Strojniški vestnik - Journal of Mechanical Engineering 58(2012)3, 203-212
DOI:10.5545/sv-jme.2011.042

Paper received: 23.2.2011, paper accepted: 12.1.2012
© 2012 Journal of Mechanical Engineering. All rights reserved.

Synthesis of Truss Structure Designs by NSGA-II and NodeSort Algorithm

Tino Stanković* – Mario Štorga – Dorian Marjanović
University of Zagreb, Faculty of Mechanical Engineering and Naval Architecture, Croatia

This paper presents a genetic algorithm based approach for synthesis of truss structure designs. Genotype represented as a collection of binary encoded nodes is decoded into the phenotype by applying the NodeSort algorithm. A genotype extension to consider a cross-section as variable and variable length chromosomes to produce designs to successfully meet the boundary conditions are all being incorporated into the NodeSort to provide an efficient truss structures synthesis framework. The introduction of multi-objective optimisation using NSGA-II will help to address more real life engineering problems.

Keywords: truss structure design synthesis, genetic algorithms, NodeSort, NSGA-II

0 INTRODUCTION

The work presented in this paper proposes a genetic algorithm based approach for synthesis of truss structure designs. It builds on the *NodeSort* [1] algorithm, which takes a collection of binary encoded nodes from a 2D domain and decodes these into a truss structure in a process similar to FEM meshing. In our previous work it was shown that the *NodeSort* is well suited for the generation of optimal truss structure designs [1]. To continue the development of a truss structures synthesis framework, within this paper the following points are set as the research goals:

- To account for the multiple search objectives a non-dominated sorting genetic algorithm II (NSGA-II) [2] and [3] will be applied.
- Rather than being just a parameter, the cross-section area of trusses is defined as a variable what should improve the quality of produced solutions. Thus, the genotype is extended with one more binary encoded gene.
- To avoid the formation of node clusters occurring in near-optimal designs that had excessive number of predefined nodes, or in the opposite, to provide nodes to resolve a design solution when the course of evolution demands more trusses, a variable length chromosomes are introduced.
- To facilitate versatility and user-centricity, an arbitrary number of nodes with imposed loads are allowed to be specified.

The next section will address the applicability of evolutionary algorithms as powerful and apt optimizers. Then, the related work section will provide an overview of truss structures optimization approaches and drawn from these will justify the motivation for the NodeSort. To be able to accomplish the genotype extension and the introduction of variable length chromosomes, the adjustments need to be performed at the evolutionary operators' level. How are the crossover and mutation resolved to accommodate these changes, as well as the NodeSort algorithm, is presented in the third section. Section 4 gives FEM model of truss girder system and Section 5 formulates the objectives and constraints of multi-objective optimisation problem. Modifications of the search process parameters and penalty functions with the search algorithm given in pseudo-code are presented in Section 6. Case study and results discussion are presented afterwards. Conclusions and the future work notice close this paper.

1 EVOLUTIONARY ALGORITHMS

Evolutionary algorithms (EA) are population based stochastic optimisers that are built on mimicking the notions from the natural evolution. Charles Darwin stated that evolution begins with the inheritance of good gene variations and that basically defines what the evolutionary algorithms are all about. Enforcing the survival of the fittest principle is managed by allowing higher ranked solutions to participate in an evolution process by the most. In a random process of mixing together the building-blocks taken from two parent solutions an offspring is produced. Presumably, if building blocks originate from a higher fitness individuals than there is a chance that newly generated individual might get a bit closer to a solution optimum. With the whole process iterated over population generations to produce offspring which replace their parents at least to an extent, the emergence of solution occurs as a consequence of the most fit building-block combinations being frequently utilised in a solution construction. For such behaviour, it can be said that algorithm exhibited a learning process in identification of which building blocks to use and how [4] and [5].

