Evolutionary truss topology optimization using a graph-based parameterization concept

Originally published in:
Structural and Multidisciplinary Optimization 32(4), https://doi.org/10.1007/s00158-006-0028-8

Struct Multidisc Optim (2006) 32: 313–326
DOI 10.1007/s00158-006-0028-8

RESEARCH PAPER

M. Giger · P. Ermanni

Evolutionary truss topology optimization using a graph-based parameterization concept

Received: 8 March 2005 / Revised manuscript received: 14 November 2005 / Published online: 5 August 2006

Abstract A novel parameterization concept for the optimization of truss structures by means of evolutionary algorithms is presented. The main idea is to represent truss structures as mathematical graphs and directly apply genetic operators, i.e., mutation and crossover, on them. For this purpose, new genetic graph operators are introduced, which are combined with graph algorithms, e.g., Cuthill–McKee reordering, to raise their efficiency. This parameterization concept allows for the concurrent optimization of topology, geometry, and sizing of the truss structures. Furthermore, it is absolutely independent from any kind of ground structure normally reducing the number of possible topologies and sometimes preventing innovative design solutions. A further advantage of this parameterization concept compared to traditional encoding of evolutionary algorithms is the possibility of handling individuals of variable size. Finally, the effectiveness of the concept is demonstrated by examining three numerical examples.

Keywords Truss topology optimization · Mathematical graph · Structural optimization · Parameterization · Evolutionary algorithms

1 Introduction

In the field of truss topology optimization, a variety of methods have been developed in the last decades. A lot of research has been done on the ground structure approach initiated by Dorn et al. (1964) where the members and the nodes are selected from a highly connected ground structure. The cross-sectional areas of the members are considered as continuous design variables, whereas the nodal locations are fixed (e.g., Gou et al. 2001) or can be moved (e.g., Wang et al. 2002 or Gou et al. 2003) to minimize the mass or the compliance of the truss structure. Most often, the optimization objective is restricted to comply with stress and eigen-frequency constraints as well as local and global stability requirements as can be seen, for example, in Pedersen and Nielsen (2003). These types of constraints extremely complicate the search for the global optimum due to the phenomenon of singular topologies. A thorough summary of many methods dealing with design-dependent constraints, e.g. the ϵ-relaxation approach by Cheng and Guo (1997), can be found in Rozvany (2003). Furthermore, stress ratio and compliance based methods, e.g., fully stressed design (FSD) or uniform energy distribution (UED), have been thoroughly investigated. A critical review of these and further popular methods, e.g., evolutionary structural optimization (ESO, see Xie and Steven 1997) or adaptive biological growth (ABG), can be found in Rozvany (2001).

Beneath all the above-mentioned methods, the application of genetic algorithms (GA) to truss topology optimization has been investigated in several publications. The well-known basic concept of GA was introduced by Holland (1975), and the basic terminology of genetic search and its principal components are discussed by Goldberg (1989). The traditional binary encoding is presented by Hajela (1992), whereas a more modern overview of applications of evolutionary algorithms (EA) to design optimization can be found in Bentley (1999). Hajela and Lee (1995) use GA to develop near-optimal topologies by subdividing the optimization task into two stages. The first stage generates a number of kinematically stable truss topologies, whereas the second stage is dedicated to the member sizing to get a minimum weight structure. Azid et al. (2002) employ evolutionary genetic search in combination with a slightly modified ground structure approach for the layout optimization of a three-dimensional truss and Lingyun et al. (2004) successfully apply a niche hybrid genetic algorithm to truss optimization with frequency constraints. Kawamura et al. (2002) present an interesting representation approach by using triangular elements as basic design entity. The topologies produced by this method are guaranteed to have neither needless members nor undesirable overlaps between members and only stable structures can occur. All these examples show that GAs are also well-suited for the optimization of truss structures, al-

M. Giger (✉) · P. Ermanni
Centre of Structure Technologies, Leonhardstrasse 27, ETH Zurich, 8092 Zurich, Switzerland
e-mail: mgiger@ethz.ch
e-mail: permanni@ethz.ch

though a global optimum solution can hardly be found. Also, all these examples are based on binary or real-valued encoding, i.e., the design variables are organized in vectors, which are manipulated by genetic operators (crossover and mutation) according to the basic theory of EA.

Basically, the topology of a truss structure can also be considered and modeled as a mathematical graph (West 2001). The nodes of the truss structure can be regarded as vertices, whereas the members correspond to the edges of the graph. Kawamoto et al. (2004) have used the graph representation in combination with the ground structure approach to perform topology optimization of symmetric mechanisms. Their idea is to embed some topological graphs into the ground structure to reduce the complexity of the optimization problem.

In this paper, a novel combination of evolutionary algorithms and mathematical graph theory is applied to truss topology optimization. The complete truss topology is represented by a graph, whereas each vertex corresponds to a node and each member is described by an edge. In other words, instead of holding the information of the truss structure in a conventional one-dimensional genotype, the graph itself is considered as the genotype, and special operators are directly applied on it. Additionally, some typical graph algorithms, e.g., Cuthill–McKee reordering and connectivity analysis, are used to improve the optimization efficiency or filter out globally unstable solutions. The main motivation to investigate this combination of methods is the independence of any kind of ground structure. Furthermore, this approach overcomes the ordinary restriction of having constant length genotypes for genetic search, i.e., the number of design entities must remain unchanged. The only way to remove members from the truss structure is to let the cross-sectional area be null or to use a kind of a binary gene determining whether the member exists or not. Only Ryoo and Hajela (2004) have investigated a binary-encoded representation concept based on variable length genotypes. The graph representation approach shows much more flexibility, as the size of the graphs can arbitrarily change during the optimization process and it is absolutely not a problem to mate graphs of unequal size.

A basic introduction to mathematical graph theory is given in Section 2 and in Section 3 the graph genotype as representation concept for EA is defined. Section 4 deals with genetic graph operators required for mutating and mating graph individuals. Finally, the Section 5 presents three planar numerical examples to demonstrate the strength of the concept at hand.

2 Basic notions of graph theory

It would be immoderate to give a thorough introduction to the mathematical graph theory at this point, as it is a field of research of inconceivable complexity. Thus, this section only covers the most important notions and concepts that are used in the scope of this publication.

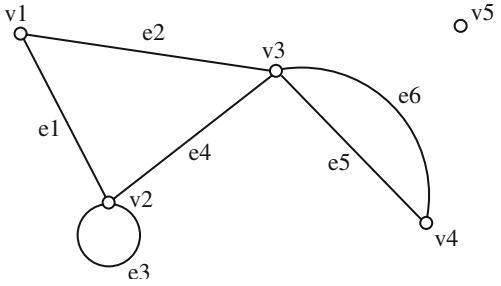

Fig. 1 Graph $G(V, E)$

In literature, graphs are defined in slightly different ways, e.g., Foulds (1994), Chen (1997), or Balakrishnan and Ranganathan (2000). A brief definition is given here.

Definition 1 A graph $G(V, E)$ is an ordered pair, where V is a finite, nonempty set whose elements are termed vertices, and where E is a set of unordered pairs of distinct vertices of V. Each element $e=(u,v) \in$ E (where u, v $\in$ V), is called an edge and is said to be incident with its vertices u and v. Also, the vertices u and v are then incident with e.

There are some other important notions that will be explained with an example. Consider the graph $G(V, E)$ in Fig. 1 in which

$$V = \{v1, v2, v3, v4, v5\}$$
$$E = \{(v1, v2), (v1, v3), (v2, v2), (v2, v3), (v3, v4)_1, (v3, v4)_2\} \quad (1)$$

Basically, vertices u and v in V are *adjacent* to each other if, and only if, there is an edge in E with u and v as its endpoints. For example, the vertices $v1$ and $v2$ are adjacent, as they are connected by the edge $e1$. A vertex having no incident edges, i.e., it is not connected to any other vertex in V, is called an *isolated vertex* (e.g., $v5$). Two edges of a graph are termed a set of *parallel edges*, if they have the same endpoints, e.g., $e5 = (v3, v4)_1$ and $e6 = (v3, v4)_2$. An edge for which the two endpoints are the same, e.g., $e3 = (v2, v2)$, is called a *loop* at the common vertex. Finally, the number of vertices and edges in the graph $G(V, E)$ are denoted the *order* and the *size*, respectively. The graph $G(V, E)$ in Fig. 1 has order $n(G(V, E)) = 5$ and size $m(G(V, E)) = 6$.

Moreover, it is possible to assign properties to graphs, i.e., to its vertices and edges. A graph is said to be *labeled*, if its $n(G)$, vertices are distinguished from one another by labels $v1, v2, \ldots, vn$ as it is done in the example. Furthermore, arbitrary additional properties can be attached to each vertex as it is for example done for the so-called graph coloring problem, where each vertex is colored. Analogously, the concept of a *weighted graph* associates a nonnegative number $w(e)$ to each edge that is called *edge weight*. The sum of all edge weights is, therefore, the total weight of the graph.

