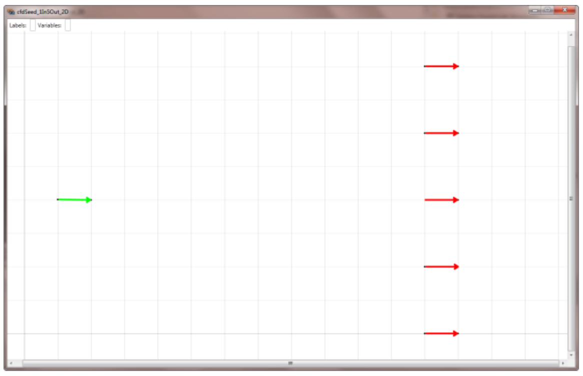


Figure 2-17: Seed graph of a channel problem with one inlet and five outlets

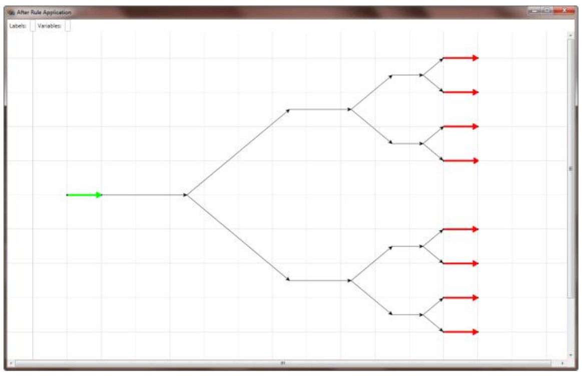


Figure 2-18: A distributor with the same head loss at each outlet

Fig. 18 shows such a design for a distributor with eight outlets. This design is created through eight rule applications: a modified version of rule 4 has been applied seven times and rule 1 has been called just once. The modified version of the rule 4 not only adds an intermediate node between two outlets, it adds a second node further behind (considering the flow direction) of the first intermediate node. Its aim is to stabilize the flow before reaching the juncture point to minimize unequal distribution of the fluid to outlets.

For this example, the aim is minimizing head loss; therefore the flow might be slightly different at different outlets. As illustrated in Fig. 1 the first step of the design synthesis is to search the design space for all valid candidates and store them in a sorted list. The search algorithm completed in 102 seconds in a search of the entire design space while found 1223 valid solutions. Fig. 19 shows six different candidates within the top 4% of all candidates based on initial evaluation. Although the shapes of all these candidates are different, many of them have the same topology. For instance candidates (a), (c) and (f) of Fig. 19 have exactly the same topology however with different shapes. The only way to find the similar topologies is after optimization II. This optimization changes the shape of the candidates and moves the position of the intermediate nodes to reach minimum head loss. At this stage the duplicates can be removed from the list of candidates.

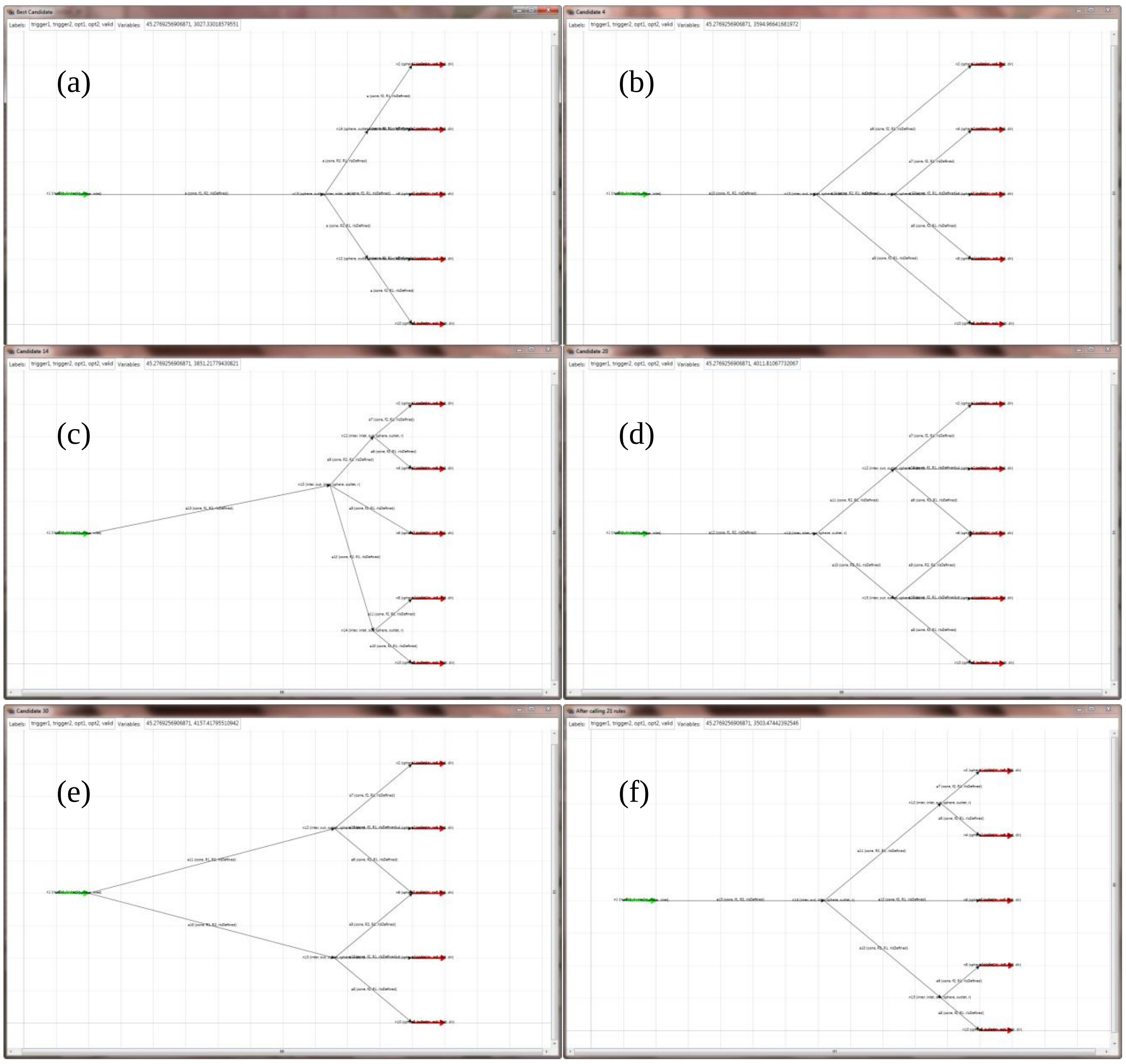
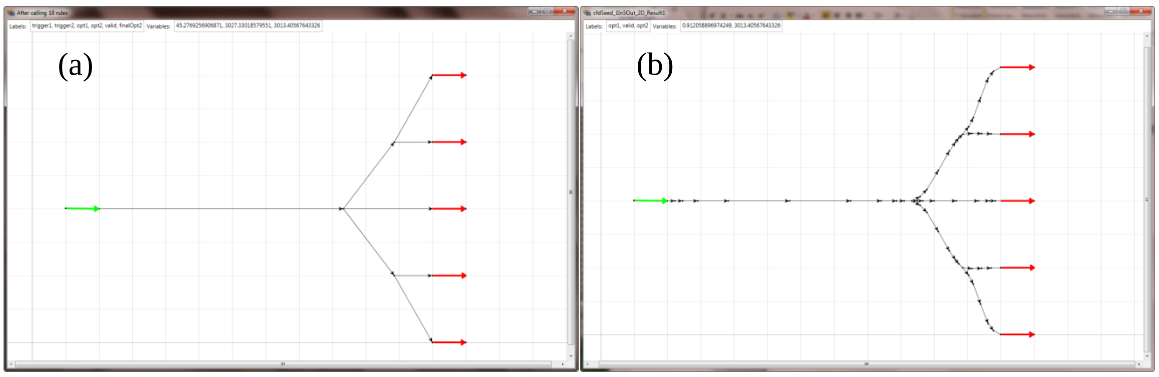


Figure 2-19: Six random candidates between the best 4% of all 1223 candidates

After storing all valid candidates in a sorted list, they must be optimized to find out candidates with best performance (defined as minimum head loss). The required time for both optimizations depends upon the number of arcs and intermediate nodes in the graph; it varies from a fraction of a second in most cases to a maximum of a few seconds. However it is not necessary to evaluate all candidates; often the best candidate is among the top candidates which has been initially evaluated and sorted in ruleset 4 (rules 10 and 11). Because rules 3 and 4 consider direction in their applications, therefore generated candidates are not very far from their optimized version. In Fig. 20 (a) the best candidate with the best objective function values is depicted. It is indeed the optimized result of (a), (c) and (f) in Fig. 19. It supports the idea regarding the appearance of the best candidate among top candidates listed in the search phase.

Up to this stage the topology of the candidates is fixed and the shape is fairly well-defined. In the third stage of the topology generation phase, the detailed shape design of candidates is accomplished. This stage is to further smooth the flow passage at joints in order to reduce the head loss due to sharp angle changes in the flow. This stage is not necessary for all applications because it creates very curvy design shapes. Although these shapes have less head loss, their production might be very tedious especially in large scale problems. For instance, a fuel cell distributor might not require such detailed shape but micro fluidic structures might be very sensitive to such shape refinements. Fig. 20 (b) shows the refined design of the best candidate.

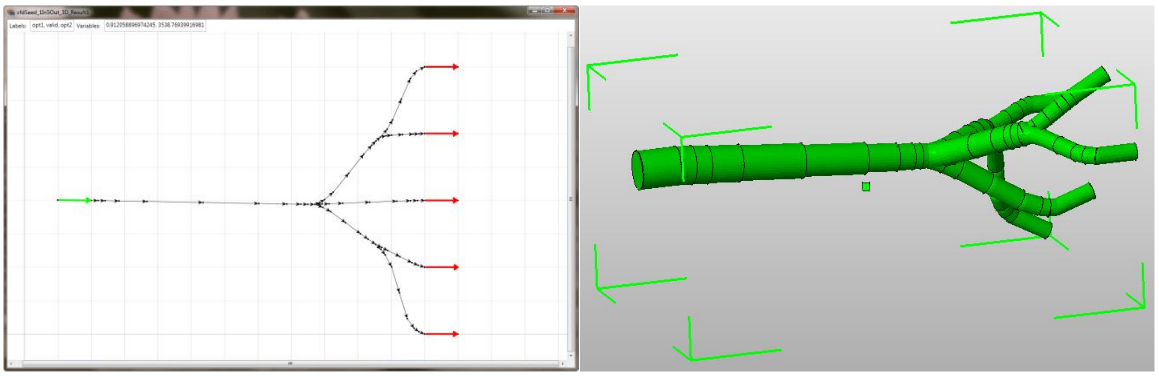


Labels: trigger1, trigger2, opt1, opt2, valid, finalOpt2 Variables: 45.2769256906871, 3027.33018579551, 3013.40567643326

Labels: opt1, valid, opt2 Variables: 0.912058896974249, 3013.40567643326

Figure 2-20: The best candidate after optimization ii (a), and detailed shape design (b)

Although the graphs are shown in 2D (x, y) they contain three dimensional data and all graph transformations are applied upon three dimensions (the graph visualization software, GraphSynth, only shows flat views of the graphs). For 3D visualization, these graphs are transformed into shapes via the Parasolid Kernel. Fig. 21 shows a flow distributor with outlets at different z positions. The graph of the Fig. 21 is different from that of Fig. 20 (b) because of the z position of the nodes.



Labels: opt1, valid, opt2 Variables: 0.912058896974245, 3538.76939916981

Figure 2-21: A 3D flow distributor

Essentially, the best design candidate is found during the first two steps of the topology generation phase. During these steps no CFD evaluation is performed to find the head loss of the channel designs, but three simple heuristics are used to reduce the head loss: length of channels, changes in the direction of flow, and changes in the radius of channels. The best candidate is one with the shortest total path length, with the minimum changes in the direction of flow, and with the minimum changes of the channel radii. Indeed after transforming the best or few best designs into 3D shape, they might be close to optimal even without CFD evaluation. Such a simplification depends on whether the control parameters and weighting factors of objective functions are adequately assigned.

Fig. 22 shows the pressure profile of the best candidate at the central slice (for better visualization), which has been calculated in a CFD solver and visualized in the Salome postprocessor (Open CASCADE, 2013). For the simulation, the flow is considered as a single phase steady state flow without turbulence. The density of the flow is the same as water (1000 kg/m^3) but with a very high viscosity (1 Pa.s). The gravity is not considered and the initial velocity at the inlet is 1 m/s. As can be seen in Fig. 22 the critical point of the design is at the base of the fork and the pressure at the middle outlet is higher than all others. The pressure for all other outlets is similar, but not the same because the objective function was to minimize the overall head loss.

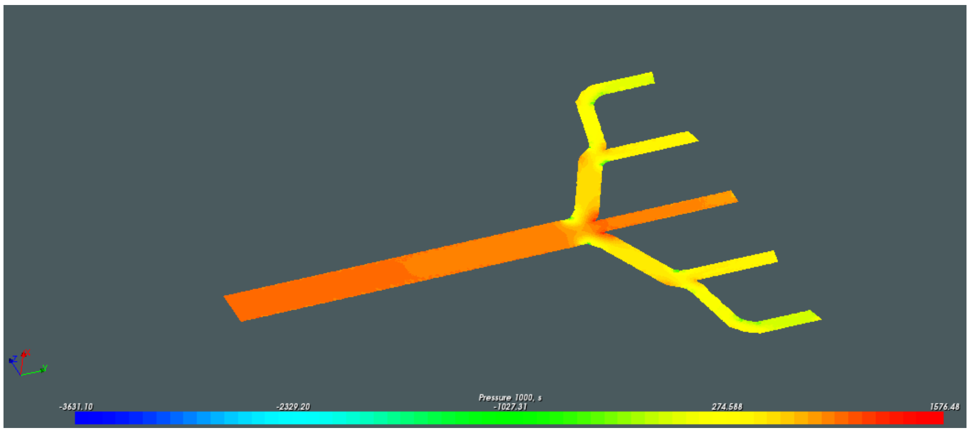


Figure 2-22: CFD evaluation results of the best candidate in figure 20

By changing such material properties and boundary conditions (e.g. speed), the control parameters and objective function weighting factors must be changed accordingly. For example, if the speed is too high all three control parameters must be increased to increase the curvature at inlets, outlets and the body of the channel. The weighting factor of the second part of the second optimization, which considered the angle between incoming and outgoing flow directions in a joint, must be increased. Because the sharper an angle and the faster the flow, the higher the head loss.

2.4.3 Complex channel layouts

In the previous section, to examine the design synthesis process a flow distributor layout problem with one inlet and five outlets was discussed. The approach is able to generate valid solutions for any kind of boundary condition with any number of inlets and outlets with arbitrary postures and directions. It is able to handle channel problems with very complex flow directions. Fig. 23 shows a candidate for a layout problem with two inlets and two outlets. As can be seen the flow directions are very unconventional, but the approach is able to smooth the shape of the channel very well.

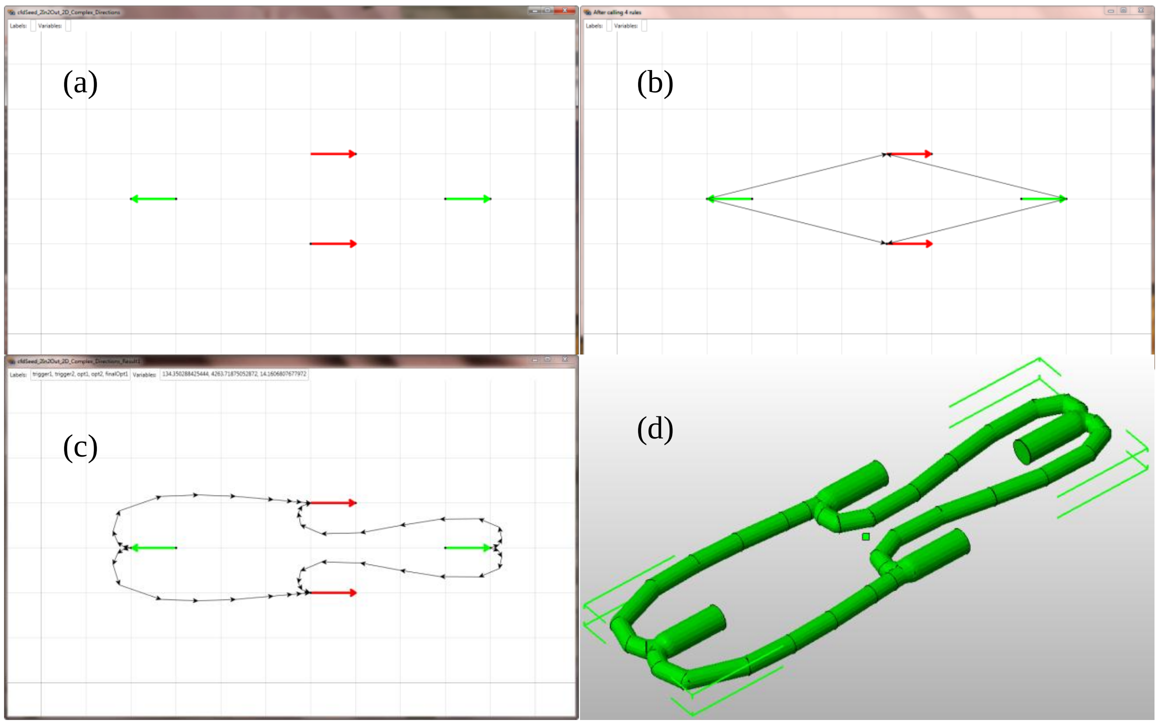


Figure 2-23: Seed (a), a candidate topology (b), graph representation (c), and 3d shape (d) of a layout problem with two inlets and two outlets

2.5 Conclusions

A new approach for shape and topology optimization of fluid channels using generative design methods is presented. This multiple representation approach uses graphs to represent both the topology and the shape of channel layouts. This allows a very fast generation of topological solutions for a given design problem. Based on results of two optimization functions, the best solutions are stored in a list for further detailed shape design. To evaluate solutions with a CFD solver, the graphs are converted to 3D shapes via Parasolid. These shapes can be used directly in downstream applications with no additional postprocessing. The simulation model is fully separated; therefore it is possible to solve problems such as that have compressible fluids with high Reynolds number and arbitrary flow directions at inlets and outlets. Large scale problems, problems with more than one fluid type, for which the mixing is to be avoided, are also solvable. The dual objective function allows designers to reach desired compression, decompression, and velocity of flow at each outlet while simultaneously minimizing the head loss. The rules may be flexible enough and independent of the simulation model that the approach might be used to create channels for other domains such as heat transfer to transfer maximum heat from sources to coolers.

The ongoing research of this study is on automating the CFD analysis into the search loop of the design synthesis approach. Implementation of this step is necessary to have an automated tuning of the control parameters of the control parameters. Using B-Splines and loft function instead of CSG primitives to convert graphs into 3D shapes is another possible research area, which would yield smoother channels. It also makes possible the creation of channels with non-circular cross sections. Another important field of research that can increase the generality of the approach is handle obstacles in the seed graph. The reason for this investigation is that obstacles are an undeniable part of the real world design problems. An interesting field of research might be to use the output results of the approach as input for conventional topology optimization methods. Due to a good initial design, convergence can be faster and many problems might be solved that are hitherto not solvable.

3. Truss Layout Optimization using Generative Design Synthesis Approach

The aim of this chapter is to demonstrate the abilities of generative design systems in achieving structural layout optimization. The combination of generative design synthesis methods with conventional simulation models produces a design technique to achieve optimal topologies and shapes for cable trusses considering various constraints such as stress, displacement, stability. Furthermore, manufacturing issues and material imperfections and limitations can be considered in the synthesis. The effectiveness and robustness of the proposed method is checked by solving a variety of available test problems found in the literature. The results show that the approach not only creates the existing solutions for the test problems, it creates new structures that have never been seen before.

Keywords: Topology Optimization, Design Automation, Graph Grammar, Truss.

3.1 Introduction

One of the most popular computational design synthesis approaches in engineering design involves topology optimization methods, which is based on using finite element methods (FEM) for the analysis, and various gradient-based optimization techniques (Bendsøe and Sigmund, 2003). Topology optimization is a mathematical approach that models a given fixed number of decision variables (cells or grids), and optimizes its objective function (material layout) for a given set of boundary conditions and loads. These optimization methods are now being used successfully in areas such as electro-magnetics, MEMS and fluids as well (Bendsøe and Sigmund, 2003; Eschenauer and Olhoff, 2001). For more than two decades, engineering designers have used topology optimization methods for a wide range of structural design problems. The objective of structural optimization is to improve the performance of the structure components in terms of material efficiency in transferring applied loads. Therefore, the performance criterion is usually the weight or cost of the structure subject to geometrical constraints and various performance-based constraints such as stress, displacement, mean compliance, frequency and buckling load.

Truss and space frames are widely used structures, because they are simple and inexpensive to build and can be used in many engineering applications. Literature shows much research based on classical topology optimization methods for the optimal design of truss structures (Dorn et al., 1964; Kirsch, 1989; Luh and Lin, 2011). Typical truss topology optimization approaches discretize the design space with a nodal mesh of a large ground structure, in which every node is connected to almost every other node in the domain. The ground structure concept has been first initiated by Dorn et al. (1964). This dense set of potential structural members along with applied loads and boundary conditions are assumed known. The optimization is used to determine the material distribution of cross-sectional areas of the connections. By removing inefficient members with slender areas below a certain threshold the connectivity of the system is changed and the structure is updated (Achtziger et al., 1992; Bendsøe and Sigmund, 2003; Bendsøe et al., 1994; Kirsch, 1990). Truss topology optimization is a combination of three optimization problems; size, topology and shape. The objective of size optimization is to find the optimal cross-sectional area of structural elements. Topology optimization aims to find the optimum existence and connectivity of the nodes. And shape optimization is concerned with finding the optimum nodal coordinates. For each optimization it is assumed that the variables of the other two optimizations are fixed.

Numerical challenges associated with the formulation of underlying governing mechanics (e.g. local and global instability) are often an important obstacle in topology optimization methods (Jalalpour et al., 2011). However, the matter becomes more acute when considering uncertainties associated with the structural stiffness such as geometry and material property imperfections. Consequently most of the researches in this area are focused mainly on deterministic problems with a limited consideration of uncertainty (Bendsøe et al., 1994; Díaz and Bendsøe, 1992; Lógó, 2007; Lógó et al., 2009; Yonekura and Kanno, 2010). The main strategy to consider these uncertainties in the formulations has been adding randomness (uncertainty) to the spatial position of the nodes (Asadpoure et al., 2011; Calafiore and Dabbene, 2008; Guest and Igusa, 2008; Sandgren and Cameron, 2002) or equivalent random forces at nodes (Jalalpour et al., 2011; Tyas et al., 2006). Jalalpour et al. (2011) aim to be the first who propose a method that is capable of handling both nodal location uncertainties and first order global buckling effects. In their proposed method, random forces at nodal points represent the potential global buckling in imperfect structures. Although considering uncertainty in the spatial location of the nodes creates (theoretically) more stable results, they are less practical structural solutions to be built. Indeed, since 1960, various optimization methods for the layout design of structures have been developed and many papers and books on the mathematical aspects of the structural optimization have been published. These contributions are mainly concerned with theoretical aspects rather than practical applications and engineering aspects (Liang, 2005). This shows a clear gap between the development of structural layout optimization theory and its practical applications in industry (Cohn and Dinovitzer, 1994; Liang, 2005). The main reasons behind this gap are the mathematical complexity of structural optimization methods (Liang, 2005), and the fact that structural optimization techniques are developed primarily for saving materials and not for automating the engineering design process (Liang, 2001). The work presented in this chapter introduces an efficient design tool for mechanical and civil engineering researchers. It is a clear and easy to understand concept and the methodology is an attempt to reduce the existing gap between academic methods for structural layout optimization and practical applications.

This chapter presents the theory and application of a generative design synthesis method for topology, shape and sizing design of structures. The method incorporates the load flow principal, simulation and optimization methods through generative design synthesis approach into a modern structural layout optimization theory. Furthermore, unlike other conventional methods, the types of bars and cables that are allowed to be used as components of the structure are defined at the beginning of the synthesis. There is no need to post-process the results, because manufacturing issues and limitations can be considered in the synthesis. This method uses a graph grammar interpreter to generate different topological solutions for a structural problem. Through exhaustive search of the design space all valid topologies for a given problem are generated and sorted based on the complexity of the solutions. An optimization algorithm is then used to optimize all (or the top *n*) topologies. This optimization algorithm changes the spatial positions of the joints, to minimize a desired objective such as stress, displacement, or overall load flow. Finally the best candidates are transformed into meaningful 3D shapes. The nodes and arcs of the generated graph represent Constructive Solid Geometry (CSG) shapes. The graph grammars rules work with graph elements to generate a new topological state, as a result the search and generation process is very fast. However, it is vitally important to embed enough information in the graph grammar rules in order to create meaningful structures. To increase the computational effectiveness of the generation process, the design process is carried out in distinct steps. To enter each step, the candidate solution must meet specific requirements.

One of the major limitations, which topology optimization methods in conceptual design are facing, is limited representation power; the synthesis process and design rules are dependent and integrated into the simulation model, the simulation model is often fixed for a given set of loads and boundary conditions. By utilizing a multiple representation approach for the topology optimization of structures, our algorithm avoids many problems associated with other approaches in setting up the mechanical behavior equations. There is no need for a parameterization scheme because representing the topology is independent of the simulation model. It causes significant computational savings, because the FE analyses and remeshing at each iteration is no longer required. By using multiple representations in our method, dimension (e.g. 2D or 3D) has almost no effect on the computation efforts in finding structure topologies. Furthermore, as the representation and simulation models are fully separated from each other, one can use the same rules for problems with completely different boundary conditions, loads and structural component types.

The proposed method not only produces results in agreement with previously solved problems, it creates new structures that have never been seen before. The effectiveness of the proposed method is checked by solving a variety of available test problems and comparing them with those found in the literature. This chapter is organized as follows. Section 2 describes a background about generative design synthesis systems and load flow principal. Section 3 provides details of the proposed approach in this chapter. Section 4 presents results and discusses the implications of results; the focus of this section is to present significant benefits of proposed methodology over previously used approaches. And finally, Section 5 concludes the study and suggests further research projects to extend the presented work.

