Computers and Structures 173 (2016) 61–70

Contents lists available at ScienceDirect

Computers and Structures

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

Multiobjective structural optimization of frameworks using enumerative topology

Kirk Martini *

Department of Architecture, University of Virginia, PO Box 400122, Charlottesville, VA 22904, USA

ARTICLE INFO

Article history:
Received 8 December 2015
Accepted 20 May 2016
Available online 14 June 2016

Keywords:
Structural design
Evolution strategies
Multiobjective optimization
Topological optimization

ABSTRACT

The paper describes a multiobjective optimization method aimed at supporting decisions in conceptual design. The paper identifies five characteristics to achieve this intent. One of those characteristics concerns accounting for variation configuration. The method introduces *enumerative topology*, a way of specifying structural configuration in terms of enumerative variables, such as the number of bays in a truss or frame. This concept is distinct from conventional topological optimization which defines configuration by specifying the presence or absence of members in a framework. The paper demonstrates the method on two benchmark problems, and then presents extended versions which incorporate enumerative topology.

1. Introduction

Optimization of structural frameworks is a mature field, with dozens of papers presenting methods to solve a prototypical problem: given a model where selected properties are defined by a vector of decision variables, find a combination of variable values that minimizes the weight of structural material, while satisfying constraints on structural strength, stiffness, and stability. While some researchers have noted the potential of population-based structural optimization methods to support creative and conceptual design [1–6], most published work has focused on the final stages of design, where the structural configuration is well established, and the goal is to find the feasible minimum-weight design. Typically, the configuration of the structure is largely, or even completely, set, and the optimization focusses on the selection of member sizes, e.g. [7–10].

This paper describes and demonstrates a method oriented at the earlier, more conceptual stages of design. The method is called Vespo, for Variable Evolution Strategy for Pareto optimization. To address some of the needs of conceptual design, the development of Vespo defined the following characteristic goals:

1. **Topology:** Account for variations in structural topology.
2. **Diversity:** Generate a diverse population of alternatives.
3. **Trade-offs:** Account for trade-offs among multiple objectives.
4. **Ability to approximate:** Produce useful though suboptimal answers with relatively few model evaluations.
5. **Adaptability:** Minimize the number of user-specified control parameters.

The following discussion explains the motivations for these goals, and the relation of these goals to preceding research in optimization, and to Vespo.

The topology goal recognizes that determining topology and configuration is a central task of conceptual design. A novel feature of Vespo with respect to this goal is *enumerative topology*. The following discussion describes this feature, beginning with a general discussion of types of decision variable commonly used in structural optimization.

Structural optimization of frameworks typically uses three types of decision variable: *shape variables*, which determine spatial dimensions of the structure; *size variables*, which determine the cross sections for members in a framework; and *topology variables*, which determine whether selective members in a framework are active. As discussed above, there is a great deal of research devoted to optimization solely with size variables. There is also research aimed more at conceptual stages, which considers both size and shape optimization [11,12], and some that considers size, shape, and topology [1–3,13–16].

When topology is addressed in structural optimization, it is usually a form that will be called *selective topology*, since the variables involved select which members are active [4,13]. A few researchers have addressed other forms of topological optimization. Rajeev [15] presented a method where decision variables identified one of several options for configuring members in the panel of a tower truss. In this method, it was possible for solution vectors in the population to have differing numbers of decision

* Tel.: +1 434 924 6445; fax: +1 434 982 2678.
E-mail address: martini@virginia.edu

http://dx.doi.org/10.1016/j.compstruc.2016.05.020
0045-7949/

variables, since the corresponding structures would have differing numbers of members and cross sections; the method used a version of a genetic algorithm (GA) modified to work with solutions of differing length. Raich [5,6] also applied a GA that could accommodate varying topologies and solution vectors of different lengths. von Buelow [1] also used GAs for topological optimization without using a ground structure. The method separated the generation of topology from the optimization of size and shape. Solutions of differing length were partitioned into distinct topology groups. Compared with Rajeev's method [15], the methods of Raich [5] and von Buelow [1] were less restrictive and could generate a wider range of topologies. Shea [2] applied simulated annealing and shape grammars to similarly generate a wide range of topologies. Simulated annealing generates a new solution using only one existing solution, rather than combining multiple solutions, so accommodating solution vectors with differing lengths was straightforward in Shea's approach.