The key issue of computational modelling [5] is to find out suitable problem representation acceptable both to the computational environment and to the

*Corr. Author's Address: Faculty of Mechanical Engineering and Naval Architecture, I. Lučića 5, Zagreb, Croatia, tino.stankovic@fsb.hr

task at hand. Thus, the problem representation has to concur to the algorithm's requirements and it should be devised in a fashion that captures the most of the form and attributes of the design search space. With EA's, the problem representation is most often brought down to finding out the appropriate genotype encoding and its counterpart, decoding into a phenotype, thus representing a key point in designing an evolutionary computation based optimization method [4] and [5]. The extension of the genotype to include truss thicknesses, the utilization of variable length chromosomes, as well as the NodeSort dependent decoding are viewed as part of an effort to provide a good material for the truss design evolution process. In that way by expanding the set of means to realize the required behaviour, the navigation to more applicable design solutions is made possible.

2 RELATED WORK AND PROBLEM FORMULATION

From the literature review, it can be found that the methods for the truss structure optimization are specialized either for the structure properties optimization [6] and [7] or the structure topology optimization [8] to [10]. By all means not diminishing the complexity of the problem, still the parametric optimisation genotypes are easily encoded because the layout of trusses is known and remains unchanged during the optimisation. As the initial variability of the shape is not being accounted for; instead the optimisation is carried over the usual truss design related variables, i.e. cross-sectional area, length, etc. By contrast, a different type of encoding is applied in the topological optimum design (TOD) cases [8] to [10]. The structure is represented in a discrete domain, most often in the form of material distribution, which is a straightforward approach in shape optimization. The genotype encodings may be accomplished in many ways, for example as matrices, Voronoi representations [9], or even by using 3-D FEM building blocks. The phenotype representation then visually depicts the resulting structure. An interesting cantilever optimization problem has been researched by Kim and de Weck [10], who addressed the quality of search with the domain resolution and chromosomal length [5]. They increased domain resolution gradually throughout the evolution to simulate the concretization process starting up from a vague concept as a small sized matrix to end up with a refined and concrete solution. TOD is computationally very demanding, it optimizes the structure in the form of the in-domain material distribution, but the other design variables, such as the cross-sectional area, remain predefined and out of reach.

A more subtle approach using the simulated annealing (SA) method together with shape grammars for the structural optimization purposes was developed by Shea and Cagan (1998) [11]. Shape grammars provide a formal language, a design language for the structure shape manipulation. Grammars are driven by a set of production rules, or simply productions, by which the solution is obtained from a series of transformations according to possible rules implementation sequences. However, for the description of all the possible truss structures, the set of rules must be equally large. Therefore, for a generic approach, to evolve the structures, the rules should be evolvable as well [12].

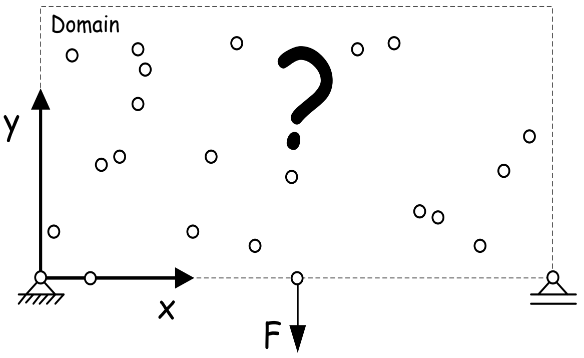

Fig. 1. *2-D continuous search domain [1]*

The problem of truss structure design in the 2-D continuous domain as seen within this work is pictured in Fig. 1. Assuming a random distribution of nodes and a number of predefined fixed nodes (supports and load nodes) all of which are contained within a chromosome, the topology of truss girder design emerges after the application of the NodeSort decoding algorithm [1] (Section 3.3 of this paper).

The genotype encoding and decoding applied in the NodeSort enable the search space to be as large and unconstrained as possible. It goes beyond parametric optimisation by allowing the optimisation of topology, it surpasses TOD approaches since it is much less computationally demandable with girder FEM's being parametric and optimised in a continuous domain. The phenotype represented as a truss structure emerging out of inter-nodal arrangement is an algorithmically driven approach. It offers an alternative to shape grammars which being knowledge-driven achieve the increase in the performance by addition of new rules, thus retaining the same rule transformation principles. Usually, there is a trade-off between the algorithmic and the knowledge-driven approaches since the former requires code interventions for the performance increase and the latter requires the shared understanding about the domain and objective rule definitions, otherwise formalized knowledge becomes inconsistent and biased. Extensions to our previous work reflecting the new encoding/decoding scheme are presented within the next section.