3.2 Load flow path principal

Load flow path is a way in which load paths through a structure or mechanism from an input point (point of application) to the output point (support or fixed point). The term load path has been first defined by (Kelly and Elsley, 1995), although various authors (Hart-Smith, 1995; Kermode, 1964; Osgood, 1970) have used this term –however– in a descriptive sense. A fundamental fact in design process of structures is ensuring an appropriate path for loads and forces to flow in the structure from application input point(s) to the fixed or reaction output point(s). Although this has been always implicitly considered, there is a direct relation between load flow and the deformation behavior of the structure. Hence an insight obtained from load flow path can greatly enhance the design process (Kelly and Elsley, 1995; Skakoon, 2008). Load paths are relatively easy to define in simple structures such as trusses which carry only axial loads. The concept of load path has been used in topology optimization methods to reach feasible solutions (Harasaki and Arora, 2002, 2001; Hoshino et al., 2003). The main limitation of these techniques is the mathematical formulation of load paths which is based on finite element analysis of previous design step (Marhadi and Venkataraman, 2009). In this chapter the knowledge obtained from load flow path principal bases a new design methodology for structural topology optimization. This methodology has three main rules;

- forces tend to flow in paths with least resistance; here, resistance is the amount of change in the direction of the load flow path,
- changes in the direction of the load flow requires an extra transmitter member to impose the change,
- and finally the more parallel the structural components are with the direction of the applied load at a particular joint, the more their utilization. For instance if at a joint, the applied load is exactly at the same direction as a specific member, the utilization of that member is maximum, because no other force components are created.

A grammar-based approach for truss topology generation and optimization has been first proposed by (Reddy and Cagan, 1995). Shea further developed the approach for the synthesis of truss structures using finite element simulation and stochastic search methods called “shape annealing” (Shea and Cagan, 1999, 1998; Shea, 1997; Shea et al., 1997). eifForm is the software tool which has been developed based on this approach (Shea et al., 2005). Shape annealing rules are simple random rules (Cagan, 2001) but our main aim in this chapter is to systematically calculate the load paths based on the rules and generate various topologies that can meet the requirements for the desired force path. This will help us to add only those components which increase the overall performance of the structure. Therefore, stresses and strain energy will be uniformly distributed throughout the topology.

3.3 Approach

The overall approach to the structure synthesis using generative graph grammars is depicted in the Fig. 1. The whole process can be divided into two phases; shape and topology synthesis, and transformation. The synthesis phase consists of two steps; search and optimization. In the search phase, all valid topologies are generated in six consecutive steps and in the optimization phase the parameters of the topology are optimized. Based on the objective function values, an optimum candidate is selected. The synthesis phase uses the graph grammar interpreter to apply graph transformations and generating topologies.

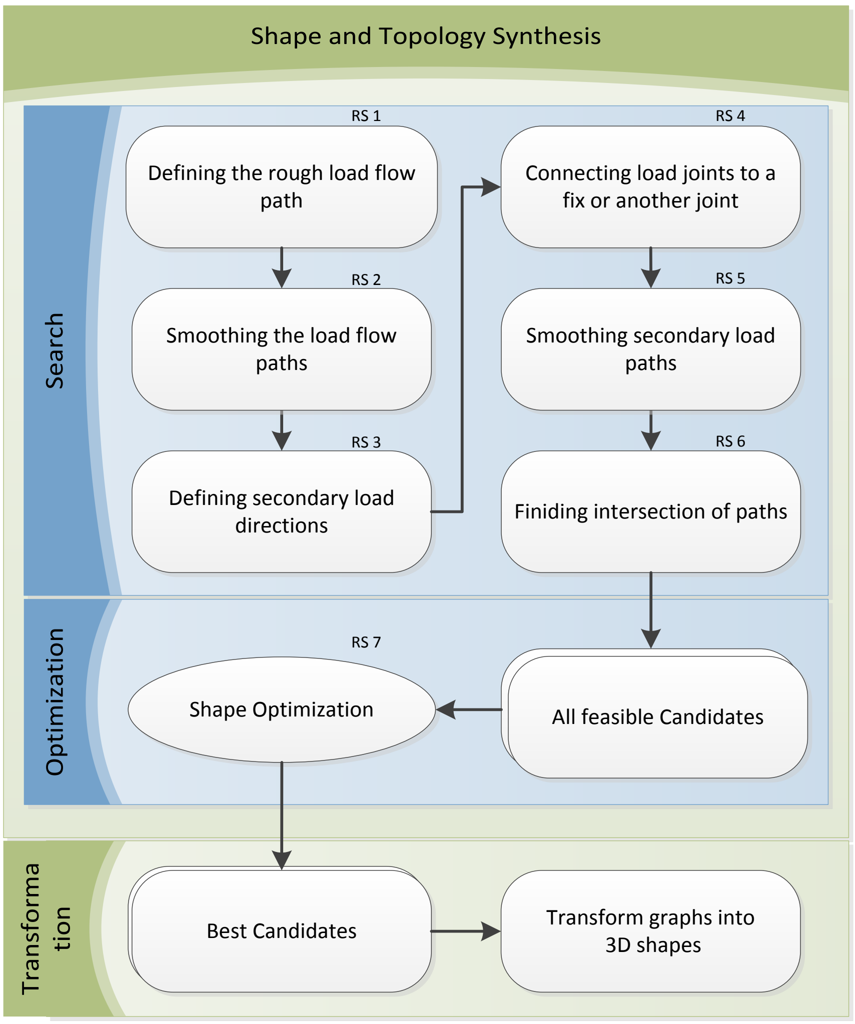

Shape and Topology Synthesis
Search
Optimization
Transforma tion
RS 1
Defining the rough load flow path
RS 2
Smoothing the load flow paths
RS 3
Defining secondary load directions
RS 4
Connecting load joints to a fix or another joint
RS 5
Smoothing secondary load paths
RS 6
Finiding intersection of paths
RS 7
Shape Optimization
All feasible Candidates
Best Candidates
Transform graphs into 3D shapes

Figure 3-1: Structure synthesis approach

In the transformation phase the generated topologies, which are represented as graphs are converted to three dimensional (3D) shapes. This step is not necessary for the synthesis of structures and is used just to visualize the final results as 3D shapes. In the following sub-sections all phases of the design are described in detail.

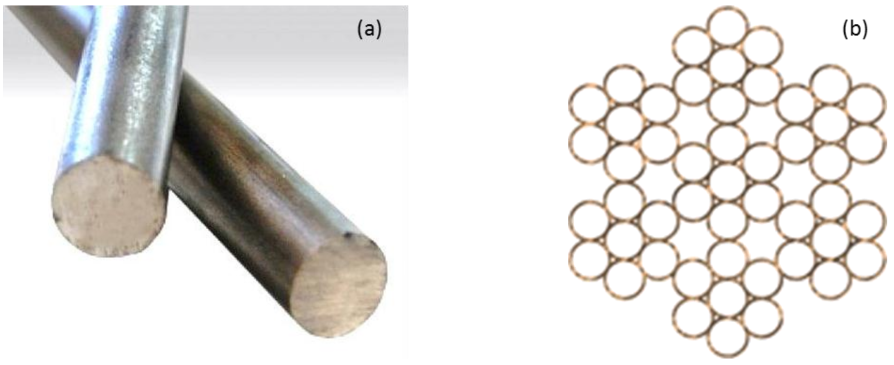
3.3.1 Analysis of the structures

This section briefly describes how the necessary structural evaluations are carried out. The approach is initially developed for cable truss synthesis, but it can be used without any modification for synthesis of truss structures. The only difference between cable truss and truss structures is in the individual components that constitute the structures; trusses constitute of tension-compression bars, whereas cable truss structures consist of tension wire ropes and compression bars.

Structure elements

A cable truss structure is a mechanical system composed of tension wire ropes and compression bars, which are connected together through frictionless hinges (nodes), and is loaded only at nodes. Consequently the axial displacement in any individual component (wire rope or bar) is linear and consequently the internal forces, strains and stresses are constant for each component in the structure. In this work the simulation does not consider the weight of the components. But since the simulation and topology generation are separate from each other, the simulation algorithm can be changed.

Fig. 2 shows the shape of the bars and the construction type of the wire ropes that have been used in this study.



$Shape = Cylindrical\ cross\ section$
$Diameter = 40\ mm$
$Young's\ Modulus = 210\ N/mm^2$

$Ultimate\ Tensile\ Strength = 1770\ N/mm^2$
$Breaking\ load = 2540\ N$
$Diameter = 2\ mm$
Rope $Construction\ Type = 6 \times 7(1+6) + 1 \times 7(1+6)$

Figure 3-2: cylindrical bar (a), and wire rope (b) used for this study

Size, shape, material characteristics and construction type of the elements (bars and ropes) can be easily changed at the onset of this approach. In the results section of this chapter, the effect of changing elements features is investigated. It is also possible to define a set of options for the structure elements, and leave the decision for choosing adequate element types and sizes for each segment to the optimization. In the optimization section (3.2.4) and also the first two sub-sections of the results the steel rope is fixed to a 1×7(1+6) construction type and the bar diameter to 24 millimeters because in these sections our aim is to investigate the capabilities of the shape optimization algorithm. So we have intentionally created conditions to challenge the shape optimization rather than the topology generation.

Evaluation

An example of a system with two elements (two bars or one bar and one rope) is shown in Fig. 3. In this example nodes N1 and N3 have unknown forces with known displacements equal to zero, whereas node N2 has an unknown displacement and a known force P. Considering the space dimension of the problem as 2D, this structure, although very simple is not solvable through Newton's laws of motion. Because the number of the equilibrium equations (a vector sum of the two forces and a sum of the moments about an arbitrary point) is less than number of the unknowns (four unknown reactions at N1 and N3 nodes). Therefore the structure is classified as statically indeterminate. To solve statically indeterminate systems, the deformations must be considered.

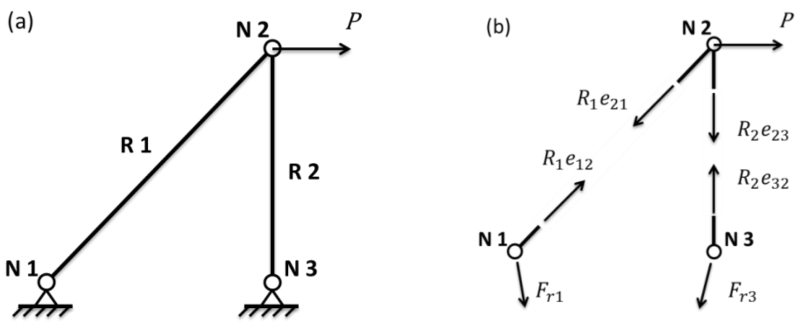


Figure 3-3: A two element structure (a), equilibrium at all three nodes (b)

As the static equilibrium equations are insufficient to determine the internal forces and reactions of the statically indeterminate systems, it is necessary to develop an analysis scheme that can be used for evaluating both determinate and indeterminate structures. This requires developing of the relevant equilibrium, kinematic, and constitutive relations for general structures and then combining these expressions to produce a set of equilibrium equations. The procedure of solving structures can be found in most classical Structural Mechanics books (Armenàkas, 1988). Unlike trusses with tension-compression bars, components used in this approach consists of tension ropes and compression bars. This requires recalculating the stiffness matrix of the structure after defining the tension elements and finally the displacement and force matrixes. This scheme has been programmed in C# and can be used for both two and three dimensional problems independent from structure's degree of freedom.

3.3.2 Shape and topology synthesis

Seed graph

A seed graph defines the scope and boundary conditions of the problem to be solved. In this case, it consists of some arcs and nodes which are labeled as fixed (n0 and n1) or loaded (n2) with different spatial positions. Fig. 4 illustrates a sample seed graph with two fix points and one load in two dimensions (x, y). The vertical distance between the fix points is 576mm and the horizontal distance between fix points and the load is 960mm. The load is 100N in -Y direction. The goal of grammar rules is to transform this seed graph to a graph that represents a meaningful structure for supporting the load.

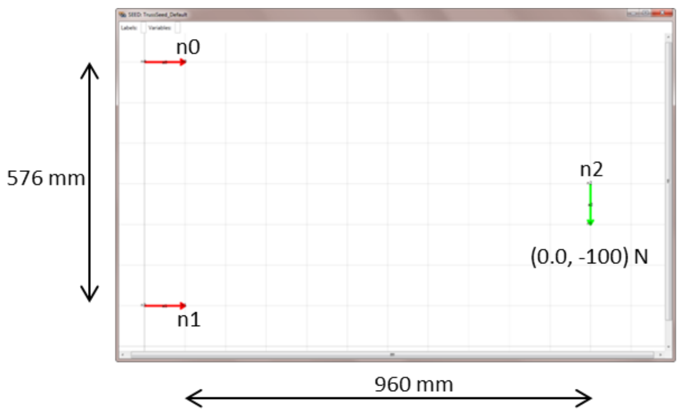


Figure 3-4: A seed graph with one load point and two fixed points

Topology generation

The graph grammar interpreter, which is used for structural synthesis, starts with a seed graph, which represents the bounds of a specific problem. The generation (graph transformation) is carried out through 27 rules which are distributed into seven rulesets (RS). A ruleset is a set of rules that transforms the design from one level of maturity to the next level. Rulesets are used as a means to compartmentalize different phases of the generation process. There are two types of rulesets used in this approach: generative rulesets and transformative rulesets. Generative rulesets define the design space of all possible valid candidates (RS1, RS2 and RS4). These rulesets define the topology of the candidates. Transformative rulesets change the state of generated solutions. They transform a candidate without changing its topology (RS3, RS5, RS6 and RS7). The branching factor of the transformative rulesets is one whereas the branching factor of the generative rulesets is more than one.

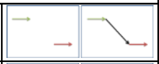
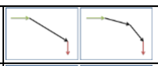
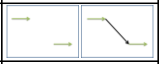
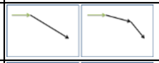
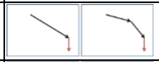
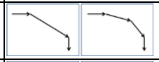
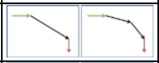
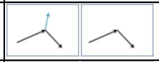
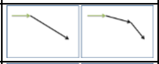
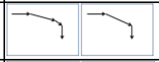
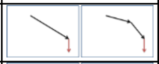
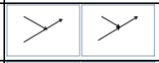
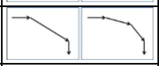
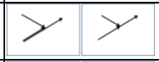
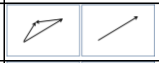
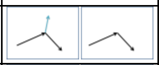
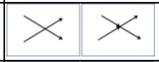
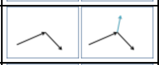
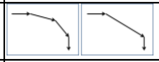
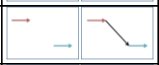
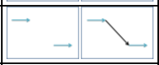
Ruleset	Type	Rule		Description		Ruleset	Type	Rule		Description
1	Generator	1		Connect loads to fixes		5	Automatic	15		Smooth the path between a load and a fix point
		2		Connect loads to load				16		Smooth the path between a load and a joint point
		3	Optimization I	Main load flow path optimization				17		Smooth the path between a joint and a fix point
		4	Trigger rule 1	Trigger rule 1				18		Smooth the path between a joint and a joint point
2	Generator	5		Insert intermediate joint between a load and a fix point		6	Automatic	19		Remove arbitrary arcs
		6		Insert intermediate joint between a load and a joint point				20		Merge Nodes that are very near to each other
		7		Insert intermediate joint between a joint and a fix point				21		Merge nodes that are located on other arcs
		8		Insert intermediate joint between a joint and a joint point				22		Merge adjacent arcs
		9	Trigger rule 2	Trigger rule 2				23		Merge triangle connections
3	Automatic	10		Remove arbitrary arcs				24		Add intersecting joints
		11		Define direction of load at joints				25		Remove arbitrary nodes
4	Generator	12		Connect joint to fixes		7	Automatic	26	Simulation	Initial evaluation of the final results
		13		connect joints to joints				27	Optimization II	Shape Optimization
		14	Trigger rule 3	Trigger rule 3						

Figure 3-5: Grammar rules

In Fig. 5 all 27 grammar rules with a short description of each are illustrated. The rules are created in a general way, so that for different types of problems the same rules can be used. The left picture in the Rule column is the left hand side of a rule (LHS) and the right picture is the RHS of the rule. The graph grammar interpreter converts that part of the seed graph which is matched to the LHS into the graph segment depicted in the RHS. Three trigger rules (4, 9 and 14), two optimization rules (3 and 27), and one evaluation rule (26) does not change the graph elements, therefore their LHS and RHS are not depicted. They are referred to as rules to coordinate their execution among the other rules. Fig. 6 shows the tree structure of the synthesis process. Figures 1, 5 and 6 represent the approach from three different viewpoints with different levels of detail. As can be seen in Fig. 6, rulesets 1, 2 and 4 increase the number of candidates and other rulesets just transform the existing candidates.

The whole approach is developed in such a way that incremental information, which is required in the next step, is added to the design. Aside from the depicted rule conditions in Fig. 5 (like connecting loads to supports), many other additional functions are compiled into the rules to define detailed matching conditions as well as rule actions. For instance, for rule 24, one function aids in the recognition process to find the exact position of the intersection and one function helps in inserting a node at that calculated spatial position. In the following three sub-sections, all seven rulesets including 27 rules are explained in detail.

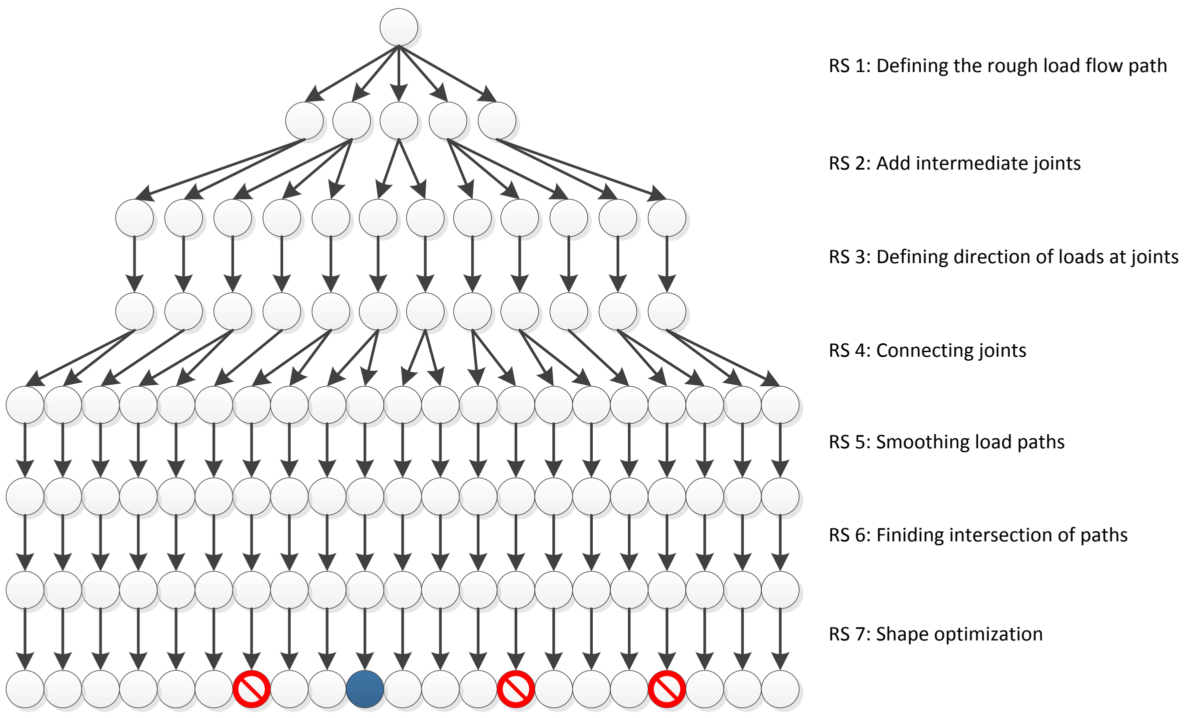

RS 1: Defining the rough load flow path
RS 2: Add intermediate joints
RS 3: Defining direction of loads at joints
RS 4: Connecting joints
RS 5: Smoothing load paths
RS 6: Finiding intersection of paths
RS 7: Shape optimization

Figure 3-6: Tree structure of the synthesis process

3.3.2.1.1 Ruleset 1

The task of the first ruleset (RS1) is to create the main path between load point(s) and the fixed (support) points. In the case of the seed graph in Fig. 4 with only one load and two fix points, just one candidate is generated. If the suggested design in Fig. 7 with one compression bar and one tension rope meets the objective requirements and does not violate different constraints such as buckling, then the ruleset is done. However in this specific example, due to using a very slender bar (24 mm diameter), the compression bar buckles. Because the types of components and their materials are fixed, one should change the path of the load in a way that causes a reduction in the flow amount. This is carried out through an optimization algorithm, which tries to find the optimum direction for the load carrier vectors (Fig. 7).

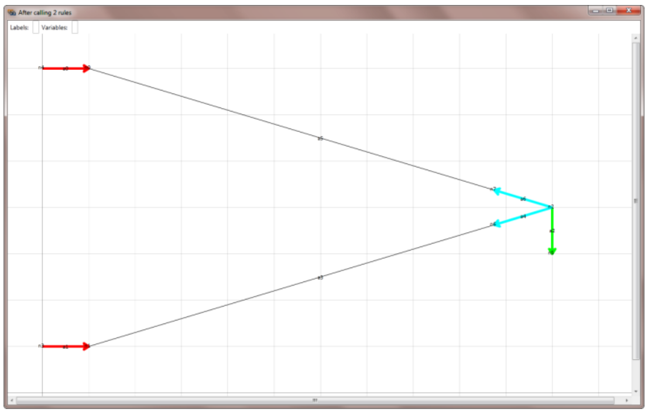


Figure 3-7: Connecting the load points to the fix points

It is clear that the optimum load carrier vectors should be in the same direction as the load vectors, because the net load to be carried will be the same as the load itself. Based on this fact, the load carrier vectors in Fig. 8 should be in the same direction as the load, but the optimization's result in the Fig 8 shows a different direction. This is due to the fact that changes in the direction of the load flow require an extra transmitter member to impose the change, and the more changes in the direction, the more lateral load flow. So in the dilemma of minimizing the main load flow and the changes in the direction of load path, a mediating direction is found through the optimization. In the section 3.2.4, the optimization function is described.

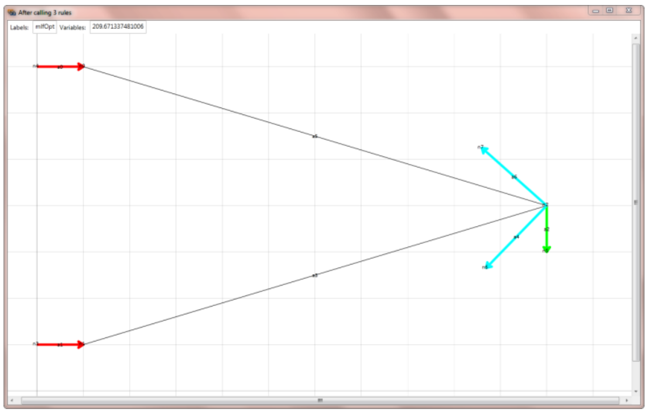


Figure 3-8: Defining the direction of main load flows

3.3.2.1.2 Ruleset 2

Based on the load flow direction from the RS1, four rules in this ruleset break the structure elements into smaller ones and one trigger rule is used to exit the rule set. The segmentation of the load paths depends upon the minimum allowed size for structure elements. In this study minimum size of each segment is 225 mm; therefore each path can be divided maximum into four segments. This minimum segmentation size is different from the minimum element size constraint of the shape optimization algorithm. The segmentation size limit indirectly limits the maximum number of the structure elements. It allows the rules of the ruleset 2 to be applied only on arcs with the minimum length of 450 mm. As illustrated in Fig. 6, this is a generative ruleset for exploring the design space; the single output candidate of the ruleset 1 is populated into sixteen candidates with different number of segments for each load path.

Fig. 9 shows one of the candidates with one segmentation at each path, which means two new joints are created.

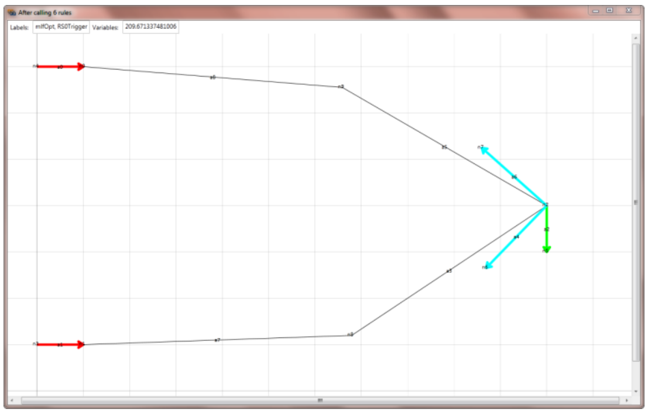


Figure 3-9: Dividing load path segments to smaller pieces

3.3.2.1.3 Ruleset 3

The newly added joints require extra transmitter members to impose the changes in the direction of the load flow. This relatively small ruleset defines the optimum direction of the force to be imposed at the joint. This is calculated based on the direction of both path segments connected to the joint. Fig. 10 shows the effect of this ruleset upon the candidate from the last ruleset. By applying four rules (each rule two times), the two directing arcs at the load point are removed, because they are no longer required and two directing arcs are added to the joints. If the number of segments and consequently the number of joints increases, the number of rule applications also will increase.

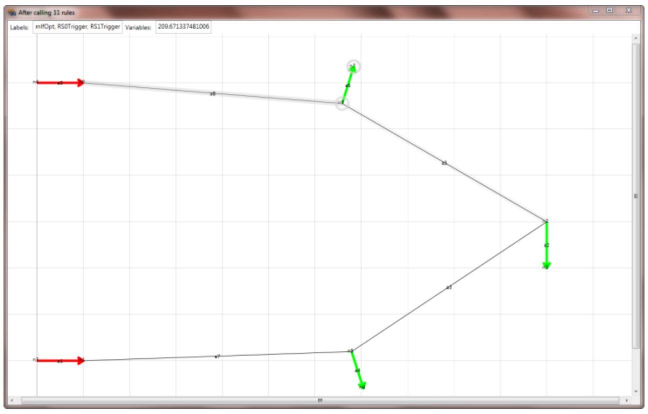

After calling 11 rules
Labels: mifOpt, RS0Trigger, RS1Trigger
Variables: 209.671337481006

Figure 3-10: Defining the direction of load at intermediate joints

3.3.2.1.4 Ruleset 4

Ruleset 4 is the final generative ruleset. Its rules are designed to connect the joints to other joints or fixed points. This ruleset adds more variety to the design space and increases the number of the candidates from 16 to 35. At this stage the topology of the candidates is fixed and the remaining exploration is of parametric variation within these 35 topologies.