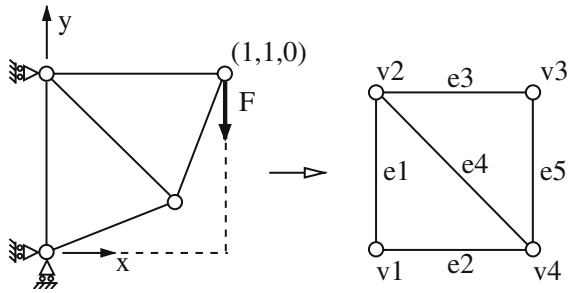

Fig. 2 Simple truss structure and the schematic illustration of its corresponding graph genotype

3 The graph genotype

The graph genotype has to comply with some requirements to make it usable as representation concept for the optimization of truss structures. It must represent the position of all nodes as well as the location and the cross-sectional area of all members to explicitly define its corresponding truss structure.

3.1 Parameterization requirements

In Fig. 2 a classic two-dimensional truss structure is depicted with its corresponding graph genotype. For each node of the structure, three different data types need to be known:

- a unique identifier,
- the planar or spatial coordinates,
- and a boolean parameter determining whether the node can be moved during the optimization process or not. Clamped or loaded nodes must not be relocated.

Additionally, some information about each member is required:

- the two unique identifiers of the member's nodes, and
- the cross-sectional area.

All the above-mentioned information is required to unambiguously define the truss structure and, therefore, has to be represented by the graph genotype. According to Balakrishnan and Ranganathan (2000) a *simple graph* has no loops and no multiple, i.e., parallel edges. Loops do not make sense in a truss structure, as they only contribute to the overall mass but not to the structural stiffness. Parallel edges are undesired because they would unnecessarily increase the complexity of the structure, and they could easily be replaced by single edges. Furthermore, the requirement of a unique identifier for each node can be realized by using a labeled graph, whereas it is reasonable to choose integers as labels. The coordinates of the nodes and the boolean parameter are assigned to the graph genotype's vertices by introducing ordinary vertex properties. As already stated in Definition 1, each edge is defined by an unordered pair of incident vertices. In practice, these vertices are identified by their labels; hence, each edge can be represented as a pair of labels. Finally, the cross-sectional area of each member can easily be mapped to the graph genotype by interpreting it as an edge weight. In summary, the graph genotype needs to be a simple, labeled, and weighted graph supplemented with additional vertex properties. Each vertex holds properties defining its label (unique integer), the position (three floating-point numbers for the spatial coordinates), and whether it is allowed to move or not (boolean variable), whereas every edge holds the information about its endpoints (labels of respective nodes) and its cross-sectional area.

Table 1 Node/vertex and member/edge properties

Vertex	Label	x	y	Movable	Edge	Label 1	Label 2	Weight /area
$v1$	1	0	0	False	$e1$	1	2	1
$v2$	2	0	1	False	$e2$	1	4	1
$v3$	3	1	1	False	$e3$	2	3	1
$v4$	4	[0,1]	[0,1]	True	$e4$	2	4	1
					$e5$	3	4	1

Recalling the example illustrated in Fig. 2, the Table 1 lists all required information. The members should all have a cross-sectional area of unity. Only vertex $v4$ is neither clamped nor loaded; thus, its position can be anywhere in the square design space.

Rozvany (1997) defines *layout optimization* as the simultaneous selection of the optimal *topology* (spatial sequence or connectivity of members), *geometry* (location of intersections), and *cross-sectional dimensions* (sizing). The graph genotype allows for the concurrent optimization of all these aspects of structural optimization, although this leads to a tremendous complexity of the optimization task. The topology of a truss structure is defined by the endpoints of each member, which are represented by the labels in the graph genotype. By either changing label 1 or label 2 of an edge (or both) or adding and removing members, the topology can easily be modified. The coordinates of the nodes define the geometry of the truss structure. It is important to strictly distinguish between clamped or loaded nodes—they must not be moved—and free nodes, which can be placed anywhere in the design space. Consequently, only the coordinates of movable nodes are subject to changes during the optimization process. Finally, the sizing optimization is realized by modifying the edge weights of the respective members. Needless to say, separate optimization of the above-mentioned properties is possible, but not recommended, as a separately optimized topology may no longer be optimal if the geometry is changed.

3.2 Graph genotype analysis

Each graph genotype produced during the optimization process has to be mapped to a finite element (FE) model for the evaluation of its fitness composed of objective and constraint values. Unfortunately, not each graph genotype (or *individual* in terms of evolutionary optimization) can be successfully mapped to a running FE model. After applying genetic operators, as they will be presented in Section 4, indi-

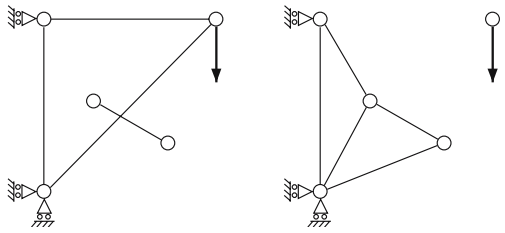

Fig. 3 Two undesired topologies filtered out due to violation of precondition 1 (*left*) and 2 (*right*), respectively

viduals can occur representing novel truss topologies, which would fail in the evaluation step. Thus, there are several obvious preconditions a graph genotype has to fulfill to guarantee feasibility of the truss structure:

1. no non-connected subgraphs[1] may occur,
2. all the clamped and loaded vertices (nodes) must be connected, and
3. the represented truss topology must be statically determinate.

It would be absolutely useless to start the mapping and evaluation process for individuals not complying with these requirements; the optimization efficiency would be decisively decreased. Such individuals as depicted, for example, in Fig. 3 need to be filtered out before by applying graph analysis algorithms.

Connected components A *path* is a sequence of vertices where each vertex is connected by an edge to the subsequent vertex in the path. If there exists a path from vertex u to v, then vertex v is said to be *reachable* from u. A *connected component* is a group of vertices in an undirected[2] graph, which are all reachable from one another, whereas a single vertex is considered to form the smallest possible component. The connected-components algorithm (based on the depth-first search algorithm; for details, see Siek et al. 2002) is able to quickly determine such connected components. Not only is the total number of connected components given, but also the grouping of the vertices is provided. Thus, the connected-components algorithm is perfectly suited to check the first and the second precondition. To comply with the first precondition, the number of connected components must be either exactly equal to one or, if there are more connected components, only one of them may contain more than one vertex. Furthermore, the second precondition says that no clamped or loaded nodes may be represented by an isolated vertex. Consequently, all the corresponding vertices must be contained in the same connected component.

Statical determinacy The statical determinacy of a truss structure can be estimated by applying Maxwell's algebraic rule (see, e.g., Fowler and Guest 2000 and references therein). Generally, the rule can be formulated as

$$f = d \cdot n(G(V, E)) - m(G(V, E)) - s, \tag{2}$$

where d is the dimensionality and s is the number of supports. If $f = 0$, at least a necessary but not in general sufficient condition for establishing statical determinacy is fulfilled. Individuals for which it holds $f < 0$ are statically overdetermined and can also be considered as feasible solutions. However, each individual is checked for compliance with (2), and the FE evaluation of individuals not fulfilling the condition is omitted. Nevertheless, all these individuals violating at least one precondition are consciously kept in the optimization process and the population, respectively, as their "genetic material" is not necessarily useless. It would be a possibility to repair individuals not complying with the preconditions by applying further graph algorithms and inserting edges to stabilize the truss structures, but practice has shown that in an optimization run only a low percentage of individuals are infeasible.

3.3 Implementation

The entire code is implemented in C++ and is based on four powerful libraries. As an optimization engine the evolving objects[3] (EO) library is employed providing the basic functionalities required for evolutionary computation. This library allows for optimizing any kind of data structure such as the graph genotype. The implementation of the graph genotype itself is a combination of the Boost Graph Library[4] (BGL) and the eoUniGene concept introduced by König (2004). The BGL provides a standardized generic interface for manipulating and traversing general purpose graphs. The graph genotype is based on such a basic graph and enhanced with the functionalities of the eoUniGene concept. Actually, the basic idea of eoUniGene is to provide a variety of gene types that can be combined to a heterogeneous list for parameter optimization with EA. Instead of using such a list, the appropriate gene types are integrated into the graph. For example, the vertex labels are represented by integer type genes, the so-called *int_genes*, providing lower and upper limits as well as basic operator functionalities to change the genes' values within these limits. Analogously, the vertex coordinates and the edge weights are described by *float_genes* having basically the same properties as the *int_genes* but for floating-point numbers.