Vespo takes an approach to topology most similar to that of Rajeev [15], and calls this approach *enumerative topology*, because it includes decision variables that enumerate some aspect of structural configuration, such as the number of panels in a truss, or bays in a frame. While the method presented by Rajeev [15] was potentially powerful, there has been little subsequent development of the concept, perhaps because it required complex custom logic to convert from the binary string representation of the solution vector to the corresponding structural configuration. Vespo addresses this difficulty first by using real-valued encoding of solution vectors, and more importantly taking advantage of currently available software for parametric representation of geometry, such as Grasshopper in the Rhino environment [17], or Dynamo in the AutoCAD environment [18]. In this context, "parametric representation" is one where the user can define a complex geometric model that incorporates a sequence of calculations. Changing a parameter can trigger a cascading recalculation to revise the model, similar to the recalculation of a spreadsheet. Vespo is implemented in the Rhino–Grasshopper environment. Rhino is a general-purpose CAD program, used widely in industrial design, and for conceptual design in architecture. Grasshopper is a plug-in that supports parametric representation within Rhino.

Enumerative topology raises the issue of dealing with solution vectors of differing length. As discussed above, Rajeev [15] and Raich [5] addressed this problem with a GA modified with logic to generate a new "offspring" solution from two existing "parent" solutions of differing length. Shea [2] addressed this problem by using simulated annealing, which generates a new offspring solution from only one existing parent solution. In this respect, Vespo's approach is most similar to Shea's, in that it uses an Evolution Strategy (ES), which like simulated annealing, generates a new offspring solution using only one parent solution, avoiding the logistic problem of combining solutions of differing length. Details are discussed in Section 2.

The diversity goal concerns generating a wide range of alternatives, rather than a single best solution. This goal reflects the exploratory nature of conceptual design. As mentioned above, several researchers have recognized the potential of population-based metaheuristic optimization methods in generating alternatives. To achieve this goal, such methods require a mechanism to preserve diversity in the population of solutions, since without it, there is a strong tendency for all solutions in the population to converge [13,14]. Vespo maintains diversity by accounting for multiple objectives, which also addresses the trade-off goal. Vespo is a multi-objective optimization method that seeks to produce a non-dominated Pareto set of solutions [19]. The method uses the non-dominated sort algorithm of NGSA-II, developed by Deb [19]. Diversity, is maintained by comparing solutions in the space defined by objective function values (i.e. *objective space*), rather than by comparing solutions in the space defined by decision variable values (i.e. *decision space*). Comparing in decision space is not straightforward when decision vectors may have differing lengths; the comparison in objective space is much simpler, since all solutions have the same objective functions, irrespective of the number of decision variables.

The ability-to-approximate goal concerns the trade-offs among speed, precision, and consistency inherent in any optimization problem. Speed concerns the computational resources needed to obtain a solution set. Precision concerns how close that solution set is to a "true" optimum or optimal set, which may be known for published benchmarks, but in general is not known. Consistency concerns the range of variation in results from multiple optimization runs. In the literature, methods which produce a solution with either greater speed or greater precision are considered to perform better, and consistency is not always addressed. This common approach for comparing methods does not fully recognize that the priorities of speed and precision may vary at different stages of the design process. In conceptual design, where there is interest in quickly comparing diverse alternatives, speed has relatively high priority: a solution with 10% precision delivered quickly may be more useful than a solution with 1% precision delivered slowly. In contrast, in the final stages of design, a solution with 10% precision may be unacceptable, no matter how quickly produced.

To address these different priorities of speed and precision, the Vespo method is organized to obtain the most precise solution set it can, given the number of model evaluations specified by the designer. This is achieved through an adaptive control mechanism described in Section 2.6. The value of this feature will be demonstrated by comparing the Vespo method with other published methods where Vespo uses many fewer model evaluations (i.e. higher speed), and comparing the precision. Consistency is evaluated by examining the variation among multiple runs.

The adaptability goal is also related to the tradeoff of speed and precision. It is common that optimization methods include control parameters that guide the search, e.g. the mutation rate in a GA. The values of these parameters can strongly influence the speed, precision, and consistency of the search, and may require experimentation in finding appropriate values for a given problem. Such experimentation can be good for precision, since it produces a good end result, but may be bad for speed, since the experimentation requires many model evaluations. As discussed above, speed has a high priority in conceptual design, and so Vespo uses an adaptive strategy for control parameters. The only parameters specified by the user are the size of the solution population, and the number of generations in an optimization run.

The remainder of the paper is organized as follows. Section 2 describes and explains the details of the algorithm. Section 3 presents application of the method to examples. The examples begin with comparisons of two previously published benchmark problems, and then present variations on those problems that include enumerative topology optimization. Section 4 closes with a summary of results and conclusions.

2. The Vespo algorithm

2.1. Overview

Like other population-based metaheuristic optimization methods, Vespo's version of an ES begins with a randomly generated population of solutions and then evolves that population through generations of incremental improvement. Vespo belongs to the family of ESs that produces a new solution solely through mutation of a single existing solution, without a recombination operator