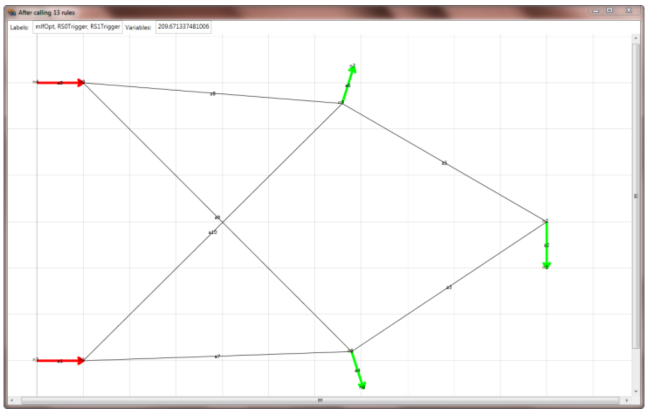

After calling 13 rules
Labels: mlfOpt, RS0Trigger, RS1Trigger
Variables: 209.671337481006

Figure 3-11: Connecting intermediate joints to other joints or fix points

3.3.2.1.5 *Ruleset 5*

The functionality of this ruleset is similar to the ruleset 2, with the segmentation of the paths applied on the secondary load paths and not the primary ones. This segmentation is necessary to find the intersection place of the load paths. In more complicated candidates the effect of this ruleset is more evident.

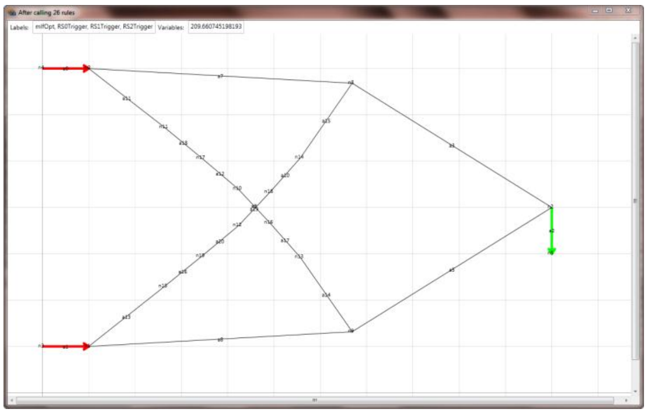

After calling 26 rules
Labels: mlfOpt, RS0Trigger, RS1Trigger, RS2Trigger
Variables: 209.660745198193

Figure 3-12: Smoothing intermediate load paths

3.3.2.1.6 *Ruleset 6*

Finally ruleset 6, which is the final ruleset of the search process, prepares the generated solution candidates for the shape optimization. It finds intersections in the load paths and adds new joints at those places. Rules of this ruleset also remove all unnecessary segmentations of the secondary load paths. Most of the rules in this ruleset are not used for the example in Fig. 13, but in more complicated design solutions, in which changes in the direction of the load flow are more, they are used. In the results section some of these examples are shown.

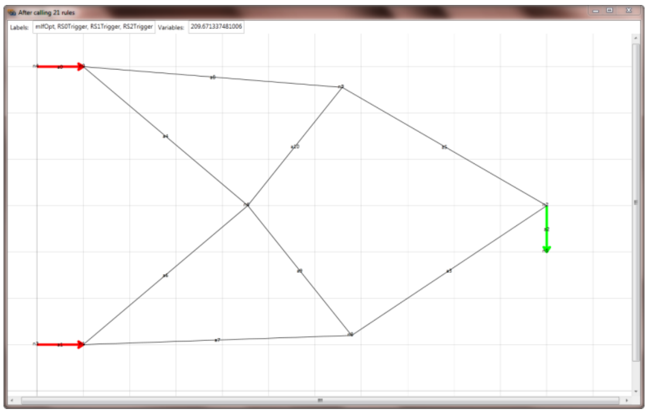


Figure 3-13: Post-processing the solutions

Fig. 14 shows the generation process of a design solution with a finer segmentation. The effect of previous rulesets is illustrated with more clarity in this figure.

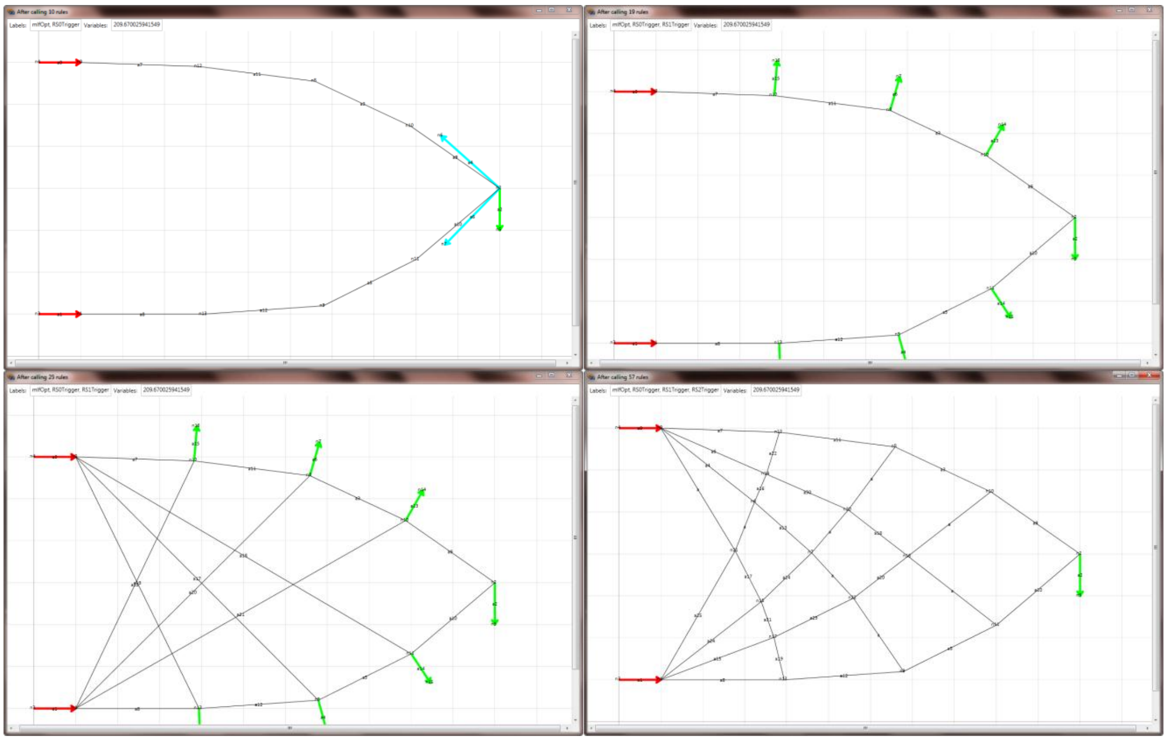


Figure 3-14: The process to generate candidate 32

Search

A breadth first search algorithm has been used to search the design space for all valid candidates. As discussed before, rulesets 1, 2 and 4 explore the design space and rulesets 3, 5, 6 and 7 transform the candidate solutions (Fig. 6). As it is possible to generate the same candidate through different sequence of rule applications, two mechanisms have been considered to prevent duplicate designs. The first mechanism is preventing confluent rules from being applied (confluent rules do not invalidate one another, see Heckel et al., 2002). The second mechanism is a duplicate check. This algorithm –after generating all candidates– compares them with each other and removes those which are repeated.

Fig. 15 illustrates six candidates among the 35 generated solutions. The entire approach requires less than 10 seconds to generate all 35 candidates. The top candidates in Fig. 15 are the simplest to generate with less than 20 rule applications. The solutions with more elements in the Fig. 15 require up to 100 rule applications. At this stage, all generated candidates are stored in a sorted list based on their complexity. The complexity criterion in this context is number of structural elements. It is assumed that the more number of elements the more its construction costs. The approach searches for the simplest structures that can meet all constraints such as stability, buckling and other spatial constraints, with the best performance (i.e. objective value). Therefore the candidates in the sorted list are fed one by one to the final ruleset for shape optimization. The soonest a candidate meets all requirements, the process is terminated and a final solution is chosen. However it is also possible to continue the optimization for all solutions.

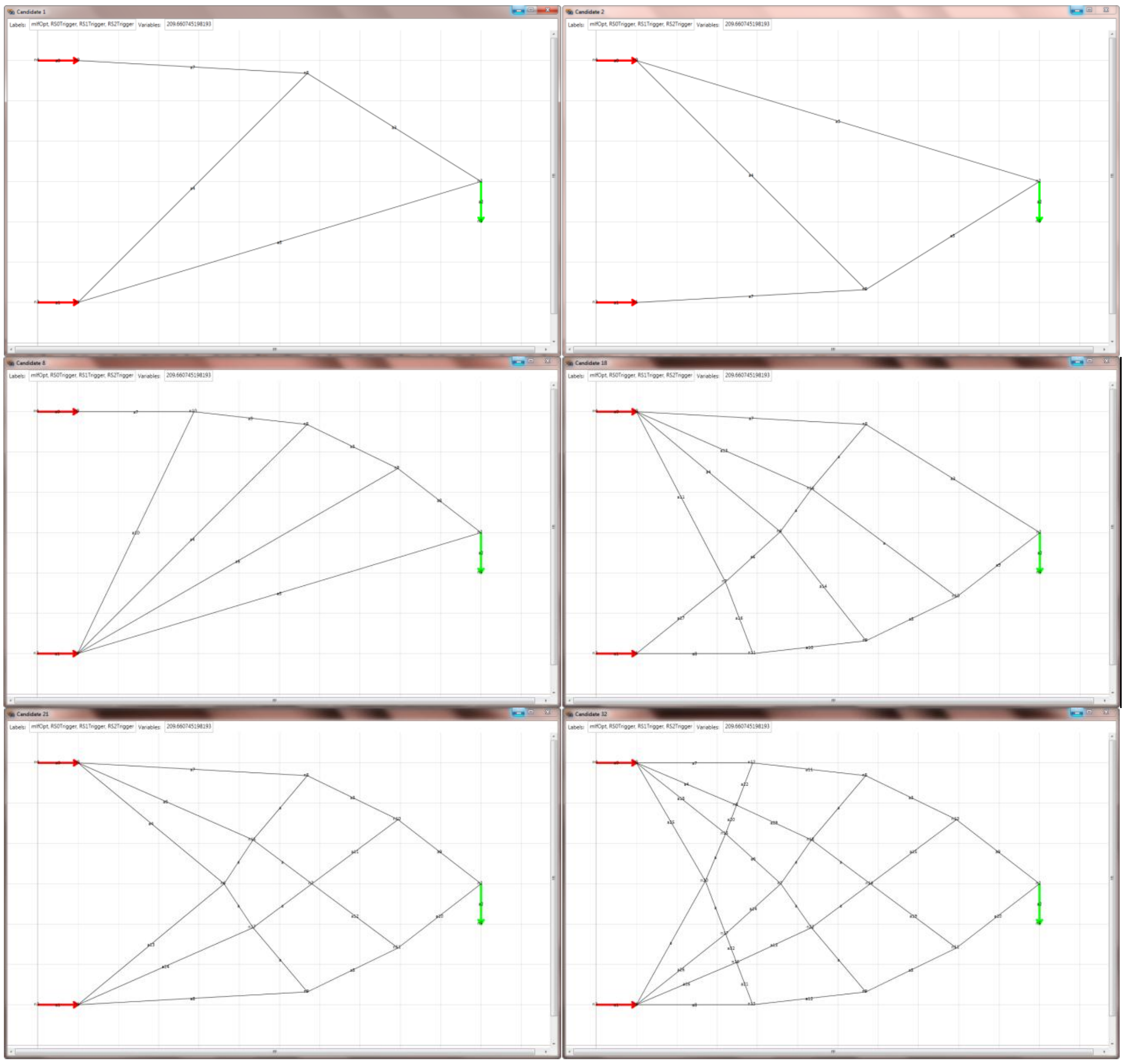


Figure 3-15: Six random candidates between 34 feasible candidates, generated in less than 10 seconds

Optimization

After storing all results of the exhaustive search in a sorted list, the candidates will be further optimized in the second step of the topology generation phase. Due to the smoothness and unimodality of the design spaces as well as the use of efficient optimization algorithms, most of the candidates can be optimized within a few seconds. Based on the idea of the load path principal, which considers forces like fluid flow, an optimum structure is one that minimizes the amount of force at all elements by best distributing the force between the elements. Therefore, the objective function of the optimization is a root-mean square (1):

$$f(x) = \sqrt{\sum_{i=0}^{N} R^2} \qquad (1)$$

where x is the spatial position of the joints, N is the number of structure segments in the layout, and R is the amount of load that flows in each segment. The objective function of the rule 3 is equation 1 plus changes in the direction of load paths, which has been discussed in the sub-section “ruleset 1”.

3.3.2.1.7 Algorithm

For both optimizations (rule 3 and rule 27), the Fletcher-Reeves gradient algorithm is used. These optimizations can be used to minimize the overall load flow of the layout (equation 1), displacement, stress, or any other desired objective. Both optimizations are continuous, because we have only one type of compression bar and one type of tension cable rope to create the structure. Consequently there is no size optimization required for this example. In cases where there are different types of elements (bars and cables) with different material or different sizes, one could use rules to capture this discrete decision making. However, the second optimization (rule 27) could be also a mixed discrete and continuous optimization. This allows us to build a structure with different element types to maximize the performance of the structure.

3.3.2.1.8 Constraints and boundary conditions

Due to the separation of the topology generation from the shape and size optimizations, it is possible to easily include additional constraints and boundary conditions in the optimization. However unlike other conventional structural optimization methods (Luh and Lin, 2011; Noilublao and Bureerat, 2011), this separation of the synthesis process does not affect the quality of the results, because the search algorithm is responsible for exploring the whole design space and generating all valid solutions not the optimization. Therefore, the following constraints have been used in this study; 1) stress, 2) displacement, 3) buckling force, 4) outer spatial boundaries to limit the design space, 5) minimum structure element length, and 6) maximum structure element length (this constraint is – aside from the buckling constraint – due to manufacturing restrictions or esthetic aspects).

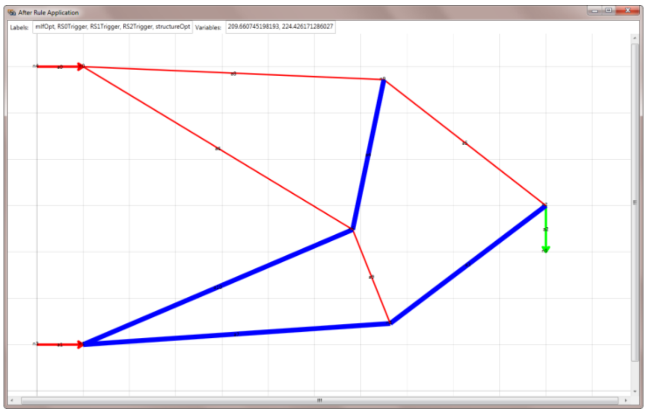
It is also possible to define other spatial constraints such as specific regions to be avoided like holes or other construction requirements. There is no constraint for possible material or manufacturing imperfections considered in this study. Unlike other conventional methods, which add uncertainty in the position of the nodes or use artificial nodal loads (Asadpoure et al., 2011; Jalalpour et al., 2011) to consider imperfections, these issues could be considered in the simulation algorithm, or with a high safety factor for stress, displacement and buckling constraints.

In the next sub section, various examples demonstrate the robust results of the optimization and approach.

3.3.2.1.9 Optimization results

Analyzing the candidate in Fig. 7 shows that the amount of load in the compression bar is equal and opposite the amount of load in the tension rope (174.00N). With the prescribed components of this study (cylindrical 24mm diameter bars) and considering the length of the segment, this structure is failing due to buckling; the maximum allowed buckling force for this element was calculated 33.61N. The objective value for this structure layout (equation 1) is 250.61N.

Fig. 16 shows the optimized shape of the structure layout which was discussed throughout the section 3.2. The overall load (equation 1) objective value is 224.43N. The maximum load in the structure is 106.96N in a tension rope and -91.28N in a compression bar. The reason of this difference lies in the buckling constraint. For this example and the following ones in the approach section, the displacement is not considered as a constraint, but it is determined through the analysis. The displacement at the loaded point is (13.10, -50.58) millimeters in x and y directions. This seems large, because the selected rope is very thin (just 2mm diameter) but it has a very high breaking load (2540N). This selection was deliberately done to challenge the optimization process.


After Rule Application

Labels: mlfOpt, RS0Trigger, RS1Trigger, RS2Trigger, structureOpt

Variables: 209.660745198193, 224.426171286027

Figure 3-16: Optimized structure (min. overall load flow objective)

Fig. 17 shows an optimized structure with 15 joints and 32 components. The optimization took about 65 seconds and the best objective function was 265.51N. This value is less than the value obtained for the Fig. 16. However the distribution of the load is much better and the shape of the structure has more symmetry. Consequently, the buckling constraint was almost inactive. The maximum tension value is 78.05N and maximum compression is 77.97N. The amount of displacement shows significant improvement; (15.90, -31.35) millimeter in x and y directions respectively.

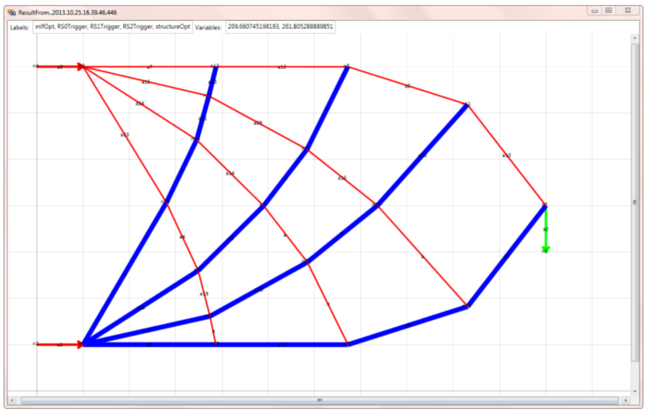

ResultFrom..2013.10.25.16.39.46.446
Labels: mlfOpt, RS0Trigger, RS1Trigger, RS2Trigger, structureOpt
Variables: 209.660745198193, 261.805288889851

Figure 3-17: Optimized structure with finer segmentations (min. overall load flow objective)

Fig. 18 has the same layout as Fig. 17. The only difference is in the fact that no design space limitation is considered for the structure, therefore the structure extends out of the initially considered design box, which shows significant improvements in the objective value and also displacement at the loaded point. The objective value of the structure in the Fig. 18 is 210.58N, and displacements (6.83, -25.99) millimeter. In this structure, we can see also that cables are maximum 60.11N under tension and bars are maximum -62.53N under compression. This shows the very good distribution of the load in the structure, which results in a very good objective function value.

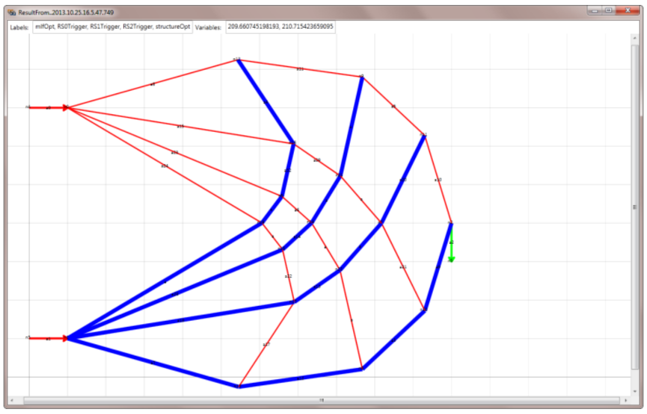

ResultFrom..2013.10.25.16.5.47.749
Labels: mlfOpt, RS0Trigger, RS1Trigger, RS2Trigger, structureOpt
Variables: 209.660745198193, 210.715423659095

Figure 3-18: Freeing the spatial constraints of the design space (min. overall load flow objective)

Again Fig. 19 shows a structure layout that is topologically similar to the ones in Fig. 17 and 18. The reason a different shape is obtained for the structure is that the objective function is now minimum displacement. The shape illustrated in the Fig. 19 gives us the minimum displacement for the layout which is (2.55, -23.04) millimeter. This is the lowest value among all 35 candidate solutions; if a displacement less than this is desired, there are two options. The type and size of the used components could be changed, a finer approximation of the load path could be allowed. This can be achieved by reducing the minimum length of the segmentation which is 225 millimeter and consequently creating more segments in the main load path. For this optimization, tension cables with maximum 47.46N load and compression bars with maximum 70.17N load are used. This shows clearly that the problem of the displacement comes from the very thin wire ropes.

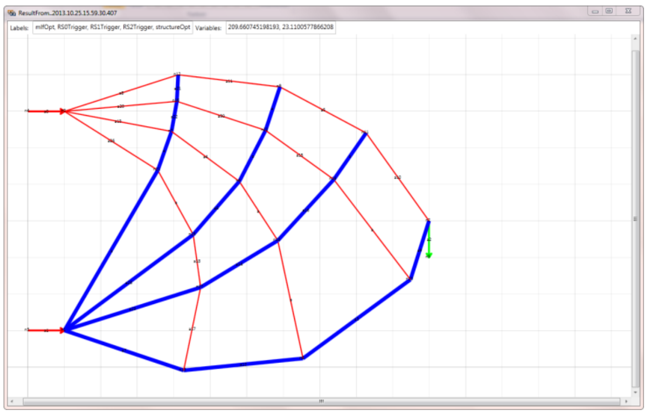


Figure 3-19: Changing the objective function (min. displacement at loaded point)

Finally Fig. 20 shows the optimal designs for the same problem as in Fig. 16, with the lower fixed point bearing only forces in the x-direction. Considering the layout total load flow as the objective function, a displacement of (9.46, -65.70) millimeters in x and y directions and an objective value of 229.73N is obtained. Whereas by considering the displacement as our objective function the displacement reduces slightly to (9.28, -62.53) millimeters and the load flow increases to 258.26N. The displacement is not significantly reduced in comparison with the previous result. It is interesting to note that the optimized candidates have higher tensile loads than compression loads due to the buckling constraint. This is also reflected in the Fig. 16, in which ropes are under more tension than bars under compression. In this example also ropes are under maximum 127.23N (Fig. 20a) and 116.67N (Fig. 20b) tension and the bars are compressed only with maximum 70.70N and 99.72N forces.

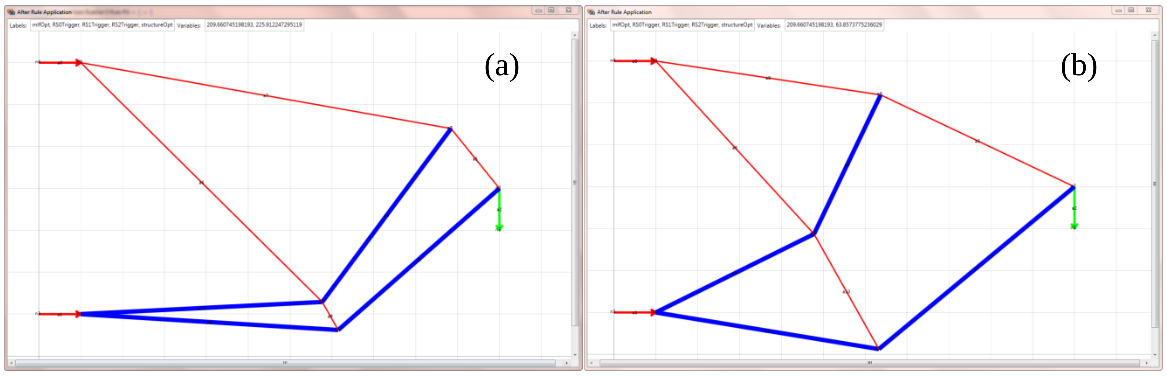


Figure 3-20: Freeing fixed point n1 from load in the y direction; min. overall load flow (a), min. displacement (b)

It is possible to play to excess with different parameters of the optimization, but the main point of this methodology is in the way that we approach the topology generation (through load flow principal) and its formulation and representation in grammar rules. Finally the use of a tree search algorithm to explore the design space appears to be novel. In the results section the abilities of the approach is explored through various examples.

3.3.3 Transformation

After automated synthesis of the topologies, they can be transformed into three dimensional shapes through a converter which uses the Parasolid geometric kernel (Siemens PLM

Software Inc., 2013). The transformer converts nodes in the graph into spheres, and arcs into cylinders. The shapes are saved as STL files. Fig. 21 shows the candidate topology of Fig. 18 converted to a 3D shape. For this specific design, the conversion took less than 2 seconds. As discussed before, this transformation is only for 3D visualization of the results, which is necessary when synthesizing problems in 3D space.

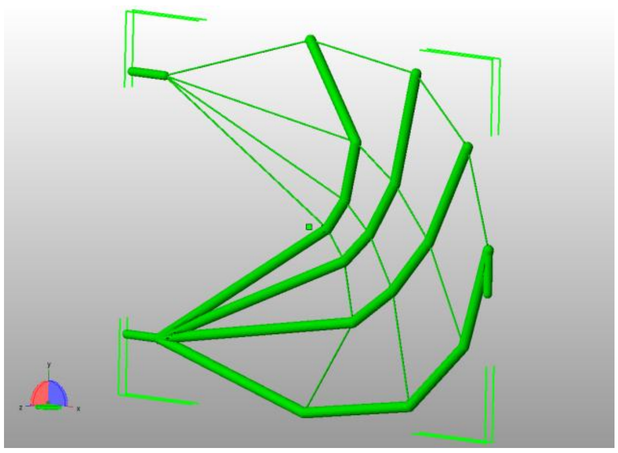


Figure 3-21: A converted topology into 3d shape

3.4 Results and Discussions

In this section the abilities of the developed approach in generating meaningful topologies with optimum shapes and sizes is shown through various examples and benchmark problems. Some examples focus more on showing the capabilities of the optimization, whereas most of them focus on the topology generation phase.

Throughout the entire approach section, a cylindrical cross section bar with 24 millimeters diameter and a steel rope (see section 3.1.1) have been used. As discussed in the previous sections, it is possible to use different type of elements in constructing the structure such as rectangular cross section bars, hollow cylindrical cross section bars, triangular cross section bars with different characteristics or ropes with different dimensions and different construction types. In the next sub-section, the diameter of the bar is changed from 24 millimeters to 40 millimeters to show the effect of changing components on the shape of the structure. In all sub-sections from 4.3 to 4.5 the construction type of the rope components is also changed to (6×7(1+6)+1×7(1+6)). Furthermore, the objective function will be for all of the examples the total load flow (formula 1), unless it is explicitly mentioned.