Due to the huge number of evaluations typically required for evolutionary optimizations, FELyX[5], an object-oriented FE code written in C++, is utilized allowing for extremely fast mapping and evaluation procedures (no commercial software is required). The graph genotype properties are mapped to 3D-spar uniaxial tension-compression elements with three degrees of freedom at each node. For any computation, the

[1] A graph H is called a *subgraph* of G, if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$

[2] All edges are *unordered* pairs of vertices, i.e., they do not have an orientation.

[3] http://eodev.sourceforge.net

[4] http://boost.org, Siek et al. (2002)

[5] http://felyx.sourceforge.net

self-weight of the truss members is not taken into consideration. The build-up of the FE model is completed by inserting the boundary conditions of clamped and loaded nodes. Finally, parallel computing is made possible by including the Parallel Virtual Machine[6] (PVM) library.

4 Genetic graph operators

When using evolutionary algorithms as optimization strategy, two different kinds of genetic operators need to be implemented. The mutation operators only act on a single individual at a time, whereas only few parameter values of the design variables are changed, i.e., mutated. The crossover operators take two individuals and create two new individuals out of them by mixing the properties of the original individuals. It is crucial that the genetic operators cover all aspects of layout optimization, i.e., they have to allow for changes concerning the topology, the geometry, and the sizing of the truss structures, otherwise, a concurrent optimization is not realistic. There are a lot of possible operators acting on vertices and edges and their properties, respectively. The only restriction is to keep the number of vertices (connected or unconnected) constant, otherwise, some graph algorithms cannot be applied anymore. Most of the genetic operators are briefly explained, as they are the core of any efficient evolutionary optimization process. For the illustration of the effect of mutation and crossover operators, it is worth introducing first the *adjacency matrix*.

4.1 Adjacency matrix

The adjacency matrix is a storage format of mathematical graphs. Additionally, it serves the purpose of visualizing the structure of a graph, i.e., the adjacency matrix shows which vertices are connected by edges. The effect of genetic topology and sizing operators can be shown by means of the adjacency matrix; only the effect of genetic geometry operators changing the node locations cannot be visualized.

In Foulds (1994), the traditional adjacency matrix is defined in the following way.

Definition 2 The adjacency matrix $A = (a_{ij})_{n\times n}$, of a vertex-labeled graph G, with n(G) vertices, is the matrix in which $a_{ij} = 1$ if vertex v_i is adjacent to v_j in G and $a_{ij} = 0$ otherwise, where v_i and v_j are vertices of G.

In some cases, it is reasonable to replace $a_{ij} = 1$ by the edge weights $a_{ij} = w_{ij}$ of the respective edge to insert more information into the adjacency matrix of the graph. For example, the traditional and the weighted adjacency matrices of the simple graph shown in Fig. 1 are

$$A = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}, \quad A_w = \begin{bmatrix} 0 & w_{12} & 0 & w_{14} \\ w_{21} & 0 & w_{23} & w_{24} \\ 0 & w_{32} & 0 & w_{34} \\ w_{41} & w_{42} & w_{43} & 0 \end{bmatrix}. \tag{3}$$

It is quite obvious that A and A_w, respectively, must be symmetric, i.e., $w_{ij} = w_{ji}$. Furthermore, the entries on the leading diagonal equal zero due to the simplicity of the graphs not allowing loops. A lot more information is contained in this kind of graph description, e.g., a block diagonal matrix indicates a non-connected graph (subgraphs exist), but in the scope of this paper the given definition is sufficient.

4.2 Mutation operators

Generally, mutation can be considered as a transformation $\mathcal{M} : A_o \rightarrow A_{mut}$ of the respective graph adjacency matrix.

4.2.1 Topology mutation operators

Random removal or insertion of edges Probably the simplest possible mutation operator concerning topology optimization is the random insertion or removal of edges. A random number generator (RNG) is used to determine an edge to be removed. Analogously, two vertices are chosen by RNG, which are newly connected with an edge, whereas the edge weight, i.e., the cross-sectional area of the newly created member, is also determined by chance. In the example, below the edge $(v2, v3)$ is removed while another edge $(v1, v3)$ is inserted.

$$A_o = \begin{bmatrix} 0 & w_{12} & 0 & w_{14} \\ w_{12} & 0 & w_{23} & w_{24} \\ 0 & w_{23} & 0 & w_{34} \\ w_{14} & w_{24} & w_{34} & 0 \end{bmatrix} \rightarrow A_{mut} = \begin{bmatrix} 0 & w_{12} & w_{13} & w_{14} \\ w_{12} & 0 & 0 & w_{24} \\ w_{13} & 0 & 0 & w_{34} \\ w_{14} & w_{24} & w_{34} & 0 \end{bmatrix}$$

Removal or connection of free edges Individuals might happen to occur having vertices with only one incident edge. Obviously, such edges are not reasonable and do not contribute to an optimum truss structure. Consequently, such vertices need to be completely isolated (if they are not loaded or clamped) or a second edge (or also a third edge in the three-dimensional case) needs to be inserted to stabilize the truss structure. Newly inserted edges are initialized with random edge weights.

Flipping edges This operator changes the topology by flipping edges, i.e., one endpoint of an existing edge is reconnected to another vertex, whereas the new endpoint is

[6] http://www.csm.ornl.gov/pvm

determined by chance. The edge weight of flipped edges is not changed as can be seen in the example below:

$$A_o = \begin{bmatrix} 0 & w_{12} & 0 & w_{24} \\ w_{12} & 0 & w_{23} & w_{24} \\ 0 & w_{23} & 0 & w_{34} \\ w_{14} & w_{24} & w_{34} & 0 \end{bmatrix} \rightarrow A_{mut} = \begin{bmatrix} 0 & w_{12} & w_{13} & w_{14} \\ w_{12} & 0 & w_{23} & w_{24} \\ w_{13} & w_{23} & 0 & 0 \\ w_{14} & w_{24} & 0 & 0 \end{bmatrix},$$

where $w_{13} = w_{34}$.

Merging or disconnection of vertices A general problem, when applying genetic operators, is that the structure can hardly become simpler with ordinary removal of edges. If a truss structure is statically determinate, one cannot simplify it by removing a single edge, as this transforms the truss topology to a mechanism. Therefore, two similar strategies are implemented as mutation operators. Either a vertex is completely disconnected from the remaining structure, i.e., all the edge weights in one row and its corresponding column in the adjacency matrix are deleted, or the incident edges of two vertices are merged as can be seen in the example below ($v3$ and $v4$ are merged):

$$A_o = \begin{bmatrix} 0 & w_{12} & 0 & w_{14} \\ w_{12} & 0 & w_{23} & w_{24} \\ 0 & w_{23} & 0 & w_{24} \\ w_{14} & w_{24} & w_{34} & 0 \end{bmatrix} \rightarrow A_{mut} = \begin{bmatrix} 0 & w_{12} & w_{13} & 0 \\ w_{12} & 0 & w_{23} & 0 \\ w_{13} & w_{23} & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

where $w_{13} = w_{14}$. If both involved vertices are movable, the new vertex lies exactly in-between the two original vertices.

4.2.2 Geometry mutation operators

The geometry of the truss structure can only be influenced by the vertex locations. Thus, mutating the vertex locations leads to a direct geometric variation of the original individual. There are two types of mutation, i.e., the *uniform* and the Gaussian mutation strategies, which are applied to all vertex location coordinates with a certain probability. For uniform mutation, a new value is computed by randomly selecting a value within the given limits, where in the majority of cases the limits are given by the boundaries of the design space. The Gaussian mutation is well-suited for the fine tuning of a structure, as the new value of the respective coordinate is determined by means of a rather narrow Gaussian distribution ($\mathcal{N}(0, \sigma)$) with mean value equal to zero and standard deviation σ that is added to the actual value (g)

$$g = g + \mathcal{N}(0, \sigma), \tag{4}$$

where the standard deviation is a predefined parameter for each component of the vertex coordinate vector. The standard deviation remains constant until the stopping criterion is reached. Changes of the standard deviation values can only be made by restarting the optimization with modified values.

4.2.3 Sizing mutation operators

The sizing mutation operators work very similar to the geometry mutation operators. Analogously, a uniform and a Gaussian mutation strategy (see (4)) is implemented for mutating the edge weights of the truss structure.

4.3 Crossover operators

First, the Cuthill–McKee reordering is discussed, because this graph algorithm is extremely important for the implementation of efficient crossover operators. Subsequently, some crossover operators are presented, which can be considered as transformations $\mathcal{C} : A_{p1}, A_{p2} \rightarrow A_{o1}, A_{o2}$ of parent to offspring individuals.