3 EXTENDED ENCODING/DECODING SCHEME FOR 2D CONTINUOUS DOMAINS

The model of truss structure is defined as the system comprised of a number of trusses and nodes. Boundary conditions specify the load cases, thus taking into account the amount of the loads being applied, as well as the type and the location of truss girder's supports. The load is modelled as a distribution of nodal forces. Within each of the evolution turns for every candidate solution the FEM system stiffness matrix is re-calculated. The position and length of the truss elements are defined as a consequence of the nodal arrangements within the 2-D domain. Assuming circular cross-section, an extension of previous work is the introduction of a truss cross-section diameter as a variable. Hence, a new structural model that is established with the addition of the cross-section diameter enables refinement in the search thus, allowing getting closer to the optimal system topology.

Not being explicitly written in the chromosomal genes, the trusses emerge as a consequence of in-plane nodal positions. Thus, in order to be able to introduce a truss cross-section dependant attributes it has been decided to extend the length of the binary encoded node by one gene to contain a gamete of a truss cross-section. The diameter of each truss is then obtained as an average calculated over two gametes belonging to the nodes which define the span of a truss.

According to [1], both fixed nodes and free nodes where predetermined in numbers. To clarify a distinction, in contrast to free nodes, the fixed nodes have a static position in the 2-D search domain thus not being subjected to a position change during the course of the evolution. However, because of the way the cross-section area is calculated, the new thickness gamete addition has to be taken into account even with the fixed nodes. The number of fixed nodes is denoted as *NoNx*. The improvement introduced within this work enables an increase or decrease of free nodes to occur as demanded by the course of evolution. For example, by prescribing the boundary conditions including the maximal allowed cross-section thickness, the search might be put in a dead end situation with no feasible results being generated unless it is allowed to increase the number *NoN* of free nodes. Hence, the increase in node numbers creates the possibility to generate more trusses since they appear as the result of nodal arrangements within the 2-D search domain. Consequently, a more stiffened and load capable structure will be designed. In contrast, by allowing the number of free nodes to be decreased is required in the case where the search has strayed in producing too complex solutions in respect to the defined loads. Then, it is necessary to reduce the number of trusses in order to obtain a best-fit solution.

3.1 Genotype Encoding

The genotype is assembled two folded; on the primary level each of the nodes is represented as a bit string, and the secondary level comprises of a collection of decoded nodes from which truss structure will be determined. For free nodes, three binary encoded genes are required, two to represent in-plain positioning of the node and one for the determination of the cross-section diameter (Table 1). Fixed nodes only comprise one gene to participate in the cross-section diameter calculation. Genotype encoding with chromosome represented as a collection of nodes is shown in Table 2. Parameter n_{max} represents the allowed number of nodes per chromosome.

Table 1. *Binary encoding of nodes*

Node$_i$		
x_i coordinate, binary string l_{i1}	y_i coordinate, binary string l_{i2}	d_i diameter gamete, binary string l_{i3}

Table 2. *Chromosome structure*

Chromosome$_j$	
Node$_1$, ..., *Node*$_i$, ...,	*Node$_n$* $n_j = NoNx + NoN_j \leq n_{max}$

Decoding is performed using standard decoding function γ [5]. Decoding per gene i for desired interval $[u_k, v_k] \in \mathbb{R}$ is defined by the following expression:

$$\gamma_k : \{0,1\}^{l_{ik}} \rightarrow [u_k, v_k]. \quad (1)$$

The complete decoding of each gene within a single free node i produces a vector of real numbers:

$$\gamma_i = \gamma_1 \times \gamma_2 \times \gamma_3 . \quad (2)$$

3.2 Evolutionary Operators

The mutation is performed simply, in a bit-flip manner, thus directly altering the nodal position or changing the cross-section diameter gamete in a random fashion. In order to improve the search process in the early stages of the solution evolution it was necessary to introduce scaling of the mutation rate. In the literature, there are a number of such mutation approaches [4] and [5]. They are driven by the notions from the natural evolution, justifying the initial mutation rate levels by the harsh environmental conditions. A bit-flip mutation rate is defined as a function of its initial rate $p'_m = 0.1$ and the number of evolution iterations N. The static mutation rate is being set to $p''_m = 0.02$. The bit-flip mutation probability is calculated by a linear scaling formula defined over a desired number of iterations:

$$p_m = \begin{cases} \dfrac{p''_m - p'_m}{1000} N + p'_m & N < 1000, \\ p''_m & otherwise. \end{cases} \quad (3)$$

Random addition or subtraction of nodes is performed by a node mutation procedure with the mutation probability taken the same as defined in Eq. (3). A simple random coin toss trial is performed to determine whether a point will be added or subtracted from the chromosome. Currently, only the free nodes are considered as legible for addition or subtraction. The following formalism defines the procedure for the node mutation of chromosome providing the maximal length n_{max} and the mutation triggering condition $p \leq p_m$:

$$\begin{cases} \text{add new randomly} & \\ \text{initilzed free node} & \text{if } head \wedge n_j < n_{max}, \\ \text{randomly delete} & \\ \text{a single free node} & \text{if } tails \wedge NoN_j - NoNx > 1. \end{cases} \quad (4)$$

Chromosome transcription during crossover acts to copy sequences of genetic material belonging to both parents in order to produce an offspring. However, to be able to perform the crossover on the chromosomal structure as shown in Tables 1 and 2, a slight adaption of the usual operator is required; namely to address all of the genetic material, a crossover has to be performed on both levels of the genotype. The crossover procedure is defined with the following:

- firstly, by randomly selecting one crossover point per each parent on the nodal levels, then,
- corresponding to selected nodes, the crossover points are randomly chosen again but on the bit-string level.

After the crossover it may turn out that the length of an offspring violates the condition $n_k > n_{max}$, as the result of an arbitrary nodal level crossover point selection. The formalism in Eq. (5) presents a way how to keep the size of the offspring k within the preset boundaries as defined by n_{max} :

$$\begin{cases} \text{Turn wise crossover point reposition :} & \\ parent_1 \text{ towards begining,} & \text{if } n_k > n_{max}, \\ parent_2 \text{ towards end.} & \\ \text{One point crossover :} & \\ \text{on free nodes,} & \text{if } n_k \leq n_{max}. \\ \text{on fixed nodes.} & \end{cases} \quad (5)$$

For the condition $n_k > n_{max}$, both chromosome segments are being shortened for one node in a turn wise manner until the condition $n_k = n_{max}$ has been met. The turn wise relocation of the crossover point tries to average the loss in diversity of genetic material to both parents. Afterwards, the standard one point crossover is performed separately on the collections of free nodes and fixed nodes. At the end, the nodes are unified into a single collection to form new offspring individual.

3.3 Phenotype Generation with the NodeSort Algorithm

Performing an automated search for the optimal system topology which employs the FEM methods for the system behaviour evaluation requires a creation of structural stiffness matrices absent of any singularities. Although the singular solutions can be ranked as infeasible by the constraint handling, their frequent occurrences will slow down and complicate the search process. To overcome the latter, drawn up on the engineering practice it is known that arranging trusses to form triangular substructures is the least requirement for the avoidance of mechanical joints formation in truss structures. Thus, if all of the sub-structures are triangular, the search space is narrowed down only to computable non-singular solutions, which helps to boost the efficiency of the search. Therefore, the *NodeSort* employs a mapping to create phenotype out of the encoded chromosome resulting in a mesh composed of triangular schemes to satisfy topological stiffness requirement. This mapping is similar to the meshing techniques applied within various FEM methods. The *NodeSort* algorithm for generating a phenotype based on the nodal positions and truss thickness gametes is presented with the following pseudo-code:

```
1.  sort and collect nodes ascending over $x$, if equal
    compare over $y$,
2.  point to first node in chromosome $i \leftarrow 0$,
3.  while $i < (n_i - 2)$ do
    A.  if $node_i.y \geq node_{i+1}.y$
            foreach truss definition collect $node_{i+1}$,
            $node_{i+2}$ ,..., $node_{i+k}$ which are ordered
            ascending over $y$ do
            break the search if $node_{i+k+1}$ satisfies any
            of the following:
                a.  is not ordered ascending over $y$,
                b.  is the second node in ascending order
                    satisfying $node_{i+k+1}.y > node_i.y$ ,
                c.  $i+k+1 > n_i$.
        od
    B. else: similar as in A but collecting nodes in
       descending ordering over $y$ with two break criteria
       redefinitions: (a) is not ordered descending over
       $y$, and (b) is the second node in descending order
       satisfying $node.y < node_i.y$.
    C. define trusses between $node_i$ and all nodes
       identified in A and B.
    D. move to next node $i \leftarrow i + 1$
4.  od.
```