3.4.1 Changing the building material of the structure

Fig. 22 illustrates the effect of changing the chosen components of the structure on the shape of the structure. In this example only the size of the bars is changed from 24 mm to 40 mm. When using the total load objective function, Fig. 22a is the result and in the case of using the displacement as the objective value, Fig. 22b is the outcome. The overall load flow, displacement and maximum tensions and compressions for both cases are as follows respectively:

- 208.38N overall load flow, (8.94, -50.19) millimeter displacement, maximum tension 100.32N, and maximum compression 98.78N
- 231.30N overall load flow, (7.99, -39.73) millimeter displacement, maximum tension 95.14N, and maximum compression 116.55N

Fig. 22a and b show that the buckling effect is not prominent and results of Fig. 22b shows that the main reason of the displacement is the stretch in the wire ropes, because in this case the components have higher compression load than tensile loads.

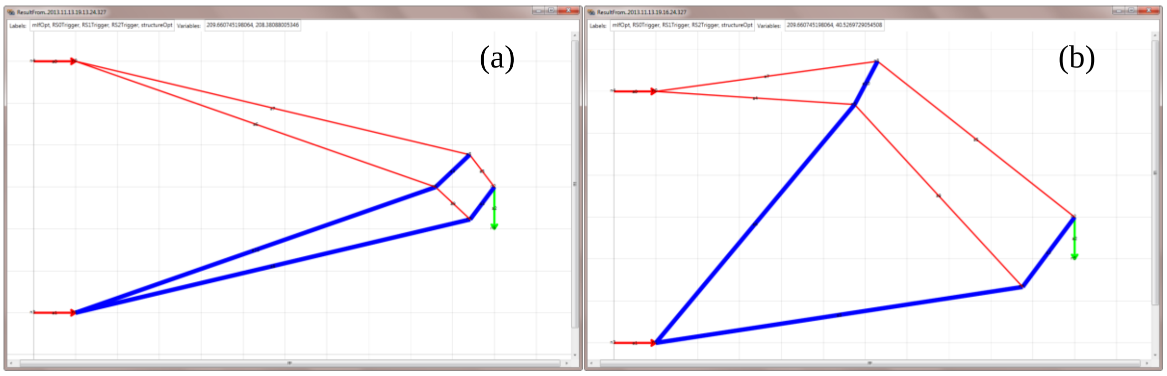

ResultFrom_2013.11.13.19.13.24.327
Labels: mffOpt, RS0Trigger, RS1Trigger, RS2Trigger, structureOpt
Variables: 209.660745198064, 208.38088005346
(a)
ResultFrom_2013.11.13.19.16.24.327
Labels: mffOpt, RS0Trigger, RS1Trigger, RS2Trigger, structureOpt
Variables: 209.660745198064, 40.5269729054508
(b)

Figure 3-22: Changing bar diameter from 24mm to 40mm. objectives are min. load flow (a), min displacement (b)

3.4.2 Changing the direction and position of the load

The effect of changing the direction of the applied load is shown in the Fig. 23. All of the components in the left picture are under tension and in the right picture are under compression, but the shape of the structure remains the same; the objective is minimizing the load. In both cases, the total load is 95.15N but the displacement of the structure under tension is (13.74, -0.44) mm in x and y directions, whereas these values are significantly smaller in the structure under compression; (-0.42, 0.01) mm.

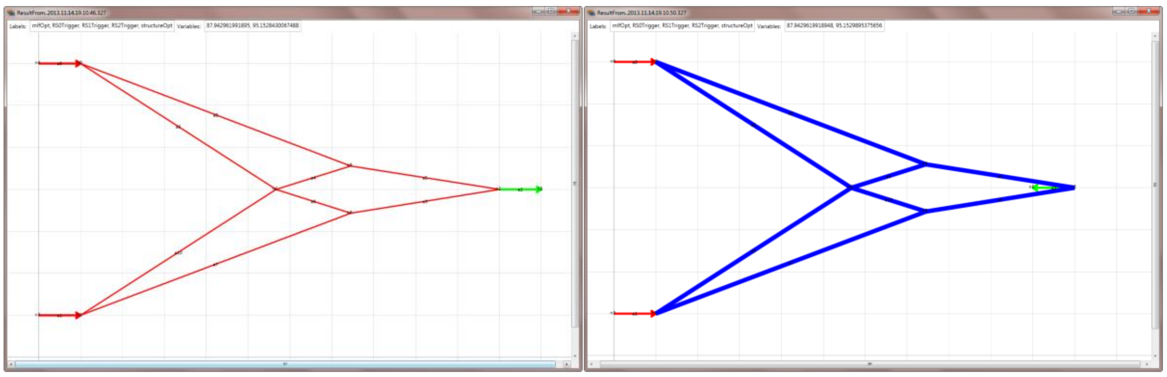

ResultFrom_2013.11.14.19.10.46.327
Labels: mffOpt, RS0Trigger, RS1Trigger, RS2Trigger, structureOpt
Variables: 87.9429619918895, 95.1528430067488
ResultFrom_2013.11.14.19.10.50.327
Labels: mffOpt, RS0Trigger, RS1Trigger, RS2Trigger, structureOpt
Variables: 87.9429619918948, 95.1529895375656

Figure 3-23: The effect of changing the direction of the load on the structure layout

In Fig. 24 the location of the load is moved to the right bottom of the design space. In these shapes, the flow of the load to the shapes is clearly illustrated. Without increasing the size of the bars, it is not possible to prevent buckling and create the desired shapes with these topologies; this will be elaborated more in the next sub-section. For these two examples and all other following examples in the result section, the construction type of ropes is (6×7(1+6)+1×7(1+6)) and the diameter of bars is increased to 40 mm.

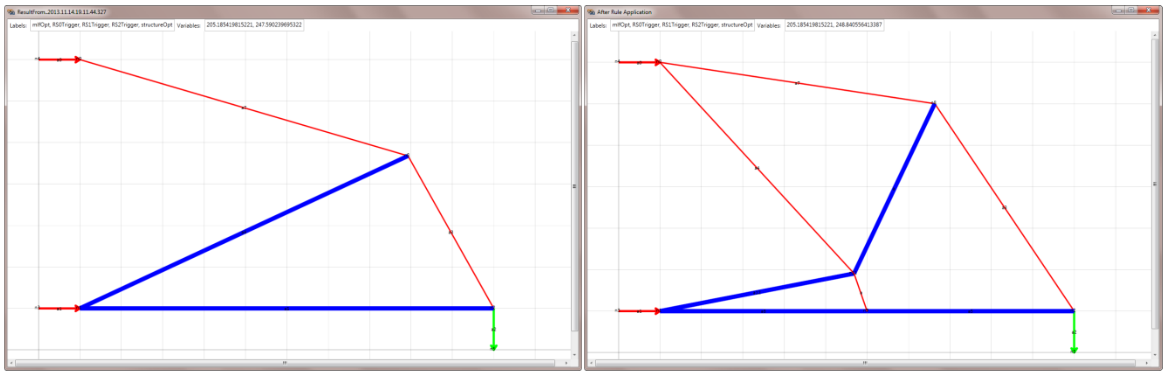


Figure 3-24: The effect of changing the location of the load on the structure layout

3.4.3 Topological benchmarking of the results

The aim of this sub section is to compare the results of the approach with diverse benchmark examples in the continuous structural optimization literature. The comparisons are initially based on the topology, and then the shapes of the results. The first benchmark example is the cantilever beam with different lengths and with loads at different positions and the second example is the Michell structure. These are the most used benchmark examples in the literature (Bendsøe and Sigmund, 2003; Bulman et al., 2001; Eschenauer and Olhoff, 2001; Liang, 2005; Luo et al., 2009; Rong and Liang, 2008; Wang and Wang, 2004; Wang et al., 2007; Yulin and Xiaoming, 2004).

The most famous benchmark example that can be found in the field is illustrated in Fig. 25. These two results have been among the 35 candidates which were created for the problem in the approach section. The shape of the left structure in Fig. 25 is slightly different from that of Fig. 16, because in this example the buckling constraint is no longer active, due to using thicker bars.

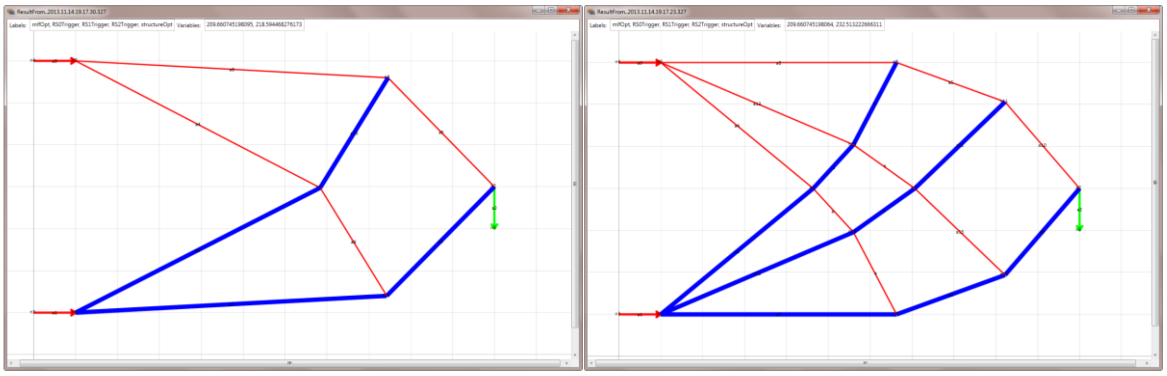


Figure 3-25: Short cantilever beams

Fig. 26 shows other diverse topologies for a problem in which the load and fix points are farther from one another, this is especially accentuated on the problem on the right. The approach solves problems with different sizes, dimensions, and loads and support numbers in the same way following the load flow principal. Therefore, the created topologies and to a very good extend the shapes are nearly the same as those found in the literature. However the shape is very dependent upon the objective function and active constraints.

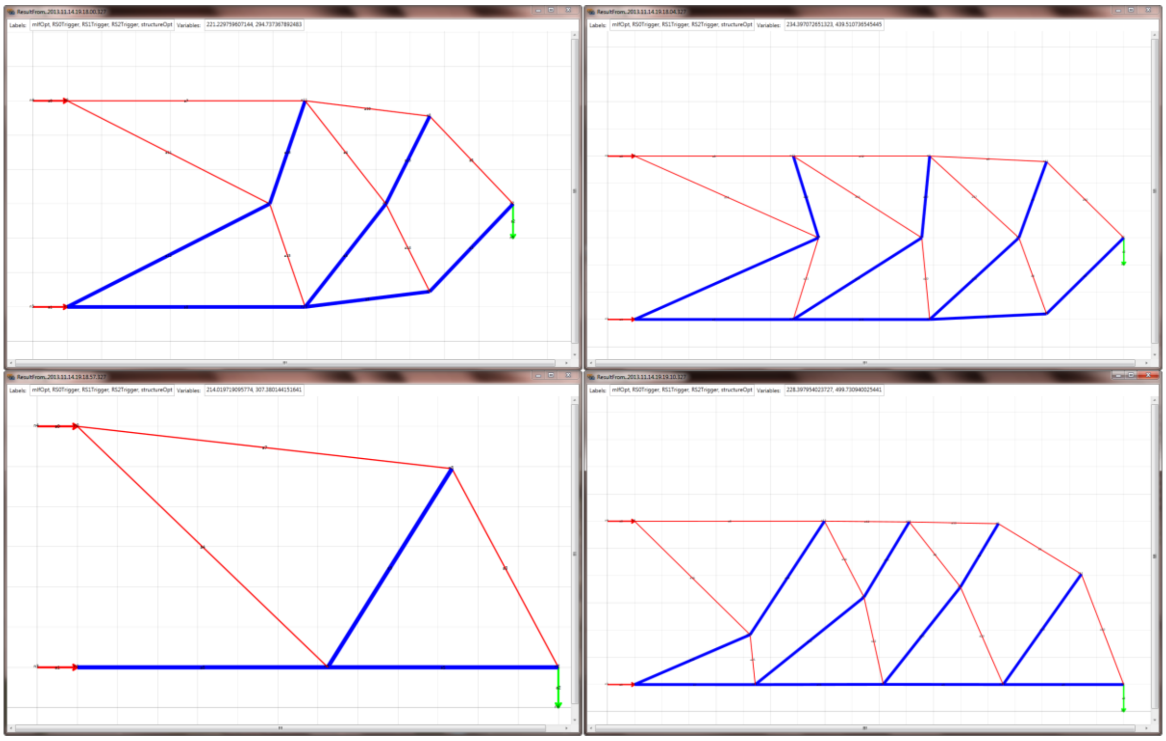
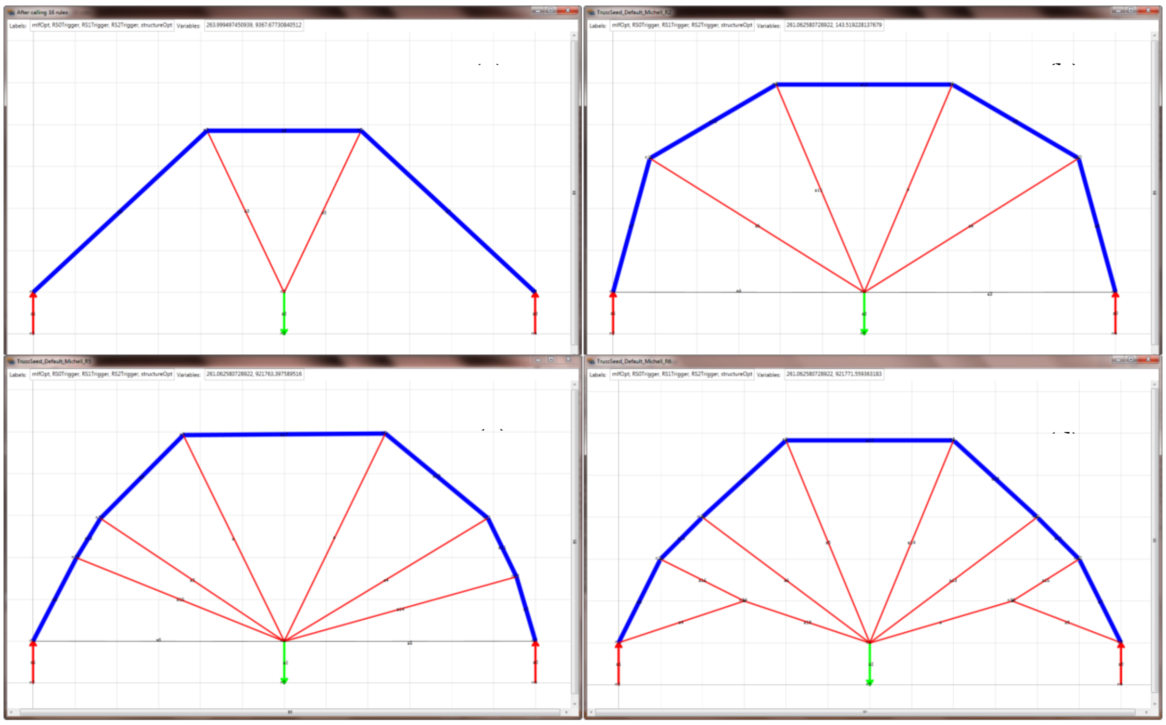


Figure 3-26: Long cantilever beams

The second most famous problem in the literature is the Michell structure. Various solutions have been suggested for this benchmark example that can be seen in the Fig. 27. These results are in accordance with those found in the literature both in terms of topology and shape. This example requires more time to be solved with the presented approach, due to the bigger change in the load flow direction. In the previous examples, the load changes maximum 90 degrees, but in this example the load changes 180 degrees.

By setting the minimum size of the path segments to 225 mm, only two candidates are generated in less than one second, one of which is illustrated in the Fig. 27 (a). By reducing the minimum size of the segments to 125 mm, in order to achieve a finer approximation of the load paths, number of candidates increases to 53 solutions in 19 seconds. Three of the 53 samples are shown in Fig. 27a, b and c. And finally, by reducing the minimum size to 100 mm, we were able to generate more than 500 candidates in 10 minutes (see a sampling in Fig. 28). It is important to note that the results from finer segmentations always include the results of the longer segmentations.


After calling 16 rules
Labels: mffOpt, RS0Trigger, RS1Trigger, RS2Trigger, structureOpt Variables: 263.999497450939, 9367.67730840512

TrussSeed_Default_Michell_R2
Labels: mffOpt, RS0Trigger, RS1Trigger, RS2Trigger, structureOpt Variables: 261.062580728922, 143.519228137679

TrussSeed_Default_Michell_R5
Labels: mffOpt, RS0Trigger, RS1Trigger, RS2Trigger, structureOpt Variables: 261.062580728922, 921763.397589516

TrussSeed_Default_Michell_R6
Labels: mffOpt, RS0Trigger, RS1Trigger, RS2Trigger, structureOpt Variables: 261.062580728922, 921771.559363183

Figure 3-27: Four solutions similar to the literature results for the Michell structure problem

Regardless of the minimum segmentation size, the set of solutions (prior to optimization) can be sorted on these criteria; 1) number of components in each candidate, 2) initial evaluation of the candidates, and 3) symmetry of the candidates (if appropriate). Therefore candidates with lower costs (number of components), better evaluation results and probably better symmetry come to the top of the list for the final optimization. However the optimization and analysis are quick enough and – even problems with about 30 variables (15 nodes in 2D) – can be solved within few minutes. In general the optimization results show that the total load flow in the Fig. 27 is slightly less than those in the Fig. 28, whereas the displacement of the structures in the Fig. 27 is more than the structures in the Fig. 28. If – aside from the stability, robustness and security of the structure – aesthetical aspects are of importance, the structures in Fig. 28 may be preferred over those in the Fig. 27. Indeed, the approach through covering of the design space, generates many sophisticated designs, this gives the designer a very wide range of possible options. A visual presentation, like that shown in this chapter might prove a useful output in allowing the designers to see the tradeoffs in the results.

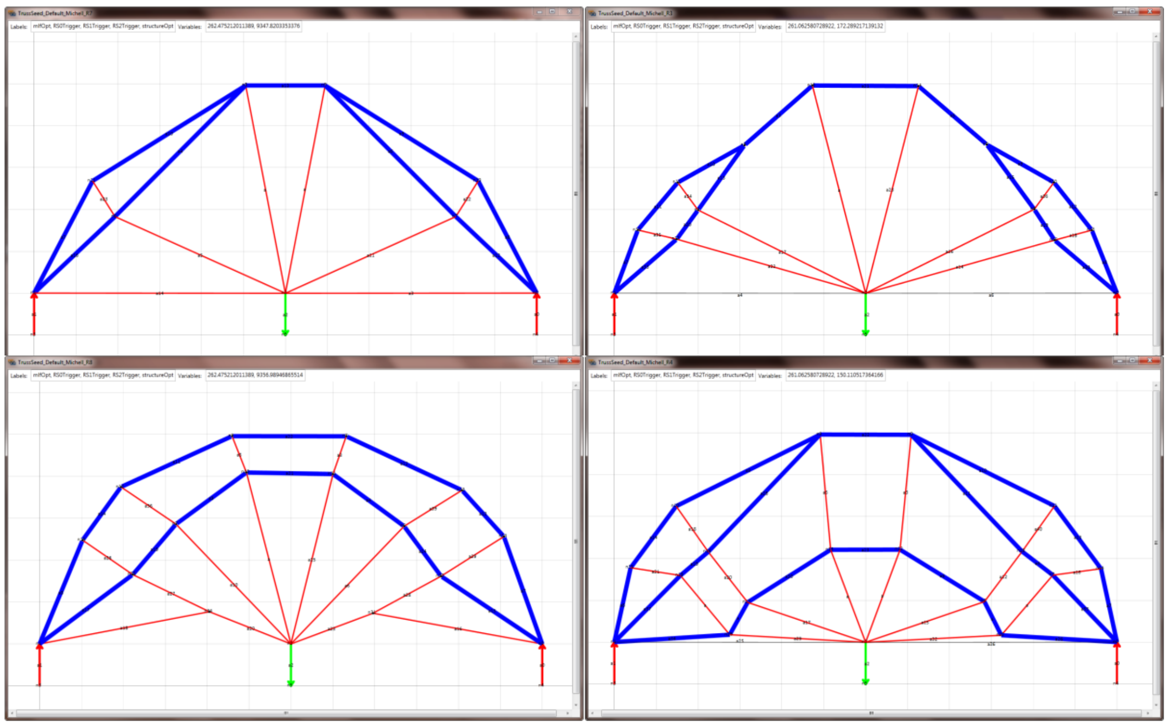


Figure 3-28: Four solutions for the Michell structure problem

3.4.4 Three dimensional problems

As discussed in the background and approach sections, the concept of load flow path is not limited to 2D problems or 3D problems. Therefore the developed methodology is able to solve 3D structural problems with almost no change in the rules. However following differences in the nature of the problems exists; three dimensional nodes have three spatial variables, consequently the optimization time increases 50%. For visualizing the graphs an extra conversion is required. The conversion may take up to 10 seconds for big size structures. And rule 24 should be modified, because finding the intersection of elements in 2D differs from 3D.

Fig. 29 and 30 show three structures for three different problems. As can be seen the placement of loads and fixed points is different in all three problems, which affects the optimum topology and shape of the structure. In Fig. 30 the distance of the load from the fix points is 50% more than the structures in the Fig. 29.

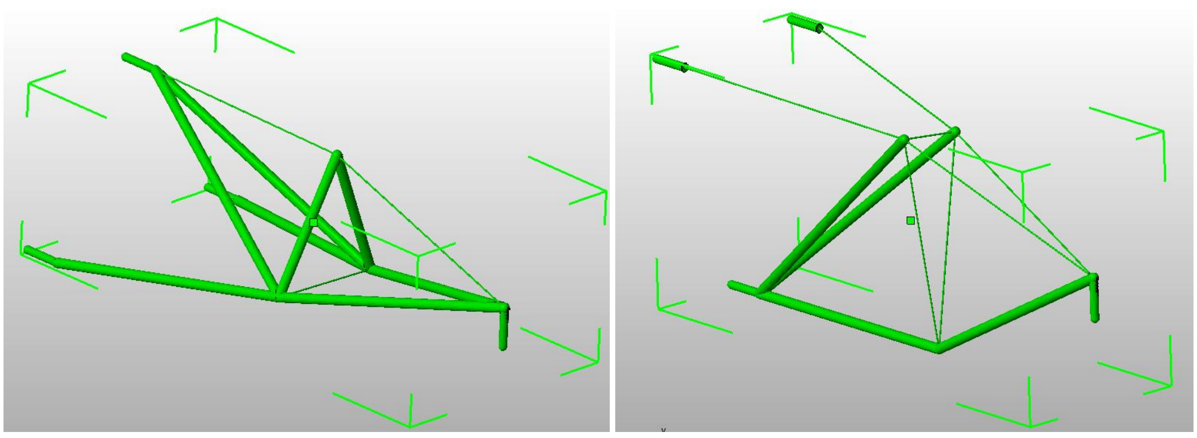


Figure 3-29: Short cantilever beams in 3d

The number of generated topologies for the examples in the Fig. 29 is almost 100% more than the similar 2D problem which has been discussed in the approach section (69 topologies with minimum segmentation 225 mm). Consequently the time has been increased to two minutes. This time is much more than the 10 seconds required for the 2D problem, because the tree-search algorithm requires more time to search a larger design space and find feasible solutions. An interesting characteristic of the truss in Fig. 30 is the joining of three wire ropes at two joints. Using cables in building trusses allows for flexible and foldable structures that act as rigid structures when fully deployed. Furthermore they are lightweight and very material efficient.

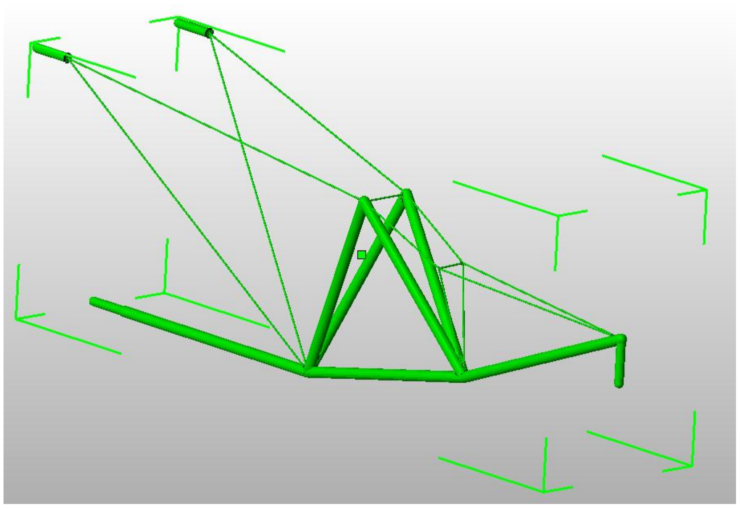


Figure 3-30: Long cantilever beams in 3d

3.4.5 Exploring an arbitrary ground structure

In this sub-section it is shown how the proposed method explores the design space in a more general manner than a fixed ground structure. The efficiency of the approach for structures with more than one load and many optional support points is demonstrated through the example shown in Fig. 31. The arrows at the top of the Fig. 31 represent three masses (which are 100N, 200N and 100N respectively) and the five arrows at the bottom of the Fig. 31 are the optional support points. The objective is to design a truss structure with minimum total load to sustain these masses.

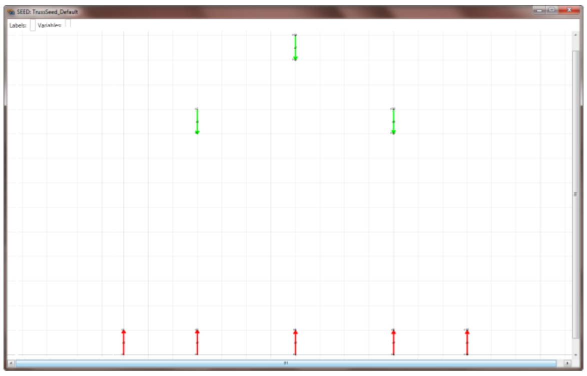


Figure 3-31: Seed graph representation of a problem with three masses and five support options

Recall that other approaches to truss topology optimization typically discretize the design space with a nodal mesh as a ground structure, in which every node is connected to almost every other node in the domain and the members have identical cross-sectional area. Fig. 32 shows two typical ground structures for the defined problem in the Fig. 31. In the left picture of the Fig. 32 all loads are connected to each other and to all optional supports, whereas in the right picture a grid of nodes is the ground structure. The proposed approach in this study has the directness of the first ground structure and at the same time has the flexibilities of the second ground structure approach.