4.3.1 Cuthill–McKee reordering

The basic purpose of the Cuthill–McKee reordering algorithm (Cuthill and Mckee 1969) is to reduce the bandwidth of a sparse symmetric matrix, e.g., the adjacency matrix. The bandwidth of an undirected graph and its adjacency matrix, respectively, is the maximum labeling distance between two adjacent vertices. In case of a graph $G(V, E)$ with labeled vertices from 1 to $n(G)$ the bandwidth $B(G)$ is defined as

$$B(G) = max\{|\, label[u] - label[v]\,| \quad |\, u, v \in V\}. \tag{5}$$

The bandwidth of the adjacency matrix can, thus, be minimized by reordering the labels assigned to each vertex, whereas a *starting vertex* needs to be provided.

In Fig. 4, a sample truss topology is presented with arbitrary vertex labeling on the left and reordered labeling on the right. The corresponding adjacency matrices are given below. The zeros beneath the leading diagonal are omitted for better perceptibility.

$$A_{arb} = \begin{bmatrix} 0 & 1 & 1 & & & & \\ & 0 & & & 1 & 1 & \\ 1 & & 0 & 1 & 1 & & \\ 1 & & 1 & 0 & 1 & & 1 \\ & & 1 & 1 & 0 & 1 & 1 \\ & 1 & & & 1 & 0 & 1 \\ & 1 & & 1 & 1 & 1 & 0 \end{bmatrix} \rightarrow A_{reordered} = \begin{bmatrix} 0 & 1 & 1 & & & & \\ 1 & 0 & 1 & 1 & & & \\ 1 & 1 & 0 & 1 & 1 & & \\ & 1 & 1 & 0 & 1 & 1 & \\ & & 1 & 1 & 0 & 1 & 1 \\ & & & 1 & 1 & 0 & 1 \\ & & & & 1 & 1 & 0 \end{bmatrix}$$

The bandwidth of the first adjacency matrix is $B(A_{arb}) = 5$ and the bandwidth of the reordered adjacency matrix is $B(A_{reordered}) = 2$.

Crossover operators, as will be presented in subsequent sections, work most efficiently if they exchange substructures of the original trusses. The purpose of applying Cuthill–McKee reordering is to precondition the graphs representing truss structures in terms of similarity. To achieve maximum similarity, the starting vertices of the reordering algorithm of both original graphs have to be the same, i.e., they should have the same coordinates. It is, therefore, most reasonable to choose a non-movable vertex (clamped or loaded)

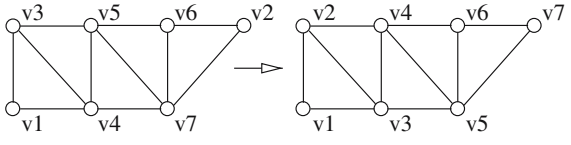

Fig. 4 Truss topologies with arbitrary (*left*) and reordered (*right*) vertex labeling (starting vertex $v1$)

at the boundary of the design space as a starting vertex. A similar vertex labeling of the original graphs guarantees that the truss structures can be split at similar positions; hence, more realistic structures occur after the crossover operations.

4.3.2 Topology crossover operators

Proportional edge crossover This crossover type is very similar to an ordinary 1-point crossover for traditional one-dimensional list genotypes, but it also works for graphs of unequal edge numbers ($m(G_{p1}) \neq m(G_{p2})$). After Cuthill–McKee reordering of both parent individuals and determining a random number $r \in [0, 1]$, the number of edges of the parent graph genotypes (A_{p1} and A_{p2}) are multiplied by r and rounded to integer numbers to get the split positions in the adjacency matrices. Two offspring graph genotypes (A_{o1} and A_{o2}) are then created by exchanging all edges following after the respective split positions as can be seen in the example below. The value of r is set to 0.52 and the edge weights of parent A_{p1} are of equal unity and the edge weights of A_{p2} are equal to two for simpler identification of parents and their offspring. The subscripts indicate the number of the respective edge.

$$A_{p1} = \begin{bmatrix} 0 & 1_1 & 1_2 & 1_3 & & \\ 1_1 & 0 & 1_4 & & & \\ 1_2 & 1_4 & 0 & 1_5 & 1_6 & \\ 1_3 & & 1_5 & 0 & 1_7 & \\ & & 1_6 & 1_7 & 0 & \\ & & & & & 0 \end{bmatrix} \quad A_{p_2} = \begin{bmatrix} 0 & 2_1 & 2_2 & & & \\ 2_1 & 0 & 2_3 & 2_4 & & \\ 2_2 & 2_3 & 0 & 2_5 & 2_6 & 2_7 \\ & 2_4 & 2_5 & 0 & 2_8 & 2_9 \\ & & 2_6 & 2_8 & 0 & 2_{10} \\ & & 2_7 & 2_9 & 2_{10} & 0 \end{bmatrix}$$

The split position for the first parent is after the fourth edge and the second parent is split after the fifth edge. Consequently, this leads to offspring of the following topology.

$$A_{o1} = \begin{bmatrix} 0 & 1_1 & 1_2 & 1_3 & & \\ 1_1 & 0 & 1_4 & & & \\ 1_2 & 1_4 & 0 & & 2_6 & 2_7 \\ 1_3 & & & 0 & 2_8 & 2_9 \\ & & 2_6 & 2_8 & 0 & 2_{10} \\ & & 2_7 & 2_9 & 2_{10} & 0 \end{bmatrix} \quad A_{o2} = \begin{bmatrix} 0 & 2_1 & 2_2 & & & \\ 2_1 & 0 & 2_3 & 2_4 & & \\ 2_2 & 2_3 & 0 & 2_5 & 1_6 & \\ & 2_4 & 2_5 & 0 & 1_7 & \\ & & 1_6 & 1_7 & 0 & \\ & & & & & 0 \end{bmatrix}$$

In Fig. 5, the parents and their offspring are depicted.

Uniform edge exchange The second genetic topology operator exchanges incident edges of some randomly chosen vertices. After the Cuthill–McKee reordering is executed, an equally labeled vertex is chosen from each parent genotype. The respective rows and columns of these vertices are then simply exchanged leading to topologically novel offspring individuals. The edge weights of all incident edges of both vertices remain unchanged. As an example, the incident edges of vertices $v6$ of the parent individuals A_{p1} and A_{p2} in Fig. 5 are exchanged.

$$A_{o1} = \begin{bmatrix} 0 & 1_1 & 1_2 & 1_3 & & \\ 1_1 & 0 & 1 & & & \\ 1_2 & 1_4 & 0 & 1_4 & 1_5 & 2_7 \\ 1_3 & & 1_4 & 0 & 1 & 2_9 \\ & & 1_5 & 1 & 0 & 2_{10} \\ & & 2_7 & 2_9 & 2_{10} & 0 \end{bmatrix} \quad A_{o2} = \begin{bmatrix} 0 & 2_1 & 2_2 & & & \\ 2_1 & 0 & 2_3 & 2_4 & & \\ 2_2 & 2_3 & 0 & 2_5 & 2_6 & \\ & 2_4 & 2_5 & 0 & 2_8 & \\ & & 2_6 & 2_8 & 0 & \\ & & & & & 0 \end{bmatrix}$$

4.3.3 Geometry crossover operators

The aim of crossovers altering the geometry of some individuals is to average two parent individuals regarding their vertex coordinates. In other words, after the Curhill–McKee reordering is executed, the operator iterates over all vertices vi of both parent individuals and geometrically mates them with a certain probability. Provided that their vertex coordinates are allowed to be changed, each component of the vertex coordinates is modified. Let $\mathbf{x_{p1_{vi}}}$ be the vector of planar or spatial coordinates of a given vertex of the first parent and $\mathbf{x_{p2_{vi}}}$ the coordinates of the corresponding vertex of the second parent, then the new coordinates of two offspring individuals $\mathbf{x_{o1_{vi}}}$ and $\mathbf{x_{o2_{vi}}}$ are calculated as

$$\begin{aligned} \mathbf{x_{o1_{vi}}} &= c \cdot \mathbf{x_{p1_{vi}}} + (1 - c) \cdot \mathbf{x_{p2_{vi}}} \\ \mathbf{x_{o2_{vi}}} &= (1 - c) \cdot \mathbf{x_{p1_{vi}}} + c \cdot \mathbf{x_{p2_{vi}}}, \end{aligned} \tag{6}$$

where the constant $c \in [0, 1]$ is randomly determined. This operator is implemented in two slightly different implementations. When using the *segment* type operator, the constant c is determined only once and remains constant until two parents are completely processed. In contrast, the *hypercube* type operator always determines a new value for c each time two vertices of the parent individuals are mated.

