Automation in Construction 39 (2014) 59–69

Contents lists available at ScienceDirect

Automation in Construction

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

Automatic innovative truss design using grammatical evolution

Michael Fenton [a,*], Ciaran McNally [a], Jonathan Byrne [a], Erik Hemberg [b], James McDermott [a], Michael O'Neill [a]

[a] UCD, Ireland
[b] MIT, United States

ARTICLE INFO

Article history:
Accepted 29 November 2013
Available online 4 January 2014

Keywords:
Structural optimization
Genetic programming
Evolutionary computation
Grammatical evolution
Truss design
Computer aided design

ABSTRACT

Truss optimization in the field of Structural Engineering is a growing discipline. The application of Grammatical Evolution, a grammar-based form of Genetic Programming (GP), has shown that it is capable of generating innovative engineering designs. Existing truss optimization methods in GP focus primarily on optimizing global topology. The standard method is to explore the search space while seeking minimum cross-sectional areas for all elements. In doing so, critical knowledge of section geometry and orientation is omitted, leading to inaccurate stress calculations and structures not meeting codes of practice. This can be addressed by constraining the optimisation method to only use standard construction elements.

The aim of this paper is not to find fully optimized solutions, but rather to show that solutions very close to the theoretical optimum can be achieved using real-world elements. This methodology can be applied to any structural engineering design which can be generated by a grammar.

1. Introduction

A major part of engineering design is the process of satisfying hard constraints. In structural engineering, topology optimization is known as the science of "optimal layout theory" [1]. It allows engineers to design highly optimized structures—maximizing material efficiency while minimizing waste and reducing material cost. This allows for structures that are stiff yet lightweight, which can lead to savings in terms of resources and cost [2–4]. Engineering optimization is an important problem as minor savings in weight or cost on a small scale can have larger implications when extrapolated over a larger design or project.

The theory of topology optimization in structural engineering states that it is possible to create a structurally "perfect" design with both optimal shape topology and member sizes by both rearranging the topological layout of the members and by varying the sizes of those individual members [4]. All members in the design should have similar high states of stress at, or close to (but not exceeding) the limits of the material as specified by design codes of practice and manufacturer specifications. This eliminates redundancy, minimizes material usage and creates a more economical design. This is most usually achieved by minimizing the cross-sectional area of each structural member, which consequently minimizes the overall weight of the entire structure.

Genetic Programming (GP) has been shown to be routinely capable of achieving human-competitive performance in a number of real-world scenarios [5–7]. Evidence of an increase in use of GP in industry can be found in the increasing number of patent applications using GP [8]. GP is particularly well suited for engineering tasks for a number of reasons, including its ability to handle multiple conflicting objectives [14,15] and its capacity to optimize both the structure and the contents of that structure in parallel [14]. Since the solution is unknown (due to incomplete information or theory), GP is one method in particular which can uncover its optimal structure/topological form [1,11,16,17]. Sizing optimization is similar in theory to solving a simple linear equation: the form (topology) is known, and the variables (member sizes) are increased/decreased to fit. It is therefore possible to use both GP and linear optimization as a hybrid approach towards topology optimization.

Grammatical Evolution (GE) is a version of GP that uses a formal grammar [9,11,12], allowing the user to easily embed domain knowledge (such as structure boundary conditions, loading conditions, and basic form including span and depth), and to generate output in any language [11]. Both GE and topology optimization represent the cutting edge of both GP and structural engineering fields respectively.

This paper introduces a new method of topology optimization: Dual Optimization in Grammatical Evolution (DO-GE). While existing structural optimization methods in GE [14,16,18] primarily focus on a structural topology scale (optimization of the structural layout), optimization of individual element sizes is also possible [1,19,20]. The combination of both topology and sizing optimization is established [1,16,17,19,29], but the use of both standard construction elements and compliance to design codes of practice in the process is novel. Standard practice is to optimize element sizings by specifying the required cross-sectional area. While this gives theoretically optimized results, the output is of little use to structural engineers as in practice

* Corresponding author.
E-mail address: michaelfenton1@gmail.com (M. Fenton).