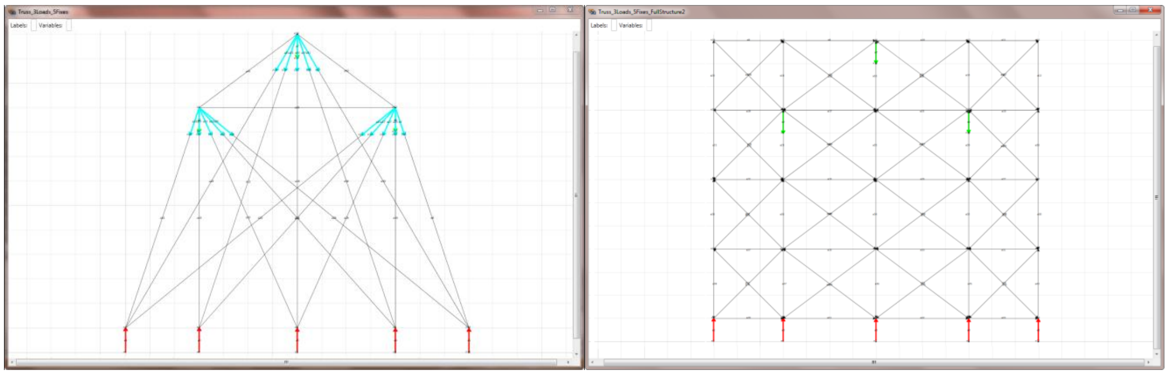


Figure 3-32: Two common ground structures for the problem defined in fig. 31

As discussed, the approach searches the design space and finds all possible ways to flow the load from the applied load points to the supports. This concept is the same for any number of load or support points. Fig. 33 shows two candidate topologies, which satisfy the minimum requirements for a stable load flow. The left structure is minimally stable as removing any component can cause instability in the structure. The approach begins with the shape and size optimization of the simplest feasible designs (e.g. Fig. 33), as soon as a design is found that meets all requirements and does not violate the constraints, the synthesis process can be terminated. However, if the manufacturing cost of the structure is not the only objective of the automated synthesis process, the optimization of the solutions can be continued to find an appropriate solution.

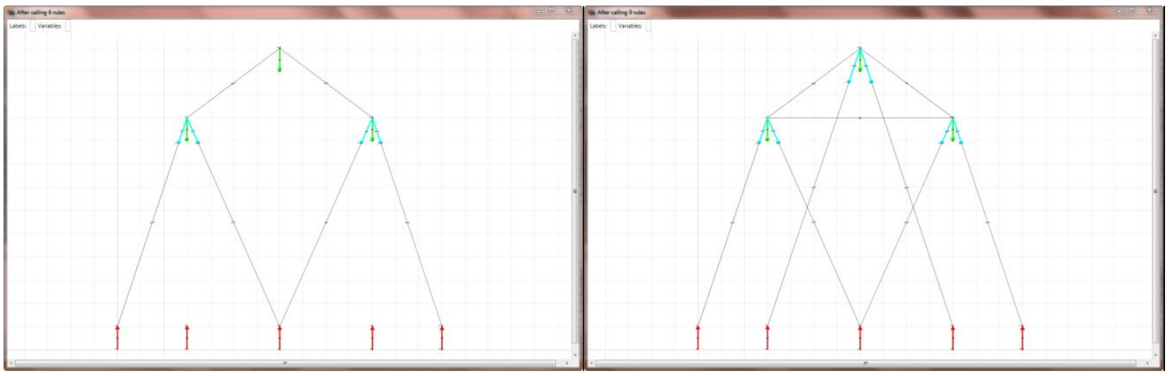
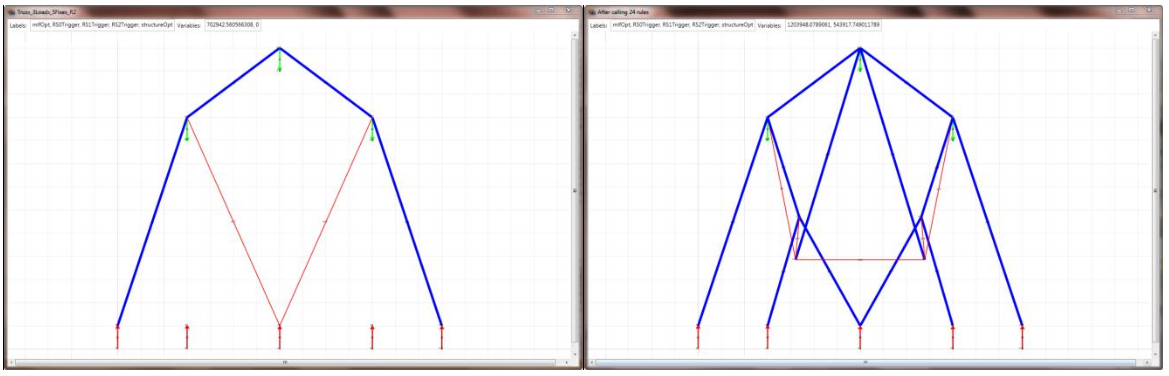


Figure 3-33: Two topological candidates

Fig. 34 shows the optimized shapes of the candidates in the Fig. 33. The right structure shows clearly that the approach is not limited to the fixed ground structure suggested in Fig. 32. It combines the directness of the first type of the ground structures with the wide covering of the second type of the ground structure to find the optimum size, shape and topology of the structures.



mifOpt, RS0Trigger, RS1Trigger, RS2Trigger, structureOpt
702942.560566308, 0
mifOpt, RS0Trigger, RS1Trigger, RS2Trigger, structureOpt
1203948.0789061, 543917.749011789

Figure 3-34: Optimized candidates in Fig. 33

3.5 Conclusions

A new approach for shape, size and topology optimization of cable truss structures using a generative design method is presented. This approach uses graphs to represent both the topology and the shape of structure layouts. This allows a very fast generation of topological solutions for a given design problem. After exhaustive search of the design space, the solutions are stored in a list for further detailed shape design and optimization. To visualize the solutions, the graphs are converted to 3D shapes via Parasolid. These shapes can be used to extract the construction drawings. The simulation model is fully separated; therefore it should be fairly easy to augment the approach to solve problems such as those with seismic loads, uncertainty in materials and construction with any number of loads and supports. Dividing the synthesis process into shape and topology phases does not affect the quality of the results, because the tree-search algorithm is responsible for exploring the whole design space and generating all valid solutions, while the optimization is responsible for the shape and size optimization of each topology.

Ongoing research in this study is focused on improving the analysis to include the effect of the weight of the components. Implementation of this step is important in order to have a better shape and size optimizations. The rules may be flexible enough and independent of the simulation model that the approach might be used for other domains such as compliant mechanisms design. Because the performance of a compliant mechanism is the ability to transfer loads from one or more applied points to one or more desired points. Another important field of research that can increase the generality of the approach is handle obstacles in the problem specifications. The reason for this investigation is that obstacles are an undeniable part of the real world design problems. An interesting field of research might be to use the output results of the approach as input for conventional continuous topology optimization methods. Due to good initial designs, convergence can be faster and many problems might be solved that are hitherto not solvable.

4. Tensegrity Form-Finding Using Generative Design Synthesis Approach

The aim of this research is to produce large irregular tensegrity structures using a generative design synthesis approach. Unlike most of the form-finding methods, the approach does not require the description of the connectivity of the tensegrity structures to define the shape of the tensegrities. It uses graphs to represent the tensegrity structures, which allows a very fast generation of stable tensegrity solutions for a given design problem. The graphs are used to define different configurations for a given design problem. After generating the graphs, they are transformed into meaningful 3D shapes. The effectiveness and robustness of the proposed method is checked by solving a variety of test problems.

Keywords: Tensegrity Structures, Form-Finding, Topology Generation, Graph Grammar.

4.1 Introduction

Tensegrities are self-supporting structures, which consist of a set of disjoint struts (rigid elements), which are connected by pre-stressed tensile strings. These structures maintain their shapes and equilibrium due to a stress imposed by compression of struts and tension in the cables (Wang, 1998), therefore the structures return to their stable configuration when subjected to a perturbation. The self-equilibrium characteristic of tensegrity structures along with their lightweight, fold- and deploy-ability, makes them attractive for architectural and engineering projects such as robotics and other spatial applications (Chan et al., 2004; Paul et al., 2006). However, form-finding process – determining shapes and self-stress states of structure elements – is not a trivial task. Different researchers have proposed different type of methods for finding the form of tensegrities such as; the force density method (Estrada et al., 2006; Masic et al., 2005; Schek, 1974; Vassart and Motro, 1999; Zhang and Ohsaki, 2006), the dynamic relaxation method (Barnes, 1999; Motro et al., 1987) and the marching procedure (Micheletti and Williams, 2007). Zhang et al. (2006) used a refined dynamic relaxation method for form-finding of non-regular tensegrity systems. Shea et al., (2002) used stochastic search for developing intelligent tensegrity structures. Rieffel et al. (2009) introduced an evolutionary algorithm to produce large tensegrity structures. Tran and Lee (2010) proposed a numerical method for initial self-stress design of tensegrity grid structures. Most recently (Koohestani, 2013) has formulated the form-finding of tensegrity structures as two unconstrained minimization problems with eigenvalues of modified force density matrixes as their objective functions. And Zhang et al. (2014) has proposed a form-finding method based on the structural stiffness matrix using various algorithms such as stochastic selecting algorithm, the restricted step algorithm, and the line search algorithm. Comprehensive review of methods proposed for form-finding (Tibert and Pellegrino, 2003), static analysis (Juan and Mirats Tur, 2008), and dynamic analysis (Mirats Tur and Juan, 2009) of tensegrity structures can be found in the literature.

In most of the existing methods a description of the connectivity of the tensegrity structures (topology) is required to define the shape and member forces and to determine the selfequilibrated structural configuration. Some researchers – through joining smaller tensegrities together – could generate large regular tensegrity towers (Masic and Skelton, 2004; Nishimura and Murakami, 2001). However, to discover novel structural forms and shapes – considering aesthetic and mechanical properties – and to explore the entire search space of large scale problems – with irregular and unsymmetrical geometries – new robust methods are required. The form-finding problem can be divided into two parts: determining the connectivity of struts and tensile elements, and specifying the spatial property and geometry of the structure elements. This work covers both aspects of the form-finding, however the main focus lies in the first step of the form-finding; generating large, irregular topologies.

In this approach, graphs are used to represent the tensegrity structures. Although we are not the first to use graphs as a means of representing tensegrity structures (Guzman and Orden, 2004; Motro, 2003; Rieffel et al., 2009), the incorporation of a graph grammar approach with conventional search algorithms creates a novel framework to explore large scale irregular tensegrities. The vertexes of the graph represent endpoint of struts and edges represent the struts and tensile cables. To discriminate between the edges, the struts are labeled as “bar” and tensile strings as “cable”. In this approach, the desired design space is first created through a set of highly connected nodes. A search algorithm uses grammar rules to explore the design space and generate all valid (statistically stable) solutions. After removing the intersections and unnecessary connections between struts, the results are transformed into three-dimensional (3D) shapes for visualization. The efficiency of the proposed approach is checked through producing large irregular tensegrity structures. This chapter is organized as follows. Section 2 describes a background about generative design synthesis systems. Section 3 provides details of the proposed approach in this chapter. Section 4 presents results and discusses the implications of results. And finally, Section 5 concludes the study and suggests further research projects to extend the presented work.

4.2 Approach

The overall approach to the tensegrity synthesis using generative graph grammars is depicted in Fig. 1. The synthesis phase consists of two steps; search and optimization. In the search phase (Fig. 1a), all valid topologies are generated in three consecutive steps and in the optimization phase (Fig. 1b) the dimensions and coordinates of the topology are optimized. Furthermore, based on the objective function values, one or more optimal candidates are selected. In the transformation phase (Fig. 1c) the generated topologies, which are represented as graphs are converted to 3D shapes. This step is not necessary for the synthesis of structures and is only used to visualize the final results as 3D shapes. In the following sub-sections all phases of the design are described in detail.

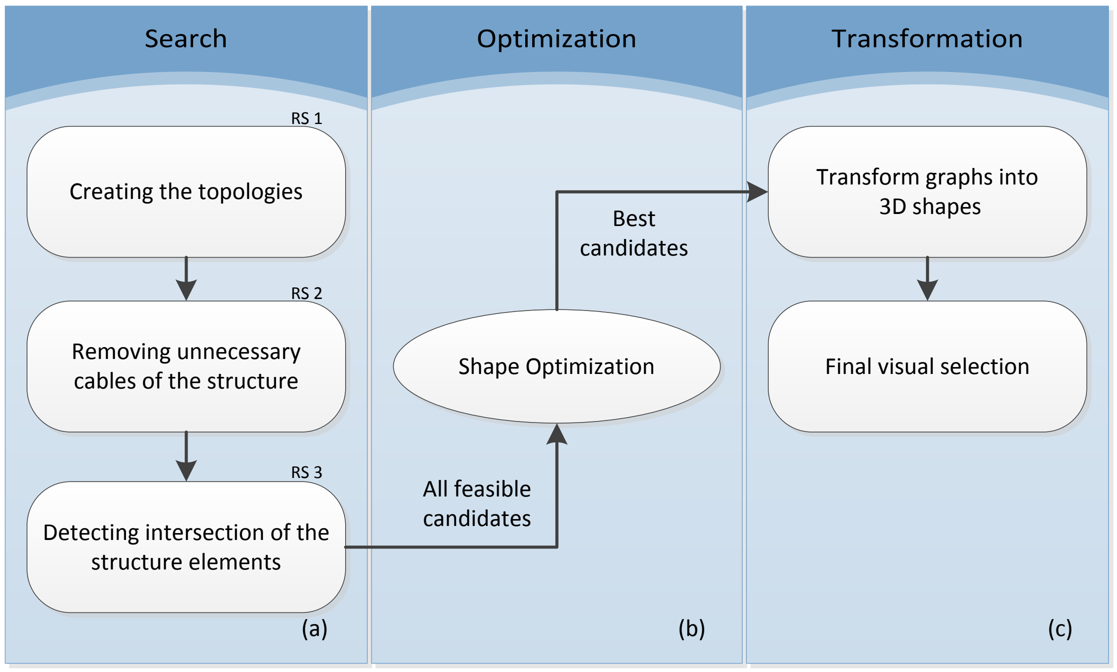


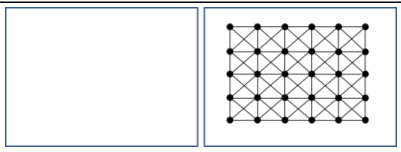
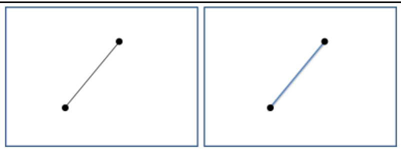
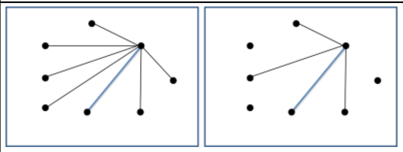
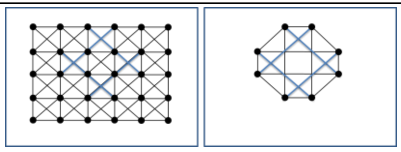
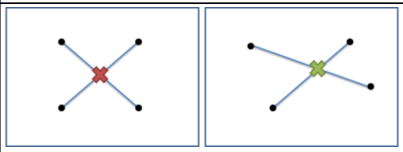
Figure 4-1: Structure synthesis approach

4.2.1 Grammar Rules

The graph grammar interpreter, which is used for the synthesis, starts with a seed graph, which represents the bounds of a specific problem. The generation (graph transformation) is carried out through six rules which are distributed into three rulesets (RS). A ruleset is a set of rules that transforms the design from one level of maturity to the next level. Rulesets are used as a means to compartmentalize different phases of the generation process. In Table 1 all grammar rules with a short description of each are illustrated. The rules are created in a general way, so that for different types of problems the same rules can be used. The left picture in the Rule column is the left hand side of a rule (LHS) and the right picture is the RHS of the rule. The graph grammar interpreter converts that part of the seed graph which is matched to the LHS into the graph segment depicted in the RHS. The trigger rule (rule 4) does not change the graph elements; therefore its LHS and RHS are not depicted. In the following three sub-sections, all rulesets including their rules are explained in detail.

Aside from the depicted rule conditions in Table 1 many other additional functions are compiled into the rules to define detailed matching conditions as well as rule action. For instance, for rule 6, one function aids in the recognition process to find the exact position of the intersections and one function helps to remove the intersections by changing the position of the nodes.

Table 1. Grammar rules

RS		Rule	Description	RS		Rule	Description
	1		Create the design space	1	4	Trigger rule 1	Trigger rule 1
1	2		Connect joints to joints and update the design space	2	5		Remove cables that are not necessary for the stability of the structure
	3		Detect stable tensegrities, which meet the required criteria	3	6		Detect intersections

A seed graph defines the scope and boundary conditions of the problem to be solved. In this case, our seed graph is empty, which means infinite possibilities. Because for any tensegrity type such as a T3 (a tensegrity with three struts), an infinite number of tensegrity positions exists, in which the tensegrity remains in a stable configuration. Stable tensegrity position configurations are the position of both sides of all struts in a tensegrity. Therefore, as can be seen in Table 1 the initial step is to reduce the design space from a space with infinite possibilities to a relatively limited field of possibilities. This is done through discretizing the design space into limited number of highly connected nodes through rule 1. The connectivity of the nodes is based on the minimum and maximum allowed size for the structure elements; it is a complete graph until two nodes are too close or too far apart. Fig. 2 illustrates a seed graph discretized into eighteen nodes in 2D and 3D views. The nodes are 250 millimeters apart in x- and y-directions and 550 millimeters in z-direction. The goal of other grammar rules is to transform this graph to a graph that represents a meaningful tensegrity structure.

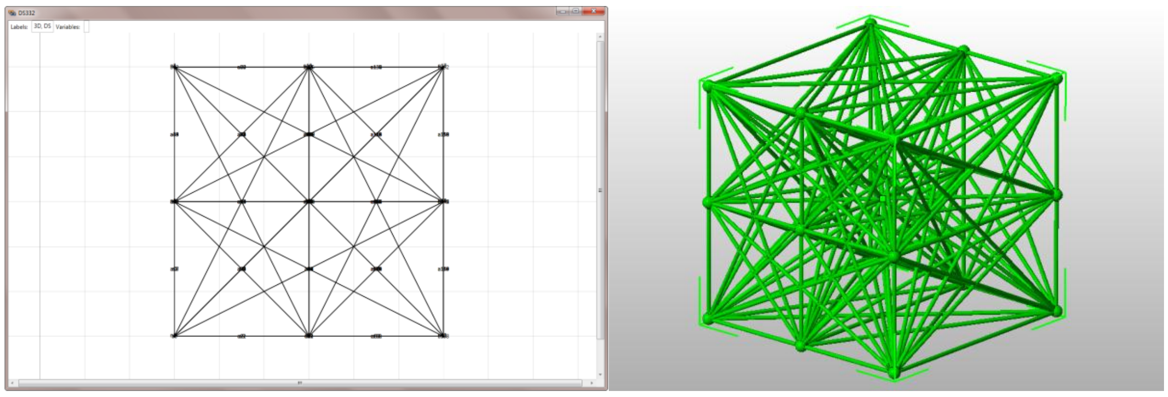


Figure 4-2: A seed graph with three eighteen nodes

Fig. 3 shows another spherical graph (design space). The radious of the sphere is 800 mm and both polar angle θ (theta), and azimuthal angle φ between the nodes are PI/8, which means the design space has 98 nodes. Maximum allowed distance for connecting the nodes is 700 mm and there is no minimum. One of the most important aspects of this research is that the exact spatial shape and boundaries of the design space, in which the tensegrities should be generated, are defined at the beginning of the synthesis process.

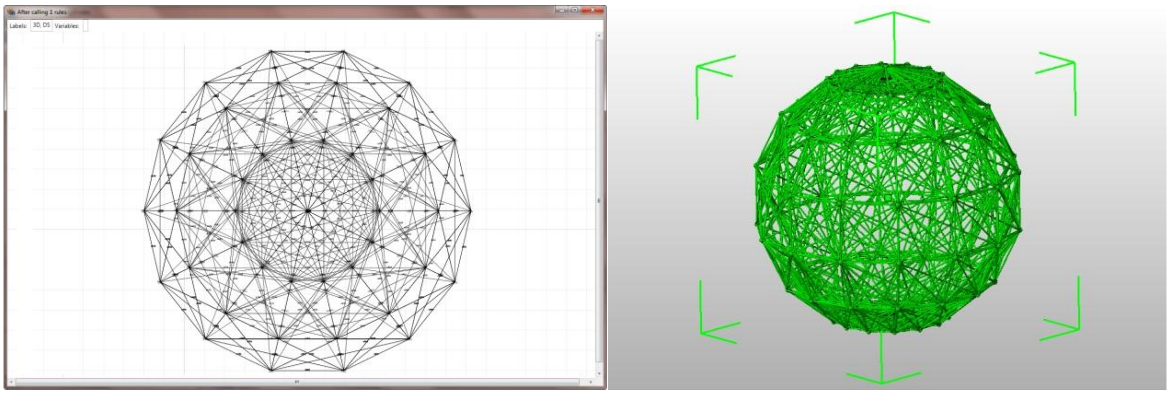


Figure 4-3: A spherical seed graph

After creating the design space, rule 2 is active. This rule can choose any arc in the design space for the first rule application. As soon as an arc is selected, this rule adds a “bar” lable to it, which means the arc will represent a strut afterwards. Furthermore, it removes all arcs that have an angle more than 90 degree with this strut (Fig. 4). It also adds a “cable” lable to all other remaining arcs connected to both sides of the strut. This lable changes the representation of the arcs to cable. For the next rule applications, rule 2 selects only those arcs, which deos not represent struts or cables and are connected to a cable from both sides. After finding the first stable T3 tensegrity structure, rule 2 selects only those arcs – to rerepresent as struts – which create a stable tensegrity configuration. Adding these limitations to the rule 2 in different stages makes the design space smaller but not the number of valid solutions. This rule is applied as many times as the desired type of the tensegrity is created. For example to create a T10 structure, ten times the rule should be called.

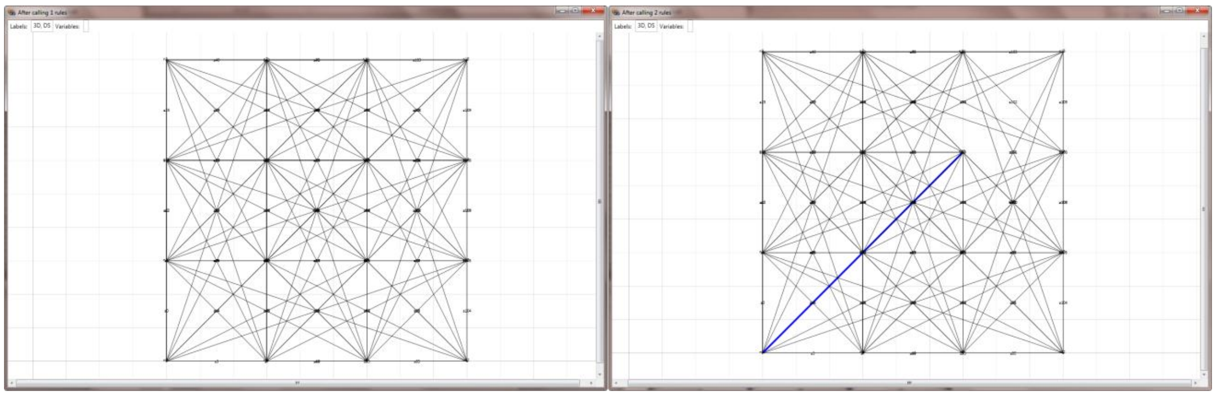


Figure 4-4: Effect of rule 2 on the design space.

After reaching the desired type of the tensegrity, rule 3 removes all unused nodes and arcs in the design space (Fig. 6). But this rule is called only if the created solution is stable. The criteria for checking the stability of the solutions are; a) each side of a strut should have at least 3 tensile strings and b) the angle between string and strut vectors. If the strut and string vectors are projected to xy, yz and zx planes (Fig. 5), on each plane we must have at least one string vector (α) with an angle more than and one string vector (β) with an angle less than the strut vector (γ). Maximum one of the strig angles is allowed to be equal to the strut angle.

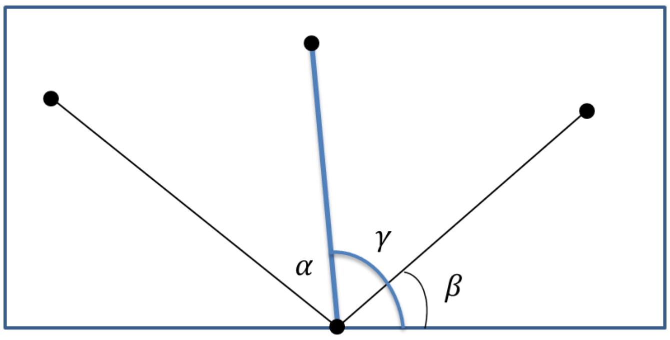


Figure 4-5: Angle between struts and strings

After applying the rule 3, rule 4 is recognised and allows the design to go to the next ruleset. Rule 4 filters out those candidate solutions, in which the stablity is not approved.