4.3.4 Sizing crossover operators

The sizing crossover operators work very similarly to the geometry crossover operators. Analogously, a segment crossover strategy and a hypercube crossover strategy are implemented for mating the edge weights of the parent individuals ($w_{p1_{ei}}$ and $w_{p2_{ei}}$). The Cuthill–McKee reordering is also applied for the preconditioning of the parent individuals and the new edge weights are calculated as

$$\begin{aligned} w_{o1_{ei}} &= c \cdot w_{p1_{ei}} + (1 - c) \cdot w_{p2_{ei}} \\ w_{o2_{ei}} &= (1 - c) \cdot w_{p1_{ei}} + c \cdot w_{p2_{ei}} \end{aligned}. \tag{7}$$

Again, it holds $c \in [0, 1]$.

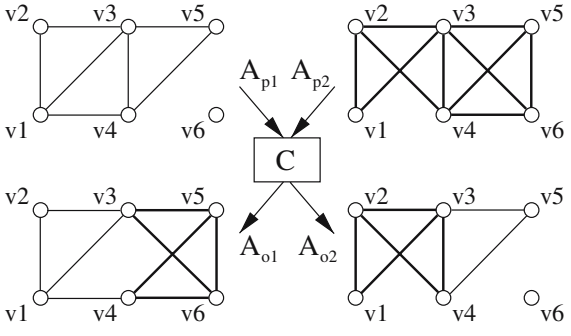

Fig. 5 Parents and offspring of a proportional edge crossover

5 Numerical examples

Three numerical examples are presented in this section. The first example is a variation of the ten-bar truss problem known from several publications (e.g., Gou et al. 2001; Stolpe and Svanberg 2003, and Pyrz 2004), which is examined for validation purposes. The second example is very similar, but the size of the structure is increased and it illustrates the independence from any kind of ground structure. Finally, the third example discusses the optimization of a 52-bar planar truss known from several publications (Wu and Chow 1995; Lemonge 1999, and Lemonge and Barbosa 2004). The full capacity of the graph representation is demonstrated by reformulating this example in a much freer way allowing arbitrary topologies and geometries of the structure. For all presented truss structures circular cross-sections are used, but the concept is absolutely not limited to such simple geometries. These examples are chosen to demonstrate the applicability of the graph representation to simple examples, but it is not intended to compete with other algorithms in terms of efficiency yet. Performance tests will be subject to further research.

5.1 The modified ten-bar truss optimization

Most often, the ten members of the ground structure indicated by solid lines in Fig. 6 are optimized by mathematical programming methods. However, this example can also be treated by using the graph genotype concept. In the first step, the positions of the nodes are fixed and only the optimum number of members m and their sizes $a^T = [a_1, a_2, \ldots, a_m]$, i.e., their cross-sectional areas, should be optimized. In contrast to the traditional problem formulation, all of the nodes may be connected to each other, hence, a total number of $m_{max} = n \cdot (n-1)/2 = 15$ members is possible, where $n = 6$ is the total number of nodes. A minimum number of $m_{min} = 4$ is required to form the simplest possible statically determinant truss topology. As the optimum number of members is not known in advance, the graph genotype is allowed to contain between 4 and 15 edges. The formulation of the optimization problem is, therefore, formulated as follows, where the subscript i refers to the i-th member.

$$\begin{aligned} \text{minimize} \quad & O(\mathbf{a}) = \sum_{i=1}^{m} l_i \rho_i a_i \\ \text{subject to} \quad & \sigma_i^{min} \le \sigma_i(a_i) \le \sigma_i^{max}, \\ & \sigma_i(a_i) \ge \sigma_i^{cr}(a_i), \\ & a_i \ge 0, \\ & m_{min} \le m \le m_{max}, \end{aligned} \tag{8}$$

where l_i is the length, ρ_i is the density, and a_i is the cross-sectional area of the i-th member. Furthermore, the stress limit in tension $\sigma_i^{max} \ge 0$ and the stress limit in compression $\sigma_i^{min} \le 0$ need to be observed in each member i. All members have a circular cross-sectional areas, hence, the Euler buckling stress can be calculated as

$$\sigma_i^{cr}(a_i) = -a_i \pi E_i / (2l_i)^2 \tag{9}$$

and the margin of safety against buckling can be defined as

$$ms = \frac{\sigma_i^{cr}(a_i)}{\sigma_i(a_i)} - 1, \tag{10}$$

where $ms < 0$ indicates buckling of the respective member. Due to the academic nature of this optimization problem, the effect of initial imperfections is neglected.

An EA-based optimization requires the definition of a fitness function or value, respectively, assessing the quality of each individual. For that purpose, the fitness formulations introduced by König (2004) are employed, defining the fitness value as a weighted sum of ratings for the optimization objective and the constraints in (8). An illustrative example of the application of this fitness formulation concept can be found in Giger and Ermanni (2005). The detailed implementation is based on lots of experience and would go beyond the scope of this paper.

Results The optimization results are compared to the results of Gou et al. (2001) and Stolpe and Svanberg (2003). Thus, the same data are used for material and geometrical parameters, i.e., the height of the structure is $h = 360$, the external load is given by $F = 100$, and for the material data and the stress limits it holds $\sigma_i^{min} = -20$, $\sigma_i^{max} = 20$, $E_i = 10^4$, and $\rho_i = 0.1$ for all i.

The optimization was randomly initialized, i.e., no prior knowledge was introduced, and run with a population size of 50 individuals over 1,600 generations, which is a moderate number of evaluations for genetic search and such a simple model. The extremely fast computation, processing some hundred individuals per second, led to very short optimization times. The crossover operators were applied with a general probability of 0.7, whereas the general mutation rate was set to 0.25. The probabilities of application of the single mutation and crossover operators as explained in Section 4 were relatively weighted but kept constant during the entire optimization process.

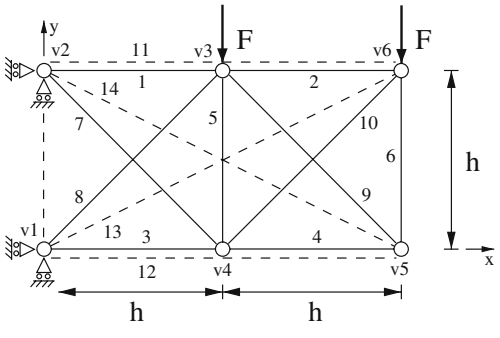

Fig. 6 The ground structure of the ten-bar truss problem (*solid lines*) with five additional members indicated by *dashed lines*

The final result of the optimization using the graph genotype concept is extremely convincing. In Table 2, the result is compared to the published results of Gou et al. (2001) and Stolpe and Svanberg (2003). Obviously, the optimization converges to the same at least local optimum as in Stolpe and Svanberg (2003), whereas they solved the problem with the sequential quadratic programming package SNOPT (Gill et al. 2002). The difference in weight is approximately 4 g only! A further optimization is almost impossible using EA because the constraint values cannot exactly be reached, i.e., the tension members show tension values of $19.90 \leq \sigma_i(a_i) < 20.00$ and the compression members very closely reach the Euler buckling value ($0 < ms \leq 8.4455 \cdot 10^{-4}$). Thus, the solution is almost fully stressed.

It is absolutely obvious that EAs are ill-suited for the fine tuning at the end of the optimization process, as the topology can hardly be changed and only geometric and sizing parameters are adjusted. Additionally, it would be useful to alter the optimization process variables as the optimization runs, because some genetic operators are more useful at the beginning when the topology is determined and some others should be forced in the end for the fine tuning. It is a matter of fact that adaptive (with regard to optimization process variables) as well as hybrid (combination of different optimization methods like stochastic search and mathematical programming) strategies could be very helpful here. However, the graph genotype concept proves to be working.

Optimization including movable nodes This example completely excludes geometry optimization, as all the node positions are fixed. A further optimization was executed enabling the nodes without supports or loading, i.e., vertices $v4$ and $v5$, to move. The result is depicted in Fig. 7, and the cross-sectional areas $\check{a}$ of the members are presented in Table 2.

The most obvious difference between this and the first solution without moving nodes is the topology. One may expect that the topology would remain the same and only the geometry would be changed, but although lots of optimization runs were performed, it was impossible to find a lighter result having the same topology as the original result with static nodes.