0926-5805/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.autcon.2013.11.009

trusses are constructed using structural elements with preset cross section and geometry. This highlights a fundamental weakness in traditional sizing optimization methods: by omitting knowledge of section geometry and orientation, it is not possible to include accurate buckling calculations as a constraint for structural design and thus structures cannot be designed according to standard codes of practice. The approach presented in this paper addresses this deficiency by allowing for any number of standard construction elements to be specified for any elements within a design, leading to code-compliant construction-ready designs which truly represent their evolved form.

The DO-GE approach has a number of advantages over a two-stage approach of optimizing topology and element sizes separately. With a single stage approach, a large number of designs can be assessed in a relatively short space of time, whereas a two-stage approach would be slower and would fail to allow for interactions between structural topology and element sizes parameters. A single-stage approach also allows for real-time analysis of both design variables and structural properties of the individuals as evolution progresses.

Section 2 will begin with a summary of related research in this area, along with a description of the DO-GE method, including our approach to design generation and analysis. Section 3 compares and contrasts recent research methods with the DO-GE method using examples from the literature, and a discussion on the implications of those results is presented in Section 4. Finally, our conclusions and suggestions for future work are presented in Section 5.

2. Evolutionary approaches to structural engineering

The use of computers in structural design has been growing rapidly in recent years. The advent of techniques such as Topology Optimization [1] and Evolutionary Computation (EC) [17] has heralded engineering applications ranging from analog circuit design [5] to the design of structures such as shelters [30] and bridges [14].

2.1. Engineering design approaches

A recent survey of the applications of evolutionary computation in structural engineering design [17] has found the most difficult aspects of the design to be i) appropriate representation of the engineering system itself and ii) finding a suitable evaluation function. Appropriate representation of the engineering system is possible using the relevant design codes of practice [2,3,30,32]. In the case of structural design this entails creating boundary conditions (supports and loading), material limits (usually expressed as stress or strain) and design limits (deflection). The use of the Finite Element method of structural analysis [4] as a fitness function has been proven useful [14,18], and it enables the EC program to assess and evaluate individuals based on the results of a finite element analysis.

Both Murawski et al. [41] and Kicinger et al. [44] successfully used Evolutionary Computation (EC) methods to evolve steel wind bracings for tall structures, based on Grierson and Cameron's SODA method

```
<S> ::= <program>{}<call>

<program> ::= def program(chromosome_b):{<init>{}<constants>{}<define_funcs>{}<make_all>{}<return>{}}

<init> ::= truss_graph = graph.graph(){}

<constants> ::= span = <span>{}depth = <depth>{}r = <r>{}genome = chromosome_b

<define_funcs> ::= <chord>{}<cross_brace>{}

<chord> ::= def chord(r):{top_chord = []{}bottom_chord = []{}for i in range (r+1):{bay_span =
i*span/(2*r){}tnode, bnode = [0,bay_span,depth],
[0,bay_span,0]{}top_chord.append(tnode){}bottom_chord.append(bnode)}return top_chord, bottom_chord}

<cross_brace> ::= def cross_brace(r):{edge_list = []{}top_ids = []{}bottom_ids = []{}chords =
chord(r){}top_chord = chords[0]{}bottom_chord = chords[1]{}<truss_type>for i, edge in
enumerate(edge_list):{truss_graph.add_edge(edge[0], edge[1], material=genome[i])}return edge_list}

<connection_type> ::= <howe>|<pratt>|<howe>|<fully_braced>|<warren>|<modified_warren>|<vierendeel>

<pratt> ::= connect_pratt_truss{}

<howe> ::= connect_howe_truss{}

<fully_braced> ::= connect_fully_braced_truss{}

<warren> ::= connect_warren_truss{}

<modified_warren> ::= connect_modified_warren_truss{}

<vierendeel> ::= connect_vierendeel_truss{}

<r> ::= 2|3|4|...|19|20

<span> ::= 24000

<depth> ::= <span>/10|<span>/11|<span>/12...|<span>/24|<span>/25

<make_all> ::= edge_list = cross_brace(r){}mirror_graph = truss_graph. mirror(truss_graph){}

<return> ::= return mirror_graph{}

<call> ::= program(chromosome_b)
```

Fig. 1. A sample truss grammar.