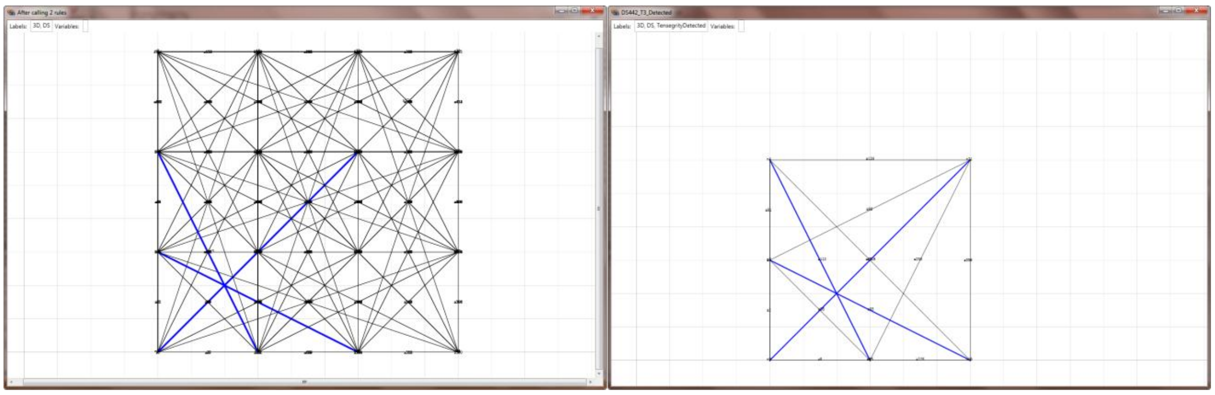


Figure 4-6: Effect of rule 3 on the design space after creating a T3

In ruleset 2, all unnecessary tensile strings for the stability of the solution are removed from the design space. Fig. 7 shows a T4 tensegrity with extra tensile elements, which has been removed in Fig. 8. This rule considers the minimum number of the cables and also stability of the structure for removing the tensile strings.

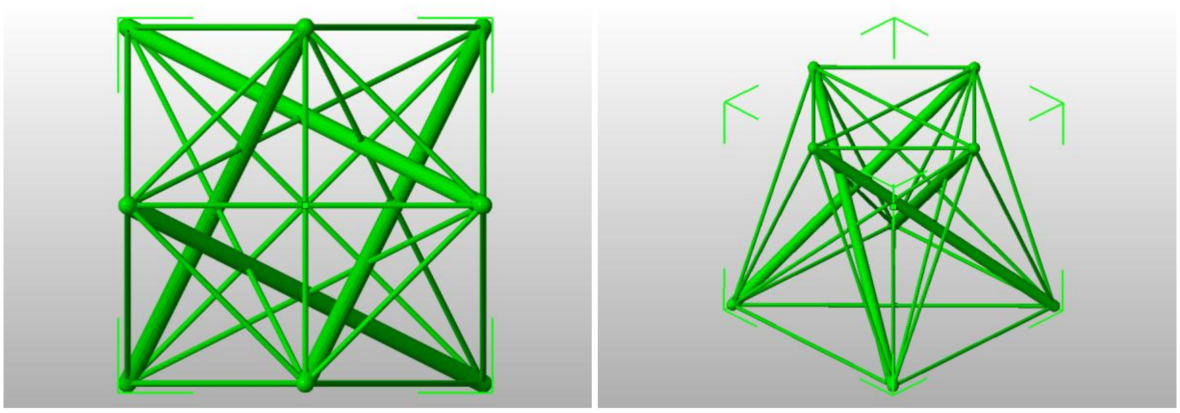


Figure 4-7: A generated T4 tensegrity with extra tensile strings

In this study we consider only the stability criteria to remove extra tensile strings. However, considering other criteria (structural analysis) helps to remove those tensile strings that are less utilized, which leads to a more stable and efficient designs.

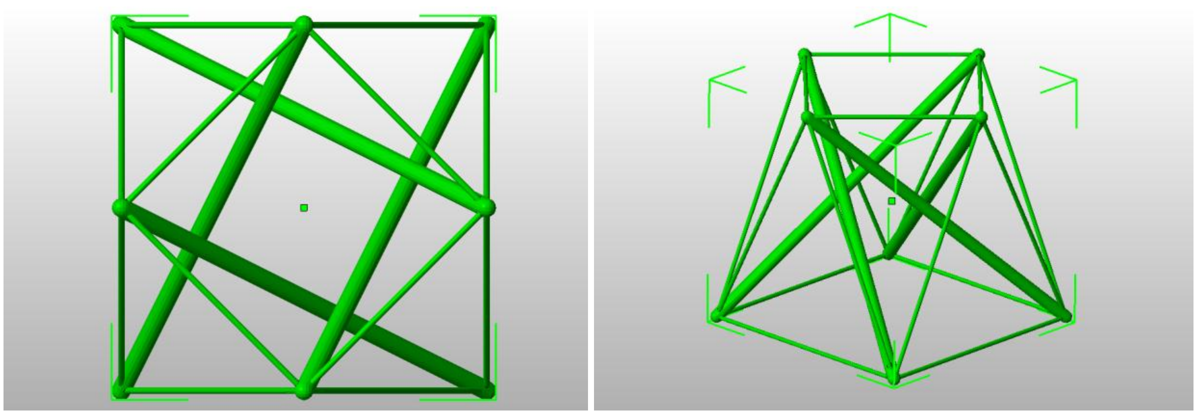


Figure 4-8: Removing unnecessary strings for the stability of the structure

After removing all extra strings from the structure, rule 6 detects elements of the structure that are intersecting each other. To remove intersections the end postion of the struts should be moved to another stable position configuration (Fig. 9). This final step of intersection removal is not implemented yet.

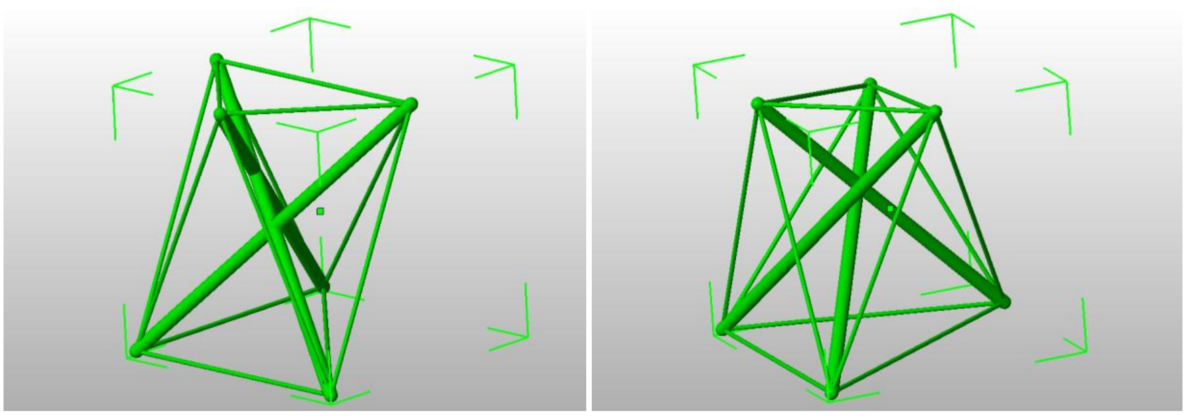


Figure 4-9: Detecting and removing intersection of the structure elements

4.2.2 Search

A depth first search algorithm has been used to search the design space for all valid candidates. Fig. 10 shows how the design space is explored through the tree-search algorithm. Indeed after creating the design space (rule 1), rulesets 1 and 2 expand the tree to its maximum size and generate automatically all possible topologies for the desired tensegrity problem. Ruleset 3 does not increase the number of solutions and just transform them to a modified state (intersection removal). In the final step of the tree-search, the shape of the solutions is optimized. In this step – like RS 3 – the existing candidates are just transformed to their final shape. The optimization is referred here, to coordinate its execution among the rules.

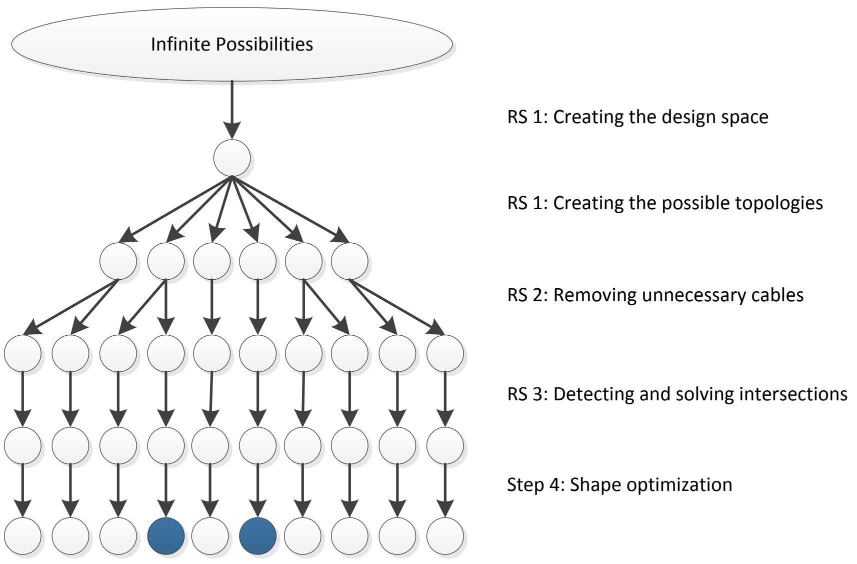

Infinite Possibilities
RS 1: Creating the design space
RS 1: Creating the possible topologies
RS 2: Removing unnecessary cables
RS 3: Detecting and solving intersections
Step 4: Shape optimization

Figure 4-10: Tree structure of creating structures

As it is possible to generate the same candidate through different sequence of rule applications, two mechanisms have been considered to prevent duplicate designs. The first mechanism is preventing confluent rules from being applied. The second mechanism is duplicate check. This algorithm compares the intermediate solutions with each other and removes those which are repeated. For instance it is possible to generate a T3 tensegrity with three struts s1, s2 and s3 in six different ways (based on the order that the struts are added). Although for generating small size tensegrities (up to T10 tensegrities), duplicates are not a menace for the tree-search algorithm, for solving large scale tensegrity problems they may drastically reduce the performance of the tree-search algorithm.

4.2.3 Optimization

The final step (step 4) is responsible for optimizing the shape of the tensegrity considering mechanical properties such as stress and buckling. This process consists of a non-linear gradient-based optimization known as “Fletcher-Reeves” algorithm (Nocedal and Wright, 2006). It can be used for defining the material and size characteristics of the strings and struts and also the most suitable position configuration of the structure elements. The objective may be the stability or any other desired criterion. Ongoing research of this study is developing this optimization algorithm along with the incorporating new analysis in the synthesis process.

4.2.4 Transformation

After automated syntheses of the topologies, they can be transformed into three dimensional shapes through the aid of a computational geometry package such as the Parasolid geometric kernel (Siemens PLM Software Inc., 2013). The transformation converts nodes in the graph into spheres, and arcs into cylinders. In this study the radius of all struts is fixed to 15 mm and the cables to 5 mmm. The shapes are saved as STL files. As discussed before, this transformation is only for 3D visualization of the results.

4.3 Results and Discussions

As discussed in the previous section the first step of the synthesis is creating a field, in which the tensegrities should be generated. This field may have a shape such as a cube or sphere, or any irregular form. The nodes can be randomly distributed throughout the design field or have a regular order such as those in Fig. 2 and 3. The approach to connect the nodes is that every node connects to every other node unless the minimum and maximum allowed length is violated. The more the number of nodes which are used to discretize the design space and the more these nodes are connected together, then the more the number of tensegrity structure topologies that can be generated. For instance for a design space with length, height and depth of 900, 900 and 500 millimeters respectively, which is discretized through 18 nodes (Fig. 2), more than 600 T3 structures are created through the application of the grammar rules. This set of 600 candidate solutions can be created in just under 7 minutes (using a desktop computer with an Intel(R) Xeon(R) processor). The struts and cables are allowed to have a length between 200 and 900 millimeters. This high number of possibilities is normal because the constraints are very loose; by applying more restrictions such as decreasing the size range of the elements the number of possibilities reduces drastically. Increasing the order of the tensegrity structure that should be generated in the same design space, reduces the number of options too. If instead of a T3 tensegrity, a T6 is desired, the number of options reduces to less than 50 solutions, one of which is depicted in Fig. 11.

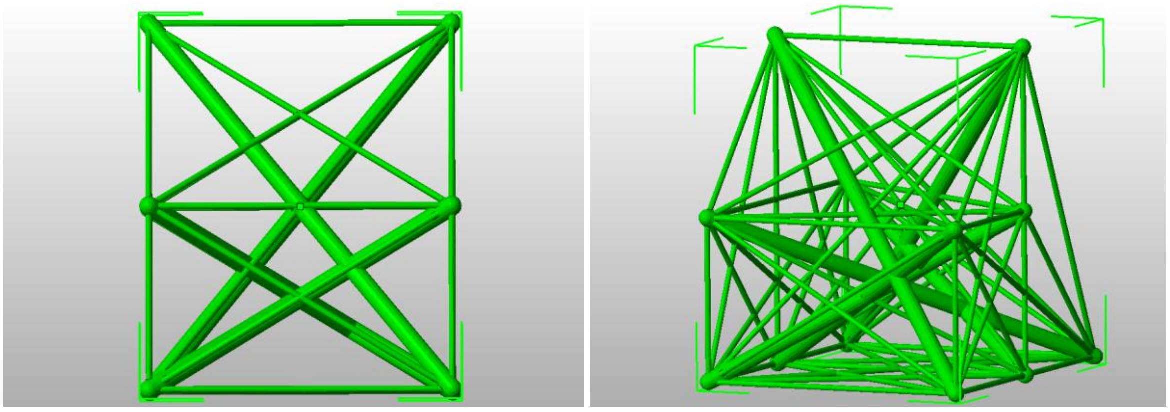


Figure 4-11: A T6 tensegrity created from a design space discretized into 18 nodes (2d and 3d views)

It is suggested to use at least four times as much node to discretize a design space as the type of a tensegrity. For instance for a T3 tensegrity, the design space is better to discretized into minimum 12 nodes. Because, using fewer nodes may eliminate many possible novel designs in view of the aesthetic and mechanical properties. For instance, Fig. 12 shows a T6 tensegrity structure, which has been generated in a design space with 32 nodes. This solution cannot be generated in the previous design space discretized into 18 nodes. Furthermore, the T6 structures generated in the former design space have more intersections than those in the later one. The unnecessary cables in both figures 11 and 12 are not removed to show that a solution created in ruleset 1, is indeed a set of tensegrity solutions with the same strut configurations but different configurations of the tensile strings. Depending upon other criteria, such as possible external loads, different set of tensile elements may be required.

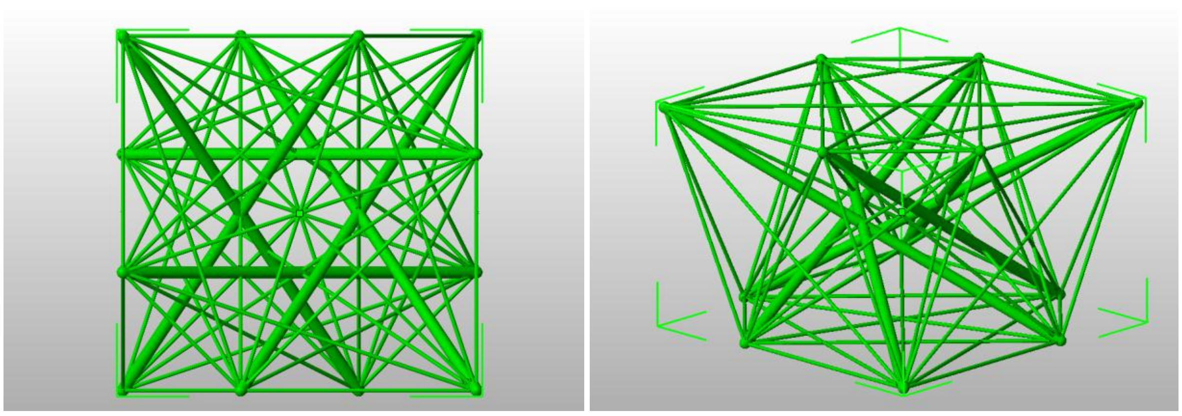


Figure 4-12: A T6 tensegrity created from a design space discretised into 32 nodes

Fig. 13 shows a T12 cylindrical tensegrity which has been created in a cylindrical design space, discretized into 24 nodes. The maximum number of struts that can be generated in a design space discretized into 24 nodes is twelve (e.g. a T12 tensegrity). This example is well known in the literature (Li et al., 2010; Zhang et al., 2014) but in most of these examples the initial topology (the description of the connectivity of the tensegrity structures) should be defined before finding the form.

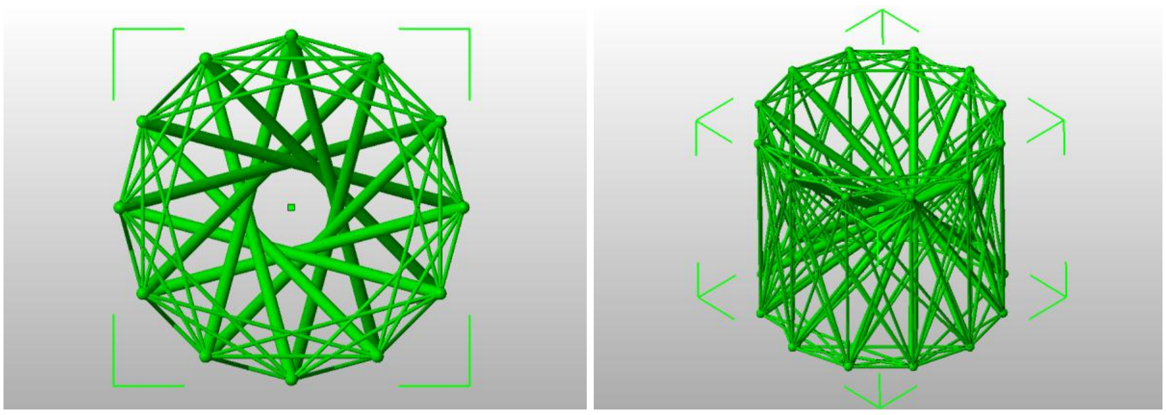


Figure 4-13: A T12 tensegrity created in a cylindrical design space

Fig. 14 shows a T34 tensegrity tower which has been created with the approach. The whole height of the tensegrity is 540 centimeters. This result was achieved after 30 seconds from starting the synthesis process. As we use a depth first search algorithm to explore the design space, the valid results are saved in as soon as they are created even before finishing the tree-search process. The whole process may take more than one hour, but the first results are normally achieved in the first few minutes.

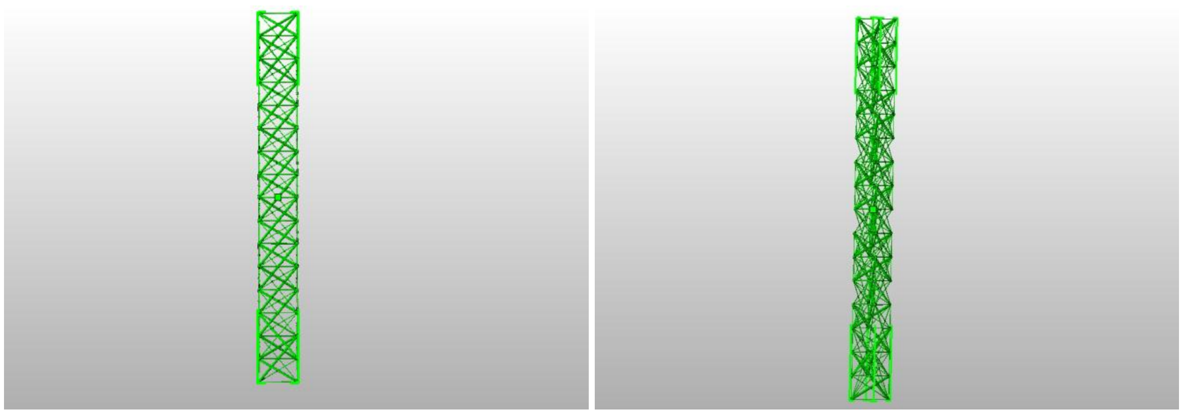


Figure 4-14: A T34 tower created from a cubic design space with 72 nodes

The same synthesis process is carried out with the same rules for a large scale irregular problem. The only difference is in the time that is required to generate the solutions. For instance, Fig. 15 shows a T30 tensegrity which has been generated from the spherical design space, illustrated in Fig. 3. This structure along with many other solutions was created in less than 10 minutes from starting the synthesis process. Although there may exists hundreds of stable T30 tensegrity solutions in the aforementioned spherical field, exploring the whole design space was not the aim of this example.

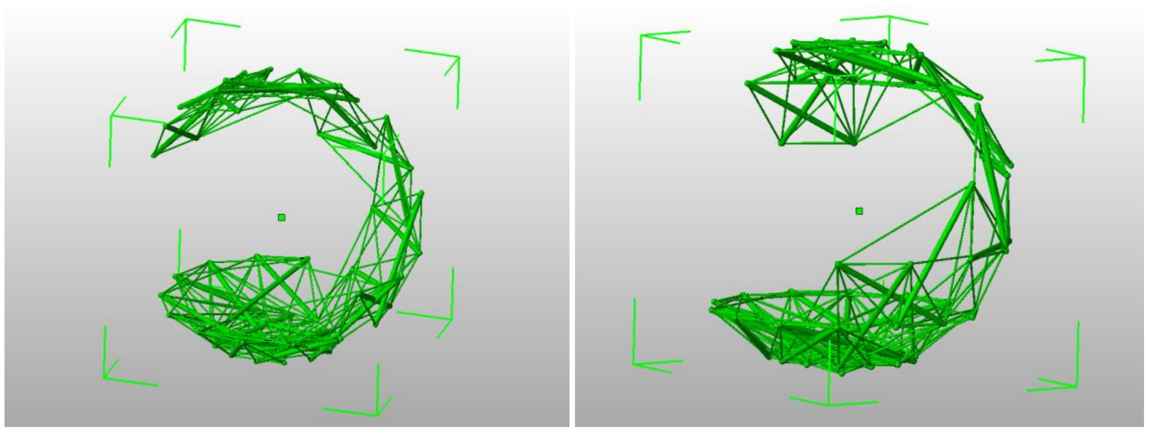


Figure 4-15: A T30 tensegrity created from a spherical design space

Fig. 16 shows the lower part of Fig. 15 which consists of eleven struts (a T11 Tensegrity). This example shows the flexibility and stability of the approach in coping with different design spaces. Interestingly enough, Fig. 15 is not only statistically stable; it shows a novel tensegrity structure, which has not ever been seen in the literature. This shows the abilities of the approach in generating many novel solutions for any type of design space.

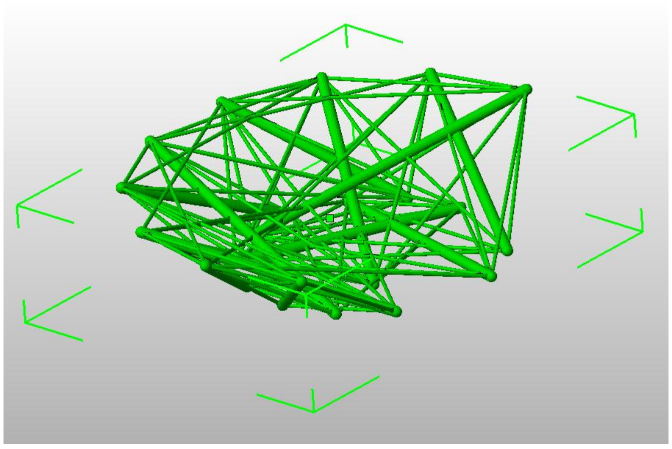


Figure 4-16: Detailed view of the base of fig. 15

4.4 Conclusions

A new approach for form-finding of tensegrity structures using generative design methods is presented. This approach uses graphs to represent both the topology and the shape of structural layouts. It allows a very fast generation of topological solutions for a given design problem. A depth first search algorithm is used to explore the design space. The solutions are stored in a list for further detailed shape design and optimization. To visualize the 3D solutions, the graphs are converted to 3D shapes via Parasolid. The design space can have any shape and size and it is possible to solve large scale irregular problems with any number of struts and cables.

Ongoing research in this study is focused on developing the shape and size optimization algorithm. Developing the optimization process along with new static and dynamic analysis methods will help to not only automatically synthesis topologically valid solutions, but also creating solutions which are optimized considering mechanical properties. Automating the intersection removal algorithm in ruleset 3 prevents filtering them out, because in the current implementation, based on the number of intersections the candidates are stored in a sorted list. And finally developing more appropriate search algorithms to explore the large design spaces can help in reaching many novel solutions in less time. The reason for this investigation is that in very large design spaces, the efficiency of the tree-search algorithm plays an important role. This investigation should focus on developing mechanisms – such as duplicate check mechanisms – which further reduce the solution space. For instance a third duplicate check mechanism might be used to remove not only those duplicate designs which have exactly the same position configurations in the absolute coordinate system, it should be able to detect those duplicates which their elements relative positions are similar.

5. Computational Design Synthesis (CDS) Platform

An important obstacle in the development of computational synthesis tools in engineering design is the difficulty in integrating the generation process with efficient simulation packages for evaluating candidates in a search process. The premise of this study is to develop and implement a platform to facilitate generative design systems in achieving more flexible design synthesis automation and optimization. This enables the designers to explore the abilities of generative design systems rather than coping with complexities of automatically integrating these analyses in the design process. The platform has been developed mainly based on open source software (OSS) to be offered to the Computational Design Synthesis (CDS) community for further development, use and investigation. Its modularity and programming based implementation provides a foundation for other researchers to build on and to achieve the next generation of CAD tools substantially faster.

Keywords: Computational Design Synthesis, Design Automation, Open Source Software, Multi-physics Simulation, Finite Element Methods, Shape Grammar, Graph Grammar

5.1 Introduction

In engineering design, complex analyses (such as FE, CFD and thermal analysis) are required for accurately predicting the engineering behavior of generated designs. Automatically integrating these analyses is a known challenge for engineers and designers. Aside from the inherent difficulties and the large amount of time typically required for embedding external software packages in the automated synthesis process, doing this in a generic yet robust way is even more complex. Thus, an important obstacle in the development of computational synthesis tools in engineering design is the difficulty in integrating simulation packages with the generation process (Bolognini et al., 2007).

Many scientists have tried to link shape grammars with a simulation model to evaluate the performance of the designs and guide the search process. Shea and Cagan have used FEM analysis to evaluate the performance of generated trusses and frames and guide the generation process (Shea and Cagan, 1998). Starling and Shea have used the behavioral modeling language “Modelica” to evaluate camera winding mechanism designs generated by the parallel grammar (Starling and Shea, 2005). Bolognini et al. has coupled COMSOL multi-physics analysis with a synthesis method to generate MEMS (Bolognini et al., 2007). However, all these examples are not general and have been developed only for one specific application, because coupling a simulation model robustly with design generation even for only one application is a complex task. Indeed the novelty of the presented platform lies in its generality.