Table 2 Resulting area vectors: $\bar{a}$ from Gou et al. (2001), $\hat{a}$ from Stolpe and Svanberg (2003), $\tilde{a}$ from the graph genotype optimization, and $\check{a}$ from the optimization with movable nodes

Member	$\bar{a}$	$\hat{a}$	$\tilde{a}$	$\check{a}$
a_1	5.00000	5.00000	5.00139	12.8538
a_2	0	5.00000	5.00139	9.48097
a_3	70.35876	70.35876	70.3595	72.6613
a_4	40.62165	0	–	27.0939
a_5	57.44769	40.62165	40.6388	26.4674
a_6	40.62165	0	–	49.7662
a_7	14.14214	14.14214	14.14214	2.53085
a_8	0	0	–	–
a_9	7.07107	0	–	2.13747
a_{10}	0	68.31720	68.3187	–
$a_{11} - a_{14}$	–	–	–	–
Weight	8785.79	8553.44	8557.45	5898.23

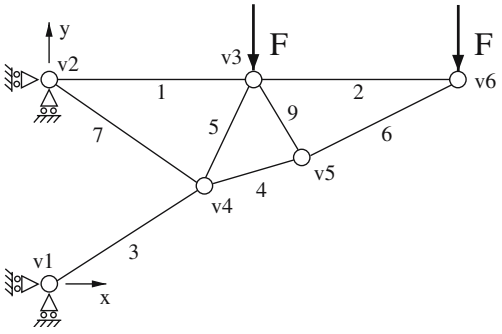

Fig. 7 Optimization result with movable nodes. Final coordinates are $v4$: (299.62/172.99) and $v5$: (453.59/219.50)

Obviously, the concurrent optimization of topology, geometry, and sizing led to a convincing result that could hardly be reached with sequential optimization of topology, geometry, and sizing. Again, the solution is almost fully stressed, i.e., the stresses in tension bars are $19.99 \leq \sigma_i(a_i) < 20.00$ and the compression members are very close to the Euler buckling value ($0 < ms \leq 2.7633 \cdot 10^{-4}$).

5.2 Extended planar truss optimization

The purpose of this example is to increase the complexity of the optimization problem by introducing more loaded nodes. In particular, there is no need to assume a ground structure and, thus, the variety of different topologies is not limited in advance. The problem formulation is exactly the same as shown in (8) except for the maximum number of vertices n and the number of edges m. For this problem it holds $n = 14$, whereas the vertices $v1, v2, \ldots, v7$ cannot move, and $10 \leq m \leq 25$ is considered to be sufficient to allow for all reasonable topologies. The external load $F = 100$ is applied five times, and the loaded nodes are equidistantly distributed over the length of the structure ($h = 360$). The optimization was randomly initialized independent of any kind of ground structure, and all seven movable vertices could be placed anywhere in the design space indicated by dashed lines in Fig. 8.

Result The optimization revealed that the number of potential vertices chosen was too high, as only three vertices incorporated into the optimum structure are also depicted in Fig. 8,

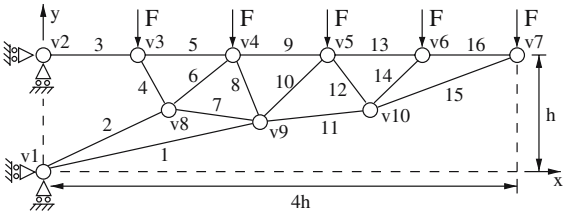

Fig. 8 Optimum result for the extended planar truss optimization. Final coordinates are $v8$: (381.58/188.12), $v9$: (673.41/144.95), and $v10$: (981.52/188.72)

and the remaining four movable vertices were isolated. In Table 3, a summary of the optimization result is given. For the tension members, the axial stresses are listed, and for the compression members, the margins of safety according to (10) are stated.

The structure consists of 16 members and has an overall weight of 36,704 g. The optimization process is totally converged to at least a local optimum solution with almost fully stressed members. The axial stresses in the tension members are extremely close to the limit, and also the compression members have only a minor Euler buckling reserve of approximately 5%. The similarity of these values is caused by the fitness function definition. The compression members could definitely be pushed closer to the limit, but this would require a restart of the optimization with adjusted fitness function definition and decreased standard deviations for Gaussian mutation according to (4). Instead of using EAs for this purpose, a mathematical programming method should be used for the final fine tuning of the structure. Finally, it has to be noted that this optimization result would not have been possible with a ground structure approach only connecting neighboring nodes as in the traditional ten-bar truss problem. The triangle structure $v1$, $v8$, $v9$ could not have been generated. This result clearly shows that the graph genotype concept proves to be working for larger problems and may lead to innovative topologies.

5.3 Planar 52-bar truss optimization problem

The planar 52-bar truss problem shown in Fig. 9 is investigated to demonstrate the applicability of the graph representation to more complex optimization tasks and the results are compared to the results presented in other publications. The total mass of the structure should be minimized for the given load case, i.e., the nodes $v1$ to $v4$ are simply supported and each of the nodes from $v17$ to $v20$ is subjected to loads $F_x = 100$ kN and $F_y = 200$ kN. The maximum allowable stresses in tension as well as in compression members are restricted to 180 MPa. Furthermore, the member areas are linked into 12 groups according to Table 4, and the values of the cross-sectional areas have to be chosen from Table 5. The Young's modulus and the density are equal for all members and their values are 2.07×10^5 MPa and 7,860 kg/m^3, respectively.

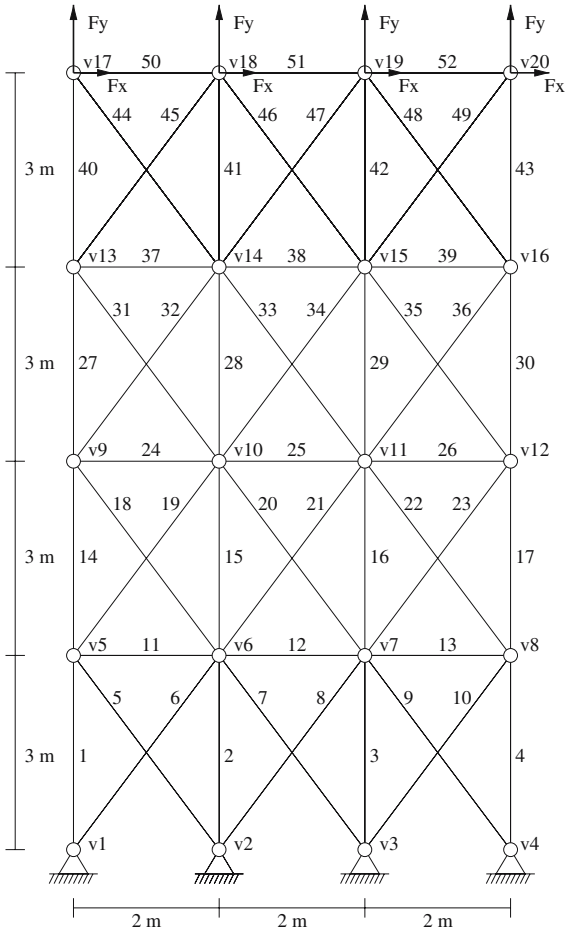

Fig. 9 The planar 52-bar truss ground structure

Table 3 Results of the extended planar optimization example

Member	$\dot{a}$	$\sigma_i(a_i)$	ms
a_1	146.131	–	$4.7825 \cdot 10^{-2}$
a_2	153.063	–	$4.7832 \cdot 10^{-2}$
a_3	60.0126	19.9958	–
a_4	24.1227	–	$4.7978 \cdot 10^{-2}$
a_5	62.7303	19.9975	–
a_6	73.5705	–	$4.7876 \cdot 10^{-2}$
a_7	74.5349	–	$4.7884 \cdot 10^{-2}$
a_8	16.4156	19.9974	–
a_9	33.3882	19.9970	–
a_{10}	58.7885	–	$4.8156 \cdot 10^{-2}$
a_{11}	77.4532	–	$4.7868 \cdot 10^{-2}$
a_{12}	8.17042	19.9961	–
a_{13}	18.3648	19.9949	–
a_{14}	33.1583	–	$4.8027 \cdot 10^{-2}$
a_{15}	95.5592	–	$4.7816 \cdot 10^{-2}$
a_{16}	13.3872	19.9947	–

Results All the presented solutions (see Table 6) satisfy the constraints, hence, the total mass can directly be compared as a solution quality measure. Benchmark 1 from Wu and

Table 4 Member grouping for the 52-bar truss optimization problem

Group	Members
A_1	1, 2, 3, 4
A_2	5, 6, 7, 8, 9, 10
A_3	11, 12, 13
A_4	14, 15, 16, 17
A_5	18, 19, 20, 21, 22, 23
A_6	24, 25, 26
A_7	27, 28, 29, 30
A_8	31, 32, 33, 34, 35, 36
A_9	37, 38, 39
A_{10}	40, 41, 42, 43
A_{11}	44, 45, 46, 47, 48, 49
A_{12}	50, 51, 52

Table 5 The available cross-sectional areas

No.	mm^2	No.	mm^2
1	71.613	33	2,477.414
2	90.968	34	2,496.769
3	126.451	35	2,503.221
4	161.290	36	2,696.769
5	198.064	37	2,722.575
6	252.258	38	2,896.768
7	285.161	39	2,961.284
8	363.225	40	3,096.768
9	388.386	41	3,206.445
10	494.193	42	3,303.219
11	506.451	43	3,703.218
12	641.289	44	4,658.055
13	645.160	45	5,141.925
14	792.256	46	5,503.215
15	816.773	47	5,999.998
16	940.000	48	6,999.986
17	1,008.385	49	7,419.340
18	1,045.159	50	8,709.660
19	1,161.288	51	8,967.724
20	1,283.868	52	9,161.272
21	1,374.191	53	9,999.980
22	1,535.481	54	10,322.560
23	1,690.319	55	10,903.204
24	1,696.771	56	12,129.008
25	1,858.061	57	12,838.684
26	1,890.319	58	14,193.520
27	1,993.544	59	14,774.164
28	2,019.351	60	15,806.420
29	2,180.641	61	17,096.740
30	2,238.705	62	18,064.480
31	2,290.318	63	19,354.800
32	2,341.191	64	21,612.860

Chow (1995) produced the worst result after 60,000 evaluations and was found by using a steady-state GA applying two-point crossovers. The results of benchmark 2 from Lemonge (1999) (20,000 evaluations) and benchmark 3 from Lemonge and Barbosa (2004) (population size 70,250 generations in 20 independent runs) both were obtained by applying a generational GA, whereas benchmark 3 additionally included an adaptive penalty scheme. According to Lemonge and Barbosa (2004), the resulting masses are essentially equal (1,903.366392 kg vs 1,903.366416 kg) with distinct values for some design variables. In fact, the masses are exactly equal, as only the order of the design variables for groups A_3, A_6, A_9, and A_{12} is exchanged.