The premise of this platform is to facilitate generative design systems, such as shape and graph grammars, in achieving more flexible design synthesis automation and optimization. This enables the scientist to explore the abilities of generative design systems rather than coping with complexities of automatically integrating these analyses in the design process. The CDS Platform uses several open source software, such as Salome (Open CASCADE, 2013), Code Aster (R&D, 2013), Open Foam (OpenCFD Ltd (ESI Group), 2013), FreeCAD (Riegel et al., 2013), SnappyHexMesh (OpenCFD Ltd (ESI Group), 2013) to perform a variety of multi-physics simulations. Unlike previous implementations, using open source software (OSS) in developing the platform enables us to offer the platform to the Computational Design Synthesis (CDS) community for further development, use and investigation. Its module and programming based implementation provides a powerful base for researchers to build their work on and help to reach the next generation of CAD tools faster.

This chapter is organized as follows. The second section presents relevant open sourcing issues and its uses in product development. The third section presents the developed platform for the field of Computational Design Synthesis. In this section the system architecture and main applications are discussed. The fourth section is devoted to simulation friendly design synthesis. In this section, smooth integration of simulation with generative design through adequate development of design rules and design spaces is discussed. In the fifth section various application domains are illustrated and future outlooks are presented. Finally, the last section contains conclusions and discussions.

5.2 Open Sourcing

The appearance of open sourcing began in the 1960s, after the computer manufacturers decided to separate hardware and software, which provided the opportunity to develop software independently from the hardware (Hertel et al., 2003; Khanjani and Sulaiman, 2011). The academic community, led by the University of California at Berkeley, defined a Unix-based Berkeley Software Distribution (BSD) that eventually lead to the Open Source Initiative (OSI; 1998). The OSI defined open source as code that: is distributed freely, can be modified freely, and is accessible to a large number of developers through the Internet. According to Deshpande & Riehle (2008), the growth in open source doubles almost every year in term of the projects and the number of lines of code. There are several open source software properties that have advantages when used for product development (Ruffin and Ebert, 2004). Open source projects have a longer lifespan, heed standardized interfaces and are easier to integrate with other software tools.

5.3 CDS Platform

To address the challenges in design synthesis and robust coupling with simulation methods, a new platform has been developed to increase the role of computers in generating alternative designs and exploring solution spaces for engineering problems. It introduces an approach that combines shape and graph grammars with conventional simulation and analysis methods to provide guidance in design engineering according to evaluated engineering criteria. The major characteristic of the presented platform that distinguishes it from all other implementations is its generality. To achieve this generality and flexibility, a programming and module-based approach has been adopted in developing the platform.

5.3.1 A module / programming based platform

Due to the complexity and the wide range of possible applications it is not feasible to develop a software that can include all possible simulation scenarios. Instead, the CDS Platform takes the approach that enables the users to develop the corresponding synthesis processes on their own. This is supported in the form of a programming framework in Python. This means it provides an interface so the user can implement the design synthesis process. By this approach multiple forms of control flow are supported and various ways of creating macros are provided, for example in form of functions or object-orientation.

5.3.2 System Architecture

The platform architecture is inspired on the general synthesis cycle illustrated in Fig. 1. For every stage of the process, one must define the corresponding modules. Additionally, some functionality is necessary to convert the output data of each step to fit the requirements of the next step. Modules have access to any information created or available in other stages, because the information is stored central in text format.

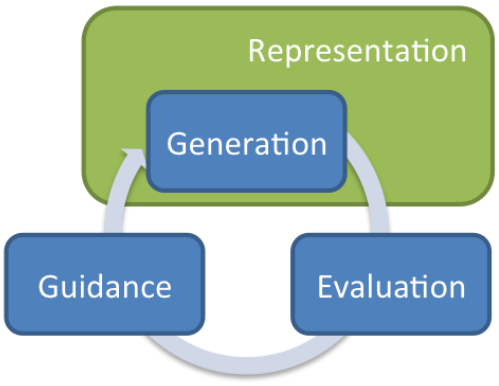


Figure 5-1: General synthesis process, modified from (Cagan et al., 2005)

The components of a CDS framework are presented in Fig. 2. For each step there are usually multiple modules that can provide the required functionality, so the platform is not limited to single tools or processes. In the following subsections, each of the seven module-types is briefly described.

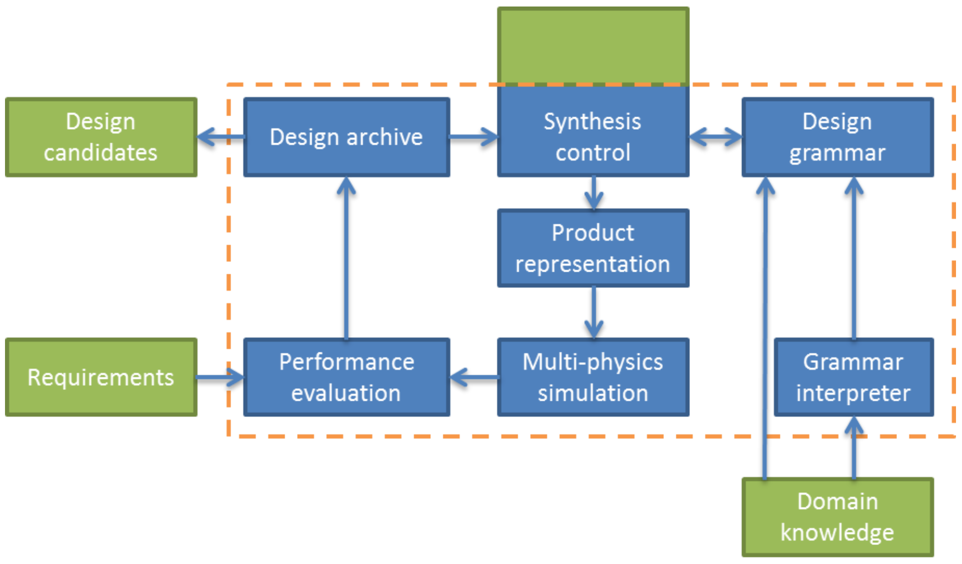


Figure 5-2: Components of a CDS framework modified from (Helms et al., 2009)

For **multi-physics simulation**, the platform integrates Code-Aster (R&D, 2013) for Finite Element analysis (FE) and OpenFOAM (OpenCFD Ltd (ESI Group), 2013) for Computational Fluid Dynamic (CFD) and thermal analysis (as part of the evaluation module). The necessary converters and preprocessors needed for these tools (Salome and snappyHexMesh) are integrated in the platform as well. By combining these sets of different preprocessors and solvers, multi-physics analysis of candidate solutions is possible. Many different criteria were considered for choosing these sets of solvers and preprocessors. The main criterion that has a direct effect on the synthesis process was the quality of results. Aside from thousands of tests, which have been carried on through developers of the software, a brief search in the literature revealed that many researchers in different disciplines have used these tools to accomplish their scientific research such as (Silva and Lage, 2011) and (Lou et al., 2010). The second important reason for selecting these tools was their open-source nature that facilitated the integration with the developed CDS Platform. Open access to the source code was of vital importance for developing a generic CDS Platform that unlike other implementations in this field is not restricted to any type of simulation or design. Due to a free licensing access to these analysis tools, the developed CDS Platform can be offered to the CDS community for further development, use and investigation.

The **performance evaluation** itself is realized by gathering information from the whole synthesis process (mainly from simulation) and combining it into a single objective value by the means of an *aggregating function* that does a weighted addition of all collected data (Wang et al., 2008).

The **synthesis control** is mainly a task of the user due to the nature of a programming framework. But the platform provides an easy access to a lot of generation approaches like optimization, search trees and knowledge-based processing. As the user influences the control flow, he can easily integrate additional approaches like Genetic Algorithm (GA).

As a simple **grammar interpreter**, the CDS Platform can access a shape grammar interpreter that is introduced by Hoisl and Shea (2011). For the **representation** the *FreeCAD* file format is used in this case. To support more sophisticated designs and grammars, the platform will be integrated GraphSynth as a graph grammar interpreter developed by Campbell (2013). It has been used by researchers such as Kurtoglu et al. (2010) and Rai et al. (2011). As the results are **represented** in graphs, they must be transformed into shapes. This task is carried out through integrating commercial or open-source CAD kernels (e.g. Parasolid, ACIS or OpenCasCade).

5.3.3 Integrated tools

Multi-physics simulation tools

Code-Aster is an Open Source software package for finite element analysis and numerical simulation of structural mechanics and Civil and Structural Engineering. It has been developed by a French company (EDF) as an “in-house” software (R&D, 2013). Code-Aster was released as free software under terms of the GNU GPL in 2001. Code-Aster is mainly a solver for mechanics, based on the theory of the finite elements (FE). This tool covers a large range of applications: 3D thermal analysis and mechanical analysis in statics and dynamics, for machines, pressure vessels and civil engineering structures. Beyond the standard functionalities of the software for solid mechanics, Code-Aster compiles specific research in various fields: fatigue, fracture, contact stresses, geo-materials, porous media, and multi-physics coupling. The **Salome Platform** can be best coupled with the Code-Aster solver to effectively preprocess the geometries. Salome is an open source software platform which has been started in 2001 and distributed with the GNU LGPL license. It provides a generic pre- and post-processing tool for numerical solvers. 3D solid shapes are transformed into tetrahedron or hexahedron meshes in the mesh module to be prepared for finite elements analysis. Post-processing module of the Salome allows importing and analyzing calculation results generated by CAE solvers (Open CASCADE, 2013).

OpenFOAM is an open source CFD software that has been developed by the OpenFOAM Team at SGI Corp. OpenFOAM can be used for solving different problems in areas of engineering and science from complex fluid flows involving chemical reactions, turbulence and heat transfer, to solid dynamics and electromagnetics (OpenCFD Ltd (ESI Group), 2013). The latest application of OpenFOAM also includes stress analysis, large strain analysis and magneto-hydrodynamic flows (Karac, 2003). It has a large user base across both commercial and academic organizations. OpenFOAM includes tools for meshing, notably snappyHexMesh, a parallelized mesher for complex CAD geometries, and for pre- and post-processing. **SnappyHexMesh** generates 3D hexahedra meshes from a triangulated surface geometry in STL format (Ribes and Caremoli, 2007).

Synthesis control tools

In the generation process defined by a grammar a decision must be made among options which include a location within the candidate (e.g. a subshape or subgraph) and a rule that modifies that location. Two main mechanisms have been developed to guide the generation in a systematic way:

- Tree-search: the state of the current generation process (including all existing candidates) is stored in a tree structure. To apply the next rule (including choosing a candidate shape and a rule) one of the tree search algorithms such as depth-first or breadth-first is used.
- Iterative mechanism: only two solutions are saved, the current solution and the best solution, and the space can be traversed randomly or by following gradients. To date, simulated annealing algorithm has been developed for guiding the process.

These two mechanisms have positive and negative aspects that should be discussed extensively. For instance, the tree search mechanism increases the chance to reach the best solution but it is time consuming. An iterative mechanism like simulated annealing algorithm does not search the whole design space, but its efficiency to find optimally directed solutions (in designing frames and trusses) has been shown by Shea (Shea and Cagan, 1998). Assessing different aspects of guidance mechanisms (tree-search and guided mechanisms) and comparing their results is not covered in the scope of this study and requires further investigation. **SciPy** is another open source optimization toolbox which has been integrated in the platform to be used in the guidance process (SciPy Developers, 2013).

Grammar interpreters

A 3D shape grammar interpreter developed by Hoisl and Shea (2011) has been fully integrated in the platform and the integration of a Graph Grammar Interpreter (GraphSynth) developed by Campbell (2013) is under development. The grammar interpreters are used for describing solution spaces and generating design alternatives. They allow both interactive and automatic generation of alternatives.

The main criteria to select the 3D shape grammar interpreter are as follows: support of 3D shapes, parametric shape grammars, transformations, shape types, definition/manipulation of rules, user friendly interface, and the capability to both execute shape rules automatically as well as interactively. These criteria have been discussed by Hoisl and Shea (2011). The shape grammar interpreter has been developed within the FreeCAD environment. FreeCAD is an open source software distributed under GNU GPL and LGPL license that supports parametric 3D building of volumetric models (Riegel et al., 2013). The software supports several import/export document formats. To make drawing in FreeCAD convenient, scripting in Python is added to the software that allows users to create and modify geometries effectively.

GraphSynth is a unique research software for creating, editing, displaying, and manipulating generative grammars. This framework stores graphs, rules and rulesets in an XML file format. This allows automatic search for creative, optimal or targeted solutions. Additionally, it is able to perform various graph transformations such as the double-pushout method and free-arc embedding; these two together cover nearly all types of required graph transformations (Campbell, 2013). One of the most important characteristics of the GraphSynth is its expandability; through additional C# functions (compiled on-the-fly by GraphSynth) any capability can be added to the rules and rulesets.

Data exchange

The data flow in the platform is highly dependent on the use case or synthesis process because the user is mainly responsible for the control flow. But there is a common pattern in data flow that is in use while communicating with the integrated tools (for each module). The general approach to communicate with any of the integrated tools is to export a file, which is used as input to the subsequent module. As next, the application is called by its command line interface to generate the necessary output. This output needs to be parsed (usually by the means of regular expressions) so it can be passed backed to the platform. This pattern is generic, platform and language agnostic, but it has its drawbacks primarily in maintainability, because the input and output specifications of the integrated tools could change in newer versions and it is harder to debug it than a software using a programming interface. Nevertheless it is the only option, as most of the tools do not provide an API in Python.

5.4 Applications

Applicability of the platform is directly dependent to the grammar interpreter abilities to generate new design solutions. The grammar interpreter defines not only the type of the problem which can be solved but the richness and quality of the solutions connected to it. In the following sub-sections, the applications are discussed that have been built using CDS Platform.

5.4.1 Shape synthesis for axisymmetric problems

The first grammar interpreter that was integrated in the platform was a shape grammar interpreter, developed by (Hoisl and Shea, 2011). They have illustrated various design problems for their 3D shape grammar interpreter including cooling fins grammar and wheel rims grammar. Through integrating the grammar interpreter in the platform, both of these problems are solvable with the platform and have been extensively discussed by the author in (Hooshmand et al., 2012). In Fig. 3, different components, which have been used in the CDS Platform for wheel rim synthesis, are illustrated. Although the generated design solutions with the shape grammar interpreter are novel concerning topological aspects, due to the nature of the interpreter –which relies only upon primitive shapes such as boxes and cones for the representation–, the results are not industrially applicable or valuable. However, the generated results by the shape grammar interpreter point to the future possibilities in the field of CDS. Specially, through using the platform in the future projects, the researchers will not have to struggle with integrating complex simulation models in their implementations. While currently limited in terms of the types of shapes that can be defined, future improvements in shape grammars would lead to more complex shapes.

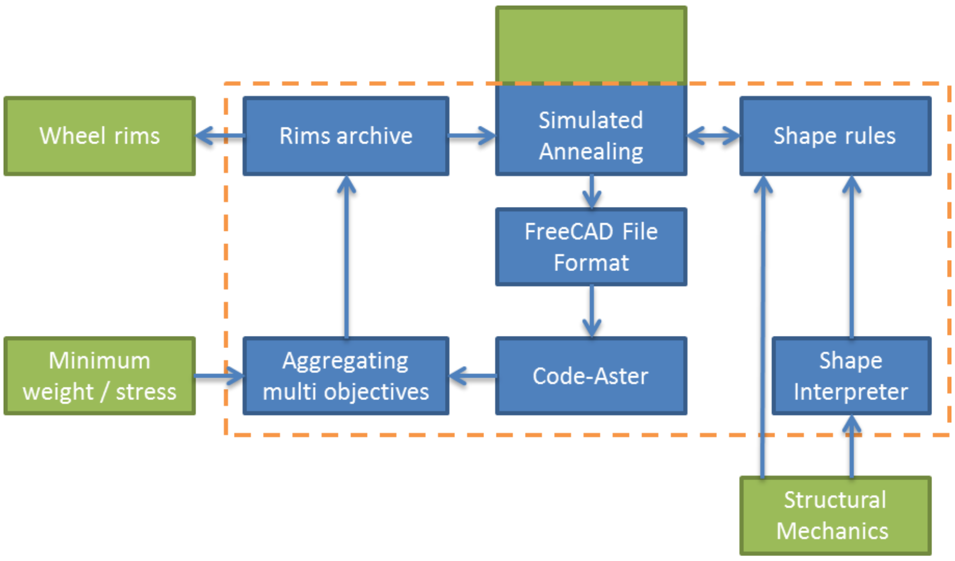


Figure 5-3: CDS Platform components for wheel rim synthesis

5.4.2 Fluid channel synthesis

Optimization of fluid channels is an essential topic in designing microfluidic devices (Andreasen et al., 2009; Vangelooven et al., 2010). The goal is mainly to find an optimal topology for the fluid subdomains along with an optimal shape of channels (Liu et al., 2010). Borrvall and Petersson (2003) used for the first time topology optimization for solving fluid problems in stokes flow. Since then, many scientists have used various grid-based topology optimization methods to solve fluid layout problems. One of the major limitations, which topology optimization methods in conceptual design are facing, is limited representation power; the synthesis process and design rules are dependent and integrated into the simulation model, the simulation model is often fixed for a given set of loads and boundary conditions (Hooshmand et al., 2012). The process of solving a layout problem, even for simple 2D is very time consuming. The topic of shape and topology optimization of fluid channels with graph grammars approach has been extensively explored in chapter 2.

Through combining the generative abilities of GraphSynth with the CDS Platform, we are overcoming the limitations of current methods. GraphSynth creates the topology design and OpenFOAM evaluates the candidates. Fig. 4 shows the CDS Platform components for solving fluid channel layout synthesis. As can be seen the components are almost fully different from those illustrated in the Fig. 3 for solving wheel rim synthesis, which shows the flexibility of the CDS Platform.

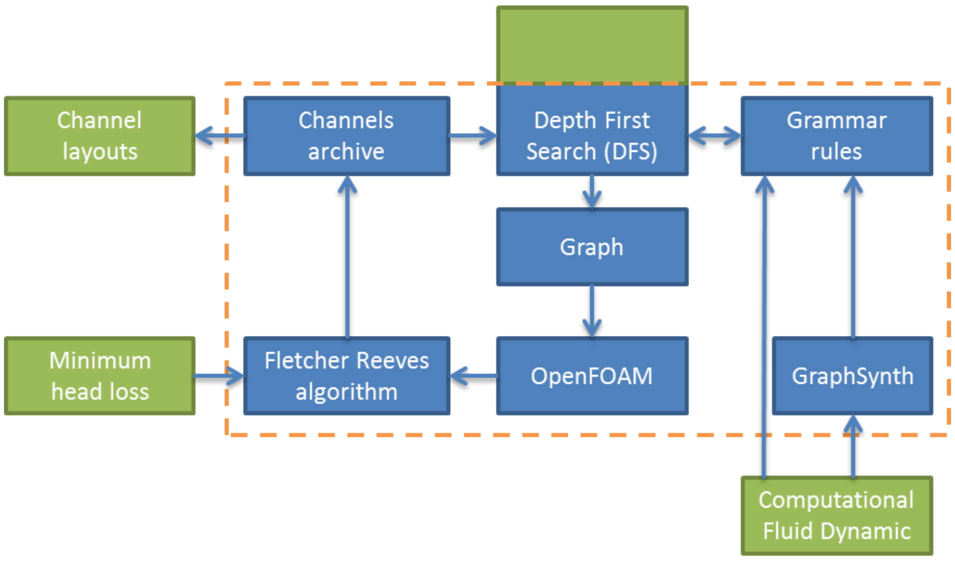


Figure 5-4: CDS Platform components for solving fluid channel layout synthesis

Fig. 5 shows the CDS Platform components for solving space frames and tensegrities synthesis. Similar to Fig. 3 and 4, the components of the platform are partly changing but the main frame remains the same. In chapters 3 and 4 the synthesis results of this CDS configuration for solving trusses and tensegrities are discussed. However in chapter 4, there is no structural analysis carried out.

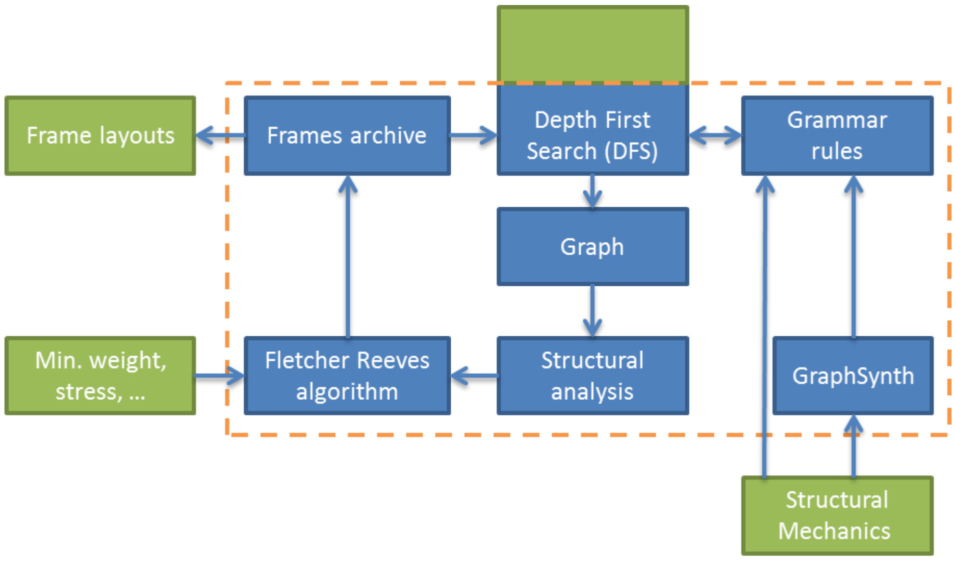


Figure 5-5: CDS Platform components for solving frame structure synthesis

5.4.3 Lightweight design of a triangle

Another interesting field of application that the platform can be used is parametric optimization of designs. For solving this problem, no grammar interpreter is required since all solutions have the same topology. The platform is able to cope with complicated designs with numerous parameters. The aim of this case study is to find the optimum lightweight design for a triangle with four parameters to be optimized (P1, P2, P3 and Θ). Unlike the previous two applications, a variety of recent commercial CAD packages are able to cope with this kind of problem. In this study the Code-Aster FE solver, Salome Preprocessor and a simulated annealing algorithm have been used.

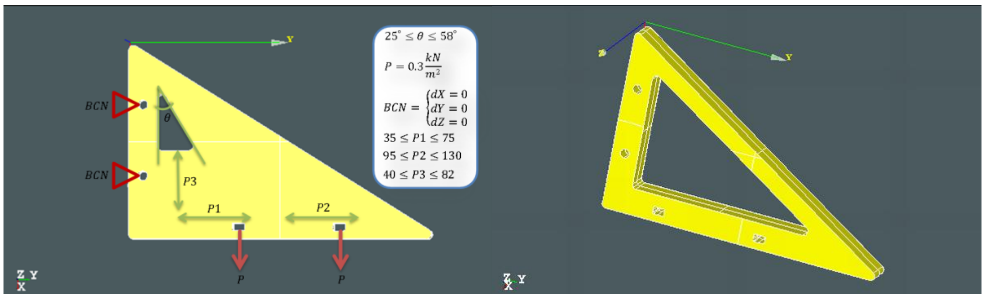


Figure 5-6: Forces, BCNs, and parameters, and best design after 990 iterations with Simulated Annealing

Fig. 6 shows the boundary conditions and forces which are applied to the geometry and also four parameters which should be optimized (P1 to P3 and theta). The objective function for this case study is minimizing weight and stress of the triangle; objectives have different weighting factors. For meshing the geometry an automatic tetrahedralization algorithm with mesh size 5 is used. A linear statistic solver of Code-Aster is used to analyze designs after each optimization iteration. Fig. 6 shows the best design after 990 iterations.

5.5 Conclusions

The Computational Design Synthesis (CDS) Platform has been developed to increase the role of computers in generating alternative designs and exploring solution spaces for engineering problems. It introduces an approach that combines generative design methods such as shape and graph grammars with conventional simulation and analysis methods to provide guidance in design process according to evaluated engineering criteria. The major characteristic of the presented platform that distinguishes it from all other implementations is its generality. To reach this generality and flexibility, a programming and module-based approach is used to develop the platform. The CDS Platform combines different optimization and grammatical algorithms with conventional simulation and analysis methods. The premise of this combination is to create an approach to synthesizing optimal shapes considering criteria requiring multi-physics analysis, which is required for calculating the engineering behavior of generated designs. The platform can be used in a very wide range of simulations and analyses like acoustics, finite element, computational fluid dynamic, and heat transfer and a combination of these analyses to solve complicated multi-physics problems. This has been achieved by integrating two preprocessors and solvers in the generation process; the Salome preprocessor and the Code-Aster solver for FE analysis and the snappyHexMesh preprocessor and the OpenFOAM solver for CFD and thermal analysis. Automatically integrating these analyses in the design process is a known challenge for engineers and designers. Unlike many commercial software, its object-oriented and module based implementation provide a unique possibility for designers to integrate any simulation module of the platform in their design processes in a few simple steps. The platform works like a high level API and prevents the direct interaction of designers with many complexities of the simulation and optimization packages. Its open source character gives the researchers the ability to extend, modify and customize the platform to their needs.

6. Conclusions

A new approach for solving engineering design problems using generative design methods is presented. The work introduces an approach for using design information and knowledge based on various classes of knowledge levels. These include: general knowledge as the most abstract level of knowledge, generic knowledge, specific knowledge, and case knowledge as the most concrete level of knowledge. In order to effectively utilize design information and knowledge, the design process is divided into three main phase: search, optimization and modification. For more abstract levels of design, which happens in the search phase, higher level knowledge is required and applicable, but in latter phases more knowledge levels are applicable. In chapters 2, 3 and 4 three design problems, which have been solved based on the proposed approach, are presented. In all design cases graphs are used to represent both the topology and the shape of structure layouts. This allows a very fast generation of topological solutions for a given design problem.

In chapter 2, the approach for shape and topology optimization of fluid channels is presented. Based on results of two optimization functions, the best solutions are stored in a list for further detailed shape design. The simulation model is fully separated; therefore it is possible to solve problems such as that have compressible fluids with high Reynolds number and arbitrary flow directions at inlets and outlets. Large scale problems, problems with more than one fluid type, for which the mixing is to be avoided, are also solvable. An interesting field of research in this area might be to use the output results of the approach as input for conventional topology optimization methods. Due to a good initial design, convergence can be faster and many problems might be solved that are hitherto not solvable.

Chapter 3 introduces the approach for shape, size and topology optimization of cable truss structures. A tree-search algorithm is responsible for exploring the whole design space and generating all valid solutions, while the optimization is responsible for the shape and size optimization of each topology. In this problem, the simulation model is fully separated; therefore it should be fairly easy to augment the approach to solve problems such as those with seismic loads, uncertainty in materials and construction with any number of loads and supports. An interesting investigation may be using the approach for other domains such as compliant mechanisms design. Like previous design problem, using the output results of the approach as input for conventional topology optimization methods can be considered.

The third design problem, which is form-finding of tensegrity structures, is presented in chapter 4. A depth first search algorithm is used to explore the design space. To visualize the 3D solutions, the graphs are converted to 3D shapes via Parasolid. The design space can have any shape and size and it is possible to solve large scale irregular problems with any number of struts and cables. An important extension for this work may be developing a shape and size optimization algorithm. Developing the optimization process along with new static and dynamic analysis methods will help to not only automatically synthesis topologically valid solutions, but also creating solutions which are optimized considering mechanical properties.

Finally in chapter 5, the Computational Design Synthesis (CDS) Platform which has been developed to increase the role of computers in generating alternative designs and exploring solution spaces for engineering problems is discussed. It introduces an approach that combines generative design methods such as shape and graph grammars with conventional simulation and analysis methods to provide guidance in design process according to evaluated engineering criteria. The major characteristic of the presented platform that distinguishes it from all other implementations is its generality. To reach this generality and flexibility, a programming and module-based approach is used to develop the platform. The platform can be used in a very wide range of simulations and analyses like acoustics, finite element, computational fluid dynamic, and heat transfer and a combination of these analyses to solve complicated multi-physics problems. Besides increasing the software maturity, future work may include the integration of graph grammars interpreter GraphSynth in the platform.

The results of the engineering design cases show the generality and flexibility of the proposed framework. Besides aforementioned future works, improving the search strategies for exploring the design spaces is very important to achieve faster results in larger design spaces. Another important field of research is automatic capturing of generic and general levels of knowledge in the grammar rules and creating a data base of these captured rules. This will help designers to understand the real design problem at all abstraction levels easier. The framework is flexible enough and independent of the problem domain and type, therefore the approach can be used for other domains and other design problems.

7. References

Aage, N., Poulsen, T.H., Gersborg-Hansen, A., Sigmund, O., 2007. Topology optimization of large scale stokes flow problems. Struct. Multidiscip. Optim. 35, 175–180.

Achtziger, W., Bendsøe, M.P., Ben-Tal, A., Zowe, J., 1992. Equivalent displacement based formulations for maximum strength truss topology design. IMPACT Comput. Sci. Eng. 4, 315–345.

Aichholzer, O., Aurenhammer, F., Alberts, D., Gärtner, B., 1996. A novel type of skeleton for polygons. J. Univers. Comput. Sci. 1, 752–761.

Alber, R., Rudolph, S., 2004. On a grammar-based design language that supports automated design generation and creativity, in: Borg, J., Farrugia, P., Camilleri, K. (Eds.), Knowledge Intensive Design Technology. Springer, pp. 19–35.

Andreasen, C.S., Gersborg, A.R., Sigmund, O., 2009. Topology optimization of microfluidic mixers. Int. J. Numer. Methods Fluids 61, 498–513.

Antonsson, E.K., Cagan, J., 2001. Formal engineering design synthesis.

Armenàkas, A., 1988. Classical structural analysis: a modern approach. McGraw-Hill Press, New York, USA.

Asadpoure, A., Tootkaboni, M., Guest, J.K., 2011. Robust topology optimization of structures with uncertainties in stiffness – Application to truss structures. Comput. Struct. 89, 1131–1141.

Barequet, G., Goodrich, M.T., Levi-Steiner, A., Steiner, D., 2004. Contour interpolation by straight skeletons. Graph. Models 66, 245–260.

Barnes, M.R., 1999. Form Finding and Analysis of Tension Structures by Dynamic Relaxation. Int. J. Sp. Struct. 14, 89–104.

Beierle, C., Kern-Isberner, G., 2008. Methoden wissensbasierter Systeme, Springer. Vieweg+Teubner Verlag, Wiesbaden, Germany.

Bendsøe, M.P., Ben-Tal, A., Zowe, J., 1994. Optimization methods for truss geometry and topology design. Struct. Optim. 7, 141–159.

Bendsøe, M.P., Sigmund, O., 2003. Topology Optimization, Vasa. Springer Berlin Heidelberg, Berlin, Heidelberg.

Bolognini, F., Seshia, A.A., Shea, K., 2007. Exploring The Application Of A Multidomain Simulation-Based Computational Synthesis Method In Mems Design. Int. Conf. Eng. Des. ICED’07.

Borrvall, T., Petersson, J., 2003. Topology optimization of fluids in Stokes flow. Int. J. Numer. Methods Fluids 41, 77–107.

Boston, O., 1998. Technical liaisons in engineering design: understanding by modelling. University of Bath, UK.

Bulman, S., Sienz, J., Hinton, E., 2001. Comparisons between algorithms for structural topology optimization using a series of benchmark studies. Comput. Struct. 79, 1203–1218.

Cagan, J., 2001. Engineering Shape Grammars: Where We Have Been and Where We are Going, in: Antonsson, E.K., Cagan, J. (Eds.), Formal Engineering Design Synthesis. Cambridge University Press, New York, pp. 65–92.

Cagan, J., Campbell, M.I., Finger, S., Tomiyama, T., 2005. A Framework for Computational Design Synthesis: Model and Applications. J. Comput. Inf. Sci. Eng. Vol. 5, 11.

Calafiore, G.C., Dabbene, F., 2008. Optimization under uncertainty with applications to design of truss structures. Struct. Multidiscip. Optim. 35, 189–200.

Campbell, M.I., 2013. GraphSynth [WWW Document]. URL www.graphsynth.com (accessed 12.1.13).

Celani, M., 2002. Beyond analysis and representation in CAD: a new computational approach to design education. Massachusetts Institute of Technology.

CGAL, 2013. CGAL, Computational Geometry Algorithms Library [WWW Document]. URL www.cgal.org (accessed 12.1.13).

Chakrabarti, A., Shea, K., Stone, R., Cagan, J., Campbell, M., Hernandez, N.V., Wood, K.L., 2011. Computer-Based Design Synthesis Research: An Overview. J. Comput. Inf. Sci. Eng. 11, 021003.

Challis, V.J., Guest, J.K., 2009. Level set topology optimization of fluids in Stokes flow 1284–1308.

Chan, W.L., Arbelaez, D., Bossens, F., Skelton, R.E., 2004. Active vibration control of a three-stage tensegrity structure, in: Wang, K.-W. (Ed.), SPIE 5386, Smart Structures and Materials 2004. pp. 340–346.

Chase, S.C., 2002. A model for user interaction in grammar-based design systems. Autom. Constr. 11, 161–172.

Cohn, M.Z., Dinovitzer, a. S., 1994. Application of Structural Optimization. J. Struct. Eng. 120, 617–650.

Court, A.W., 1995. Modelling and classification of information for engineering design. Proc. 1995 Des. Eng. …. University of Bath, UK.

Deshpande, A., Riehle, D., 2008. The total growth of open source, in: Russo, B., Damiani, E., Hissam, S., Lundell, B., Succi, G. (Eds.), Open Source Development, Communities and Quality - IFIP, Volume 275. Springer US, pp. 197–209.

Díaz, A.R., Bendsøe, M.P., 1992. Shape optimization of structures for multiple loading conditions using a homogenization method. Struct. Optim. 4, 17–22.

Dorn, W.S., Gomory, R.E., Greenberg, H.J., 1964. Automatic design of optimal structures. J. Mec. 3, 25–52.

Drumheller, M., 2002. Constraint-Based Design of Optimal Transport Elements. J. Comput. Inf. Sci. Eng. 2, 302.

Duan, X.-B., Ma, Y.-C., Zhang, R., 2008. Shape-topology optimization for Navier–Stokes problem using variational level set method. J. Comput. Appl. Math. 222, 487–499.

Dym, C.L., Levitt, R.E., 1991. Knowledge-Based Systems in Engineering, Lecture Notes in Computer Science, vol. 5178. McGraw-Hill Press, New York, NY, USA.

Eftekharian, A. a., Ilieş, H.T., 2012. Medial zones: Formulation and applications. Comput. Des. 44, 413–423.

Ehrlenspiel, K., 1997. Knowledge explosion and its consequences, in: Riitahuhta, A. (Ed.), International Conference on Engineering Design, ICED97. Tampere, Finland, pp. 477–484.

Eschenauer, H.A., Olhoff, N., 2001. Topology optimization of continuum structures: A review. Appl. Mech. Rev. 54, 331.

Estrada, G.G., Bungartz, H., Mohrdieck, C., 2006. Numerical form-finding of tensegrity structures. Int. J. Solids Struct. 43, 6855–6868.

Evgrafov, A., 2006. Topology optimization of slightly compressible fluids. ZAMM 86, 46–62.

Fay, J.A., 1994. Introduction to fluid mechanics. MIT Press, Cambridge, MA, USA.

Gersborg-Hansen, A., Sigmund, O., Haber, R., 2005. Topology optimization of channel flow problems. Struct. Multidiscip. Optim. 30, 181–192.

Guest, J.K., Igusa, T., 2008. Structural optimization under uncertain loads and nodal locations. Comput. Methods Appl. Mech. Eng. 198, 116–124.

Guest, J.K., Prévost, J.H., 2006a. Topology optimization of creeping fluid flows using a Darcy–Stokes finite element. Int. J. Numer. Methods Eng. 66, 461–484.

Guest, J.K., Prévost, J.H., 2006b. Optimizing multifunctional materials: Design of microstructures for maximized stiffness and fluid permeability. Int. J. Solids Struct. 43, 7028–7047.

Guest, J.K., Prévost, J.H., 2007. Design of maximum permeability material structures. Comput. Methods Appl. Mech. Eng. 196, 1006–1017.

Guzman, M. De, Orden, D., 2004. From graphs to tensegrity structures: geometric and symbolic approaches. arXiv Prepr. math/0404334.

Harasaki, H., Arora, J.S., 2001. New concepts of transferred and potential transferred forces in structures. Comput. Methods Appl. Mech. Eng. 191, 385–406.

Harasaki, H., Arora, J.S., 2002. Topology design based on transferred and potential transferred forces. Struct. Multidiscip. Optim. 23, 372–381.

Hart-Smith, L.J., 1995. An engineer's viewpoint on design and analysis of aircraft structural joints. Arch. Proc. Inst. Mech. Eng. Part G J. Aerosp. Eng. 1989-1996 (vols 203-210) 209, 105–129.

Heckel, R., Küster, J., Taentzer, G., 2002. Confluence of typed attributed graph transformation systems. Graph Transform. 2505, 161–176.

Helms, B., 2013. Object-Oriented Graph Grammars for Computational Design Synthesis. Technische Universität München.

Helms, B., Shea, K., Hoisl, F., 2009. A Framework for Computational Design Synthesis Based on Graph-Grammars and Function-Behavior-Structure, in: ASME 2009 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference. San Diego, USA.

Hertel, G., Niedner, S., Herrmann, S., 2003. Motivation of software developers in Open Source projects: an Internet-based survey of contributors to the Linux kernel. Res. Policy 32, 1159–1177.

Hicks, B., Culley, S., Allen, R., Mullineux, G., 2002. A framework for the requirements of capturing, storing and reusing information and knowledge in engineering design. Int. J. Inf. Manage. 22, 263–280.

Hoisl, F., Shea, K., 2011. An Interactive, Visual Approach to Developing and Applying Parametric Three-Dimensional Spatial Grammars. Artif. Intell. Eng. Des. Anal. Manuf. 25, 333–356.

Hooshmand, A., Campbell, M.I., 2013. Topology Optimization of Fluid Channels Using Generative Graph Grammars, in: 39th Design Automation Conference,. ASME.

Hooshmand, A., Campbell, M.I., 2014. Layout Synthesis of Fluid Channels Using Generative Graph Grammars. AI EDAM 28.

Hooshmand, A., Campbell, M.I., Shea, K., 2012. Steps in Transforming Shapes Generated With Generative Design Into Simulation Models, in: Volume 3: 38th Design Automation Conference, Parts A and B. ASME, pp. 883–892.

Hooshmand, A., Schlaich, M., Belaus, L., Campbell, M.I., 2013. CDS Platform - a Platform for Multi-Physics Computational Design Synthesis, in: Lindemann, U., Venkataraman, S., Kim, Y., Lee, S., Papalambros, P., Chen, W. (Eds.), 19th International Conference on Engineering Design (ICED13), Design for Harmonies, Vol.9: Design Methods and Tools. Design Society, Seoul, Korea, pp. 099–108.

Hoshino, H., Sakuraib, T., Takahashic, K., 2003. Vibration reduction in the cabins of heavy-duty trucks using the theory of load transfer paths. JSAE Rev. 24, 165–171.

Hubka, V., Andreasen, M., Eder, W., Hills, P., 1988. Practical studies in systematic design. Butterworths, London and Boston.

Hwang, F., Richards, D., Winter, P., 1992. The Steiner tree problem. North-Holland.

Jalalpour, M., Igusa, T., Guest, J.K., 2011. Optimal design of trusses with geometric imperfections: Accounting for global instability. Int. J. Solids Struct. 48, 3011–3019.

Jang, G., Panganiban, H., Chung, T.J., 2010. P1-nonconforming quadrilateral finite element for topology optimization 685–707.

Juan, S.H., Mirats Tur, J.M., 2008. Tensegrity frameworks: Static analysis review. Mech. Mach. Theory 43, 859–881.

Karac, A., 2003. Drop impact of fluid-filled polyethylene containers. Imperial College London.

Kelly, D., Elsley, M., 1995. A procedure for determining load paths in elastic continua. Eng. Comput. 12, 415–424.

Kermode, A.C., 1964. Aeroplane Structure, 2nd Editio. ed. Pitman Publishing.

Khanjani, A., Sulaiman, R., 2011. The aspects of choosing open source versus closed source, in: 2011 IEEE Symposium on Computers & Informatics. IEEE, pp. 646–649.

Kirsch, U., 1989. Optimal topologies of truss structures. Comput. Methods Appl. Mech. Eng. 72, 15–28.

Kirsch, U., 1990. On the relationship between optimum structural topologies and geometries. Struct. Optim. 2, 39–45.

Koohestani, K., 2013. A computational framework for the form-finding and design of tensegrity structures. Mech. Res. Commun. 54, 41–49.

Kurtoglu, T., Swantner, A., Campbell, M.I., 2010. Automating the conceptual design process: “From black box to component selection”. Artif. Intell. Eng. Des. Anal. Manuf. 24, 49.

Li, Y., Feng, X.-Q., Cao, Y.-P., Gao, H., 2010. A Monte Carlo form-finding method for large scale regular and irregular tensegrity structures. Int. J. Solids Struct. 47, 1888–1898.

Liang, Q., 2001. Performance-based optimization method for structural topology and shape design. Victoria University of Technology, Australia.

Liang, Q., 2005. Performance-based Optimization of Structures: Theory and applications. Taylor & Francis Ltd, London and New York.

Lindsay, R.K., Buchanan, B.G., Feigenbaum, E.A., Lederberg, J., 1993. DENDRAL: A case study of the first expert system for scientific hypothesis formation. Artif. Intell. 61, 209–261.

Liu, H., Li, P., 2013. Maintaining equal operating conditions for all cells in a fuel cell stack using an external flow distributor. Int. J. Hydrogen Energy 38, 3757–3766.

Liu, Z., Gao, Q., Zhang, P., Xuan, M., Wu, Y., 2010. Topology optimization of fluid channels with flow rate equality constraints. Struct. Multidiscip. Optim. 44, 31–37.

Lógó, J., 2007. New Type of Optimality Criteria Method in Case of Probabilistic Loading Conditions #. Mech. Based Des. Struct. Mach. 35, 147–162.

Lógó, J., Ghaemi, M., Rad, M.M., 2009. Optimal Topologies in Case of Probabilistic Loading: The Influence of Load Correlation. Mech. Based Des. Struct. Mach. 37, 327–348.

Lou, R., Pernot, J.-P., Mikchevitch, A., Véron, P., 2010. Merging enriched Finite Element triangle meshes for fast prototyping of alternate solutions in the context of industrial maintenance. Comput. Des. 42, 670–681.

Luger, G.F., 2008. Artificial intelligence: Structures and strategies for complex problem solving, 6th ed. Addison-Wesley, Harlow, England.

Luh, G.-C., Lin, C.-Y., 2011. Optimal design of truss-structures using particle swarm optimization. Comput. Struct. 89, 2221–2232.

Luo, Z., Tong, L., Kang, Z., 2009. A level set method for structural shape and topology optimization using radial basis functions. Comput. Struct. 87, 425–434.

Marhadi, K., Venkataraman, S., 2009. Comparison of Quantitative and Qualitative Information Provided by Different Structural Load Path Definitions. Int. J. Simul. Multidiscip. Des. Optim. 3, 384–400.

Marsh, R., 1997. The capture and utilisation of experience in engineering design. University of Cambridge.

Masic, M., Skelton, R.E., 2004. Open-loop control of class-2 tensegrity towers, in: Smith, R.C. (Ed.), Smart …. pp. 298–308.

Masic, M., Skelton, R.E., Gill, P.E., 2005. Algebraic tensegrity form-finding. Int. J. Solids Struct. 42, 4833–4858.

Micheletti, A., Williams, W., 2007. A marching procedure for form-finding for tensegrity structures. J. Mech. Mater. Struct. 2, 857–882.

Mirats Tur, J.M., Juan, S.H., 2009. Tensegrity frameworks: Dynamic analysis review and open problems. Mech. Mach. Theory 44, 1–18.

Motro, R., 2003. Tensegrity: structural systems for the future. Elsevier Science Publishers B.V.

Motro, R., Najari, S., Jouanna, P., 1987. Static and dynamic analysis of tensegrity systems. Shell Spat. Struct. Comput. Asp. 26, 270–279.

Netten, B., Vingerhoeds, R., 1997. EADOCS: Conceptual design in three phases—An application to fibre reinforced composite panels. Eng. Appl. Artif. Intell. 10, 129–138.

Nishimura, Y., Murakami, H., 2001. Initial shape-finding and modal analyses of cyclic frustum tensegrity modules. Comput. Methods Appl. Mech. Eng. 190, 5795–5818.

Nocedal, J., Wright, S., 2006. Numerical Optimization, 2nd ed. Springer-Verlag, New York Berlin Heidelberg.

Noilublao, N., Bureerat, S., 2011. Simultaneous topology, shape and sizing optimisation of a three-dimensional slender truss tower using multiobjective evolutionary algorithms. Comput. Struct. 89, 2531–2538.

Ognjanovic, M., 1999. Creativity in design incited by knowledge modelling, in: International Conference on Engineering Design. Munich, Germany, pp. 1925–1928.

Olesen, L.H., Okkels, F., Bruus, H., 2006. A high-level programming-language implementation of topology optimization applied to steady-state Navier-Stokes flow. Int. J. Numer. Methods Eng. 65, 975–1001.

Open CASCADE, 2013. Salome Platform [WWW Document]. URL www.salome-platform.org (accessed 12.1.13).

OpenCFD Ltd (ESI Group), 2013. OpenFoam [WWW Document]. URL www.openfoam.com (accessed 12.1.13).

Osgood, C.C., 1970. Fatigue Design. Wiley-Interscience.

Paul, C., Valero-Cuevas, F.J., Lipson, H., 2006. Design and control of tensegrity robots for locomotion. IEEE Trans. Robot. 22, 944–957.

R&D, E., 2013. Code-Aster [WWW Document]. URL www.code-aster.org (accessed 12.1.13).

Rai, R., Kilaru, P., Vallepalli, R., Campbell, M.I., 2011. A Novel Search Algorithm for Interactive Automated Conceptual Design Generator (ACDG), in: Volume 5: 37th Design Automation Conference, Parts A and B. ASME, pp. 987–996.

Reddy, G., Cagan, J., 1995. An Improved Shape Annealing Algorithm For Truss Topology Generation. J. Mech. Des. 117, 315.

Ribes, A., Caremoli, C., 2007. Salome platform component model for numerical simulation, in: 31st Annual International Computer Software and Applications Conference - Vol. 2 - (COMPSAC 2007). IEEE, pp. 553–564.

Rieffel, J., Valero-Cuevas, F., Lipson, H., 2009. Automated discovery and optimization of large irregular tensegrity structures. Comput. Struct. 87, 368–379.

Riegel, J., Mayer, W., van Havre, Y., 2013. FreeCAD [WWW Document]. URL www.freecadweb.org (accessed 12.1.13).

Rong, J.H., Liang, Q.Q., 2008. A level set method for topology optimization of continuum structures with bounded design domains. Comput. Methods Appl. Mech. Eng. 197, 1447–1465.

Rude, S., 1998. Wissensbasiertes Konstruieren. Shaker Verlag, Aachen, Germany.

Ruffin, M., Ebert, C., 2004. Using open source software in product development: a primer. IEEE Softw. 21, 82–86.

Sandgren, E., Cameron, T.M., 2002. Robust design optimization of structures through consideration of variation. Comput. Struct. 80, 1605–1613.

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

Schotborgh, W.O., Tragter, H., Kokkeler, F.G.M., van Houten, F.J.A.M., 2006. A Bottom-Up Approach For Automated Synthesis Support In The Engineering Design Process Prototypes, in: Marjanovic, D. (Ed.), DESIGN 2006, the 9th International Design Conference. Dubrovnik - Croatia, pp. 349–356.

SciPy Developers, 2013. Scipy [WWW Document]. URL www.scipy.org (accessed 12.1.13).

Shea, K., 1997. Essays of discrete structures: purposeful design of grammatical structures by directed stochastic search. Carnegie Mellon University.

Shea, K., Aish, R., Gourtovaia, M., 2005. Towards integrated performance-driven generative design tools. Autom. Constr. 14, 253–264.

Shea, K., Cagan, J., 1998. Topology Design of Truss Structures by Shape Annealing. Proc. DETC98 1998 ASME Des. Eng. Tech. Conf.

Shea, K., Cagan, J., 1999. The Design of Novel Roof Trusses with Shape Annealing: Assessing the Ability of a Computational Method in Aiding Structural Designers with Varying Design Intent. Des. Stud. 20, 3–23.

Shea, K., Cagan, J., Fenves, S.J., 1997. A Shape Annealing Approach to Optimal Truss Design with Dynamic Grouping of Members. ASME J. Mech. Des. 119, 388–394.

Shea, K., Fest, E., Smith, I.F.C., 2002. Developing intelligent tensegrity structures with stochastic search. Adv. Eng. Informatics 16, 21–40.

Shortliffe, E.H., Davis, R., Axline, S.G., Buchanan, B.G., Green, C.C., Cohen, S.N., 1975. Computer-based consultations in clinical therapeutics: Explanation and rule acquisition capabilities of the MYCIN system. Comput. Biomed. Res. 8, 303–320.

Siemens PLM Software Inc., 2013. Parasolid [WWW Document]. URL http://www.plm.automation.siemens.com/en_us/products/open/parasolid/index.shtml (accessed 12.1.13).

Silva, L.F.L.R., Lage, P.L.C., 2011. Development and implementation of a polydispersed multiphase flow model in OpenFOAM. Comput. Chem. Eng. 35, 2653–2666.

Skakoon, J.G., 2008. The Elements of Mechanical Design. ASME, Three Park Avenue New York, NY 10016-5990.

Starling, A.C., Shea, K., 2005. A Parallel Grammar for Simulation-Driven Mechanical Design Synthesis. Proc. Des. Autom. Conf. DETC05 ASME Des. Eng. Tech. Conf.

Tibert, A., Pellegrino, S., 2003. Review of Form-Finding Methods for Tensegrity Structures. Int. J. Sp. Struct. 18, 209–223.

Tran, H.C., Lee, J., 2010. Initial self-stress design of tensegrity grid structures. Comput. Struct. 88, 558–566.

Tyas, a., Gilbert, M., Pritchard, T., 2006. Practical plastic layout optimization of trusses incorporating stability considerations. Comput. Struct. 84, 115–126.

Vangelooven, J., De Malsche, W., Op De Beeck, J., Eghbali, H., Gardeniers, H., Desmet, G., 2010. Design and evaluation of flow distributors for microfabricated pillar array columns. Lab Chip 10, 349–56.

Vassart, N., Motro, R., 1999. Multiparametered formfinding method: application to tensegrity systems. Int. J. Sp. Struct. 14, 147–154.

Visser, W., 2006. Designing as Construction of Representations: A Dynamic Viewpoint in Cognitive Design Research. Human-Computer Interact. 21, 103–152.

Wang, B.-B., 1998. Cable-strut systems: part I — tensegrity. J. Constr. Steel Res. 45, 281–289.

Wang, M.Y., Wang, X., 2004. “Color” level sets: a multi-phase method for structural topology optimization with multiple materials. Comput. Methods Appl. Mech. Eng. 193, 469–496.

Wang, S.Y., Lim, K.M., Khoo, B.C., Wang, M.Y., 2007. An extended level set method for shape and topology optimization. J. Comput. Phys. 221, 395–421.

Wang, X.-J., Zhang, C.-Y., Gao, L., Li, P.-G., 2008. A Survey and Future Trend of Study on Multi-Objective Scheduling, in: 2008 Fourth International Conference on Natural Computation. IEEE, pp. 382–391.

Whitney, D.E., 1996. Why mechanical design cannot be like VLSI design. Res. Eng. Des. 8, 125–138.

Yamaguchi, H., 2008. Engineering Fluid Mechanics. Springer Netherlands, Dordrecht, The Netherlands.

Yonekura, K., Kanno, Y., 2010. Global optimization of robust truss topology via mixed integer semidefinite programming. Optim. Eng. 11, 355–379.

Yulin, M., Xiaoming, W., 2004. A level set method for structural topology optimization and its applications. Adv. Eng. Softw. 35, 415–441.

Zhang, J., Ohsaki, M., 2006. Adaptive force density method for form-finding problem of tensegrity structures. Int. J. Solids Struct. 43, 5658–5673.

Zhang, L., Maurin, B., Motro, R., 2006. Form-Finding of Nonregular Tensegrity Systems. J. Struct. Eng. 132, 1435–1440.

Zhang, L.-Y., Li, Y., Cao, Y.-P., Feng, X.-Q., 2014. Stiffness matrix based form-finding method of tensegrity structures. Eng. Struct. 58, 36–48.

Zhou, S., Li, Q., 2008. A variational level set method for the topology optimization of steady-state Navier–Stokes flow. J. Comput. Phys. 227, 10178–10195.