The graph-based truss optimization leads to a set of six very similar design solutions (see Table 6), all of them having exactly the same mass. The first two solutions are already known from benchmarks 2 and 3, but four further solutions (variations 1 to 4) having the same mass could be found respecting the stress constraint. As already mentioned, the only difference of these solutions is the order of the cross-sectional areas of the horizontal members. These results are obtained by running 20 independent and random initialized runs with a population size of 100 graph individuals over 5,000 generations. This inefficiency can probably be explained with the fact that only sizing mutation operators (Section 4.2.3) and sizing crossover operators (Section 4.3.4) are applied because of the fact that the topology and the geometry of the structure must remain unchanged. The graph representation is definitely not suited to only perform sizing optimization; the binary representation obviously outperforms the presented concept when applied to such optimization tasks in terms of efficiency. Nevertheless, the graph representation is able to cope with this optimization task and finds a set of at least six local optimum solutions. However, the six presented solutions are not exactly identical as the maximum stresses in their members are different. All solutions are very close to the upper limit regarding the tension members, but there are slight differences considering the compression members. For variation 2, the absolute value of the maximum compression stress is lower than for all other solutions and, therefore, this solution has to be considered as being superior to the others.

Table 6 Comparison of the 52-bar planar truss optimization results

Group	Benchmark 1	Benchmark 2	Benchmark 3	Variation 1	Variation 2	Variation 3	Variation 4	Buckling
A_1	4,658.055	4,658.055	4,658.055	4,658.055	4,658.055	4,658.055	4,658.055	5,999.998
A_2	1,161.288	1,161.288	1,161.288	1,161.288	1,161.288	1,161.288	1,161.288	3,703.218
A_3	645.160	*363.225*	*494.193*	*494.193*	*494.193*	*494.193*	*363.225*	1,993.544
A_4	3,303.219	3,303.219	3,303.219	3,303.219	3,303.219	3,303.219	3,303.219	3,703.218
A_5	1,045.159	940.000	940.000	940.000	940.000	940.000	940.000	3,703.218
A_6	494.193	*641.289*	*641.289*	*494.193*	*363.225*	*363.225*	*494.193*	2,180.641
A_7	2,477.414	2,238.705	2,238.705	2,238.705	2,238.705	2,238.705	2,238.705	2,180.641
A_8	1,045.159	1,008.385	1,008.385	1,008.385	1,008.385	1,008.385	1,008.385	3,096.768
A_9	285.161	*494.193*	*363.225*	*363.225*	*494.193*	*641.289*	*641.289*	1,993.544
A_{10}	1,696.771	1,283.868	1,283.868	1,283.868	1,283.868	1,283.868	1,283.868	792.256
A_{11}	1,045.159	1,161.288	1,161.288	1,161.288	1,161.288	1,161.288	1,161.288	2,477.414
A_{12}	641.289	*494.193*	*494.193*	*641.289*	*641.289*	*494.193*	*494.193*	1,890.319
W	1,970.142	1,903.3664	1,903.3664	1,903.3664	1,903.3664	1,903.3664	1,903.3664	3,782.8242
σ_{tmax}	178.924	179.906	179.897	179.783	179.962	179.987	179.963	169.468
σ_{cmin}	−153.109	−165.81	−165.143	−155.272	−150.432	−165.484	−165.848	−79.296

Benchmark 1: Wu and Chow (1995), benchmark 2: Lemonge (1999), and benchmark 3: Lemonge and Barbosa (2004). The cross-sectional areas of the horizontal members are printed in italic shape for the six variations of equal mass. For each solution the total mass, the maximum tension stresses as well as the maximum compression stresses are stated

Consideration of Euler buckling From an engineering point of view, it is absolutely mandatory to perform a buckling analysis, as the truss members are extremely slender. For this analysis the Euler buckling criterion according to (10) is applied, which reveals that none of the previously presented *optimum* truss structures is stable. Consequently, the following truss optimizations are extended by a Euler buckling constraint analogously to the examples of Sections 5.1 and 5.2 section. The additional Euler buckling constraint in combination with the discrete design variables extremely complicates the optimization task. A lot of individuals violating the buckling constraint are created by the operators, as choosing the next smaller cross-sectional area from Table 5 often leads to buckling designs. Therefore, 20 independent optimizations with population size 100 are run over 10,000 generations leading to the optimum design with a mass of 3,782.8242 kg; see Table 6. The solution fulfilling the Euler stability criteria is almost 100% heavier than the results of the previously presented optimizations.

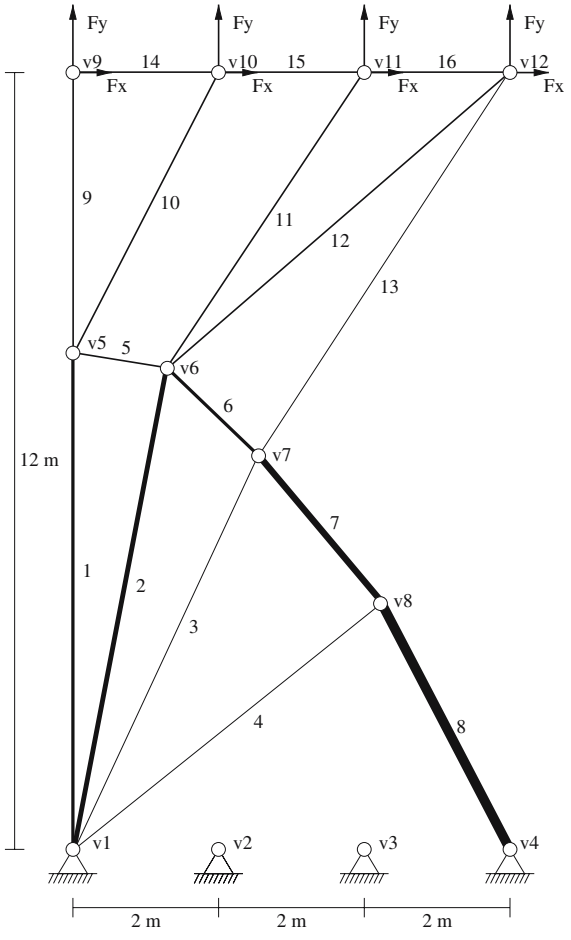

Fig. 10 Optimum result for the planar truss optimization. Final coordinates in mm are $v5$: (0.0/7,665.7), $v6$: (1,306.4/7,365.3), and $v7$: (2,502.4/6088.6), $v8$: (4,204.0/3,799.1)

Table 7 Results of the free optimization. The final mass is 1,097.7158 kg

Member	a	$\sigma_i(a_i)$	ms (10)
a_1	2,341.191	179.91837	–
a_2	3,703.218	175.91492	–
a_3	494.193	172.49449	–
a_4	494.193	176.52827	–
a_5	1,045.159	–	$4.36602 \cdot 10^{-2}$
a_6	2,896.768	–	0.12584
a_7	4,658.055	–	$1.88401 \cdot 10^{-2}$
a_8	6,999.986	–	$1.96118 \cdot 10^{-2}$
a_9	1,161.288	172.22227	–
a_{10}	1,283.868	171.56366	–
a_{11}	1,374.191	168.33515	–
a_{12}	1,374.191	179.91868	–
a_{13}	198.064	154.17152	–
a_{14}	1,690.319	–	0.16130
a_{15}	1,690.319	–	$7.80826 \cdot 10^{-2}$
a_{16}	1,535.481	–	$4.75843 \cdot 10^{-2}$

Free optimzation The planar 52-bar truss optimization problem is reformulated to demonstrate the full capacity of the graph representation. So far, the topology as well as the geometry of the 52-bar truss could not be changed and the design space was further reduced by the prescribed grouping of the member cross-sectional areas. Actually, these restrictions inhibit the optimizer from discovering innovative design solutions. Thus, the optimization task is formulated much freer, i.e., only the supported and the loaded nodes of the truss are given. All other nodes are not restricted to any location and can move through the design domain, i.e., the rectangle defined by the vertices $v1$, $v4$, $v12$, and $v9$ in Fig. 10. The maximum number of nodes is chosen to be 20 and the number of members must be between 10 and 60. Naturally, the maximum stress constraint and the buckling constraint have to be fulfilled, i.e., the optimization formulation is analogous to the definition in (8).

The complexity of this optimization task is extremely increased and, therefore, requires many more evaluations to converge to an optimum solution. First, the population size is set to 500 and the optimization is run over 10,000 generations. Afterwards, the best solution of the first run is taken as initial design for a second optimization run (population size 250 over 4,000 generations) with adjusted optimization parameters to emphasize the fine tuning of the structure. After six million evaluations, the solution depicted in Fig. 10 is found with a final mass of 1,097.7158 kg. The cross-sectional areas, the maximum tension stress values, and the margins of safety for buckling of each member are within the limits and can be found in Table 7. The resulting design consists of 16 members nine of which are tension members and seven are under compression, and none of the compression members reaches critical stresses close to the limit of 180 MPa. Only two of the four supported nodes are used and the members are quite logically arranged. In general, the compression members are kept rather short to fulfill the buckling constraint with relatively slender members. Furthermore, the sequence of the

compression members 5 to 8 approximates a parabolic shape that seems to be optimal to block the rotation of the structure around the node $v1$. The resulting design solution is approximately 71% lighter compared to the design solution with buckling constraint presented in Table 6. This convincing optimization result clearly shows that the graph representation of truss structures is able to fit sophisticated structures into given boundary conditions, although the computation costs are still rather high.

6 Conclusion

In the scope of this paper, a novel graph-based parameterization concept is introduced allowing for the optimization of truss structures with EA. The concurrent optimization of topology, geometry, and sizing is a central property of the presented parameterization approach. The optimization examples demonstrate the quality of the method leading to innovative topologies independent of any kind of ground structure. Basically, only the loaded and clamped nodes must be given, and the optimization method itself tries to fit an optimum topology with optimum geometry and sizing into the given design space.

Nevertheless, the concept needs to be further developed in terms of adaptivity and fine tuning. Practice has shown that the fitness function definition and single-gene properties need to be adjusted during the optimization process; otherwise, it is hardly possible to obtain fully stressed designs. A self-adaptive fitness definition will be implemented by incorporating the degree of constraint violation. If a constraint is not satisfied from any of the individuals, e.g., the stress constraint, the penalty value for this constraint will be increased to force the optimization to fulfill the respective constraint. In the same way, it seems to be reasonable to adjust (reduce) the standard deviation of the Gaussian mutation for nodal coordinates and member cross-sectional areas towards the end of the optimization to fine tune the near-optimal solution. This can be realized by analyzing the success of the Gaussian mutation operators and the location of improved solutions in the search space. Furthermore, the EA-based optimization algorithm should be extended with self-adaptive mechanisms adjusting the probabilities of application of the genetic operators during the optimization process. For example, genetic topology operators can hardly produce improved solutions when the optimization is almost converged to an optimum. Thus, the probability of the topology operators should be reduced to push the genetic geometry and sizing operators. This can be realized by analyzing the success of the genetic operators. The probability of application for successful operators should be moderately increased, whereas the probabilities of inefficient operators are reduced.

A further extension could be a hybrid optimization algorithm combining the traits of genetic search methods and mathematical programming to accelerate the optimization process providing the same result quality.

References

Azid IA, Kwan ASK, Seetharamu KN (2002) An evolutionary approach for layout optimization of three-dimensional truss. Struct Multidisc Optim 24(4):333–337

Balakrishnan R, Ranganathan K (2000) A textbook of graph theory. Springer, Berlin Heidelberg New York, pp 1–227

Bentley Peter J (ed) (1999) Evolutionary design by computers. Morgan Kaufmann, San Francisco California

Chen W-K (1997) Graph theory and its engineering applications. World Scientific, River Edge, New Jersey

Cheng G, Guo X (1997) ϵ-relaxed approach in structural topology optimization. Struct Multidisc Optim 13(4):258–266

Cuthill E, McKee J (1969) Reducing the bandwidth of sparse symmetric matrices. In: ACM Proceedings of the 24th National Conference. Association of Computing Machinery, New York

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

Foulds LR (1994) Graph theory applications. Springer, Berlin Heidelberg New York

Fowler PW, Guest SD (2000) A symmetry extension of maxwell's rule for rigidity of frames. Int J Solids Struct 37:1793–1804

Giger M, Ermanni P (2005) Development of cfrp racing motorcycle rims using a heuristic evolutionary algorithm approach. Struct Multidisc Optim 30:54–65

Gill PE, Murray W, Sanders MA (2002) SNOPT: an sqp algorithm for large-scale constrained optimization. SIAM J Optim 12(4):979–1006 (electronic)

Goldberg DE (1989) Genetic algorithms in search, optimization and machine learning. Addison-Wesley, Reading, Massachusetts

Gou X, Cheng G, Yamazaki K (2001) A new approach for the solution of singular optima in truss topology optimization with stress and local buckling constraints. Struct Multidisc Optim 22:364–372

Gou X, Liu W, Li H (2003) Simultaneous shape and topology optimization of truss under local and global stability constraints. Acta Mech Solida Sinica

Hajela P (1992) Stochastic search in structural optimization: genetic algorithms and simulated annealing. In: Structural optimization: status and promise. Prog Astronaut Aeronaut 150:611–637

Hajela P, Lee J (1995) Genetic algorithms in truss topological optimization. Int J Solids Struct 32(22):3341–3357

Holland JH (1975) Adaptation in natural and artificial systems. University of Michigan, Ann Arbor, MI

Kawamoto A, Bendsøe MP, Sigmund O (2004) Planar articulated mechanism design by graph theoretical enumeration. Struct Multidisc Optim

Kawamura H, Ohmori H, Kito N (2002) Truss topology optimization by a modified genetic algorithm. Struct Multidisc Optim 23:467–472

König O (2004) Evolutionary design optimization: tools and applications. PhD thesis, Swiss Federal Institute of Technology, Zürich, 2004. Diss. ETH No. 15486

Lemonge ACC (1999) Application of genetic algorithms in structural optimization problems. Ph.D. thesis, Federal University of Rio de Janeiro, Brazil, 1999. Program of Civil Engineering-COOPE

Lemonge ACC, Barbosa HJC (2004) An adaptive penalty scheme for genetic algorithms in structural optimization. Int J Numer Meth Engng

Lingyun W, Mei Z, Guangming W, Guang M (2004) Truss optimization on shape and sizing with frequency constraints based on genetic algorithm. Comput Mech 35(5):361–368

Pedersen NL, Nielsen AK (2003) Optimization of practical trusses with constraints on eigenfrequencies, displacements, stresses, and buckling. Struct Multidisc Optim 25(5–6):436–445

Pyrz M (2004) Evolutionary algorithm integrating stress heuristics for truss optimization. Opt Eng 5(1):45–57

Rozvany GIN (1997) Aims, scope basic concepts and methods of topology optimization. In: Rozvany GIN (ed) Topology optimization in structural mechanics, CISM courses and lectures, No. 374. Springer, Berlin Heidelberg New York, pp 1–55

Rozvany GIN (2001) Stress ratio and compliance based methods in topology optimization—a critical review. Struct Multidisc Optim 21:109–119

Rozvany GIN (2003) On design-dependent constraints and singular topologies. Struct Multidisc Optim 21:164–172

Ryoo J, Hajela P (2004) Handling variable string lengths in ga-based structural topology optimization. Struct Multidisc Optim 26:318–325

Siek JG, Lee LQ, Lumsdaine A (2002) The boost graph library. C++ in-depth series. Addison Wesley

Stolpe M, Svanberg K (2003) A note on stress-constrained truss topology optimization. Struct Multidisc Optim 25:62–64

Wang D, Zhang WH, Jiang JS (2002) Combined shape and sizing optimization of truss structures. Comput Mech 29:307–312

West DB (2001) Introduction to graph theory, 2nd edn. Prentice Hall, Englewood Cliffs, New Jersey

Wu S-J, Chow P-T (1995) Steady-state genetic algorithms for discrete optimization of trusses. Comput Struct 56(6):979–991

Xie YM, Steven GP (1997) Evolutionary structural optimization. Springer, Berlin Heidelberg New York