The *NodeSort* algorithm starts with all of the nodes being sorted ascending based on their x coordinate. The first node from chromosome is taken into consideration by setting the counter i to zero (see pseudo-code line 2). In the following *while-do* loop which is marked with 3, the algorithm will search for possible ways to define truss elements between the considered node i and all of the nodes with greater x coordinate. The resulting structure must be composed of triangular schemes with no trusses creating intersections. Inside the *while-do* loop two possibilities which are marked with letters A and B can appear: based on its position the first following node $i + 1$ can either be below or on equal height (A) or above the considered node i (B). Inside A all of the nodes will be ranked legible for truss definition if they do not violate the conditions a, b and c. The first condition (a), makes sure whether all of the following i+k+1 nodes are in an ascending order over y, the following (b) stops the search if the node is the second one above the node i and in the end (c) it is prevented for the counter to be larger the number of nodes n_i. The condition B takes into account situations opposite of A - the first following node being above node i thus collecting truss definition nodes in a descending order. Except for the step (c) which considers the counter exceeding the number of nodes, the break criteria (a) and (b) are redefined for B to consider descending order of nodes as shown in pseudo-code. Afterwards, the trusses are defined (C) over all possible attributes including truss cross-section diameter and the whole procedure is repeated for the following node (D). Fig. 2 pictures a truss structure phenotype generated by the *NodeSort* algorithm which corresponds to the nodal arrangement which has already been shown in Fig. 1:

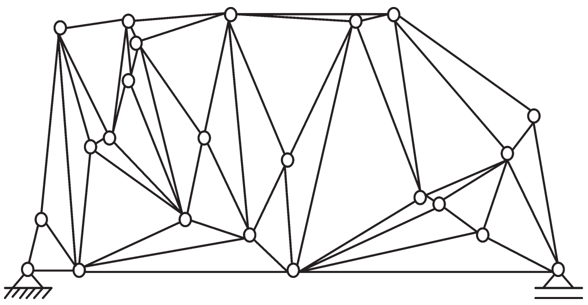
Fig. 2. *Truss structure obtained from nodal positions shown in Fig. 1 [1]*

The singular condition that may occur when two or more nodes are very close to each other or when they overlap is regulated with a constraint that proscribes the minimal allowed length of trusses:

$$l \geq l_{min}\,. \tag{6}$$

4 STRUCTURAL FEM MODEL

The structure is modelled using FEM planar trusses with possessing in total 6 degrees of freedom each. It was necessary to introduce bending to trusses and implicitly convert them into beams. Otherwise, the result of the evolution taking the infinite stiffness to bending of truss FEM element will always converge to a single horizontal truss. Such structure would have zero displacement since it cannot bend, it would be minimal in mass since it is just a horizontal line. Normal forces would also be equal to zero for the force vector put vertically as in Fig. 1. The relation between load F and the system's nodal displacement u is given here by with the following expression:

$$\{F\}=[\mathrm{K}]\{u\}. \tag{7}$$

The $[K]$ is a standard stiffness matrix defined for planar girder elements. The load vector considering the start and end points of each of the trusses is given in its transposed form as:

$$\{F^T\}=\{N_1, Q_1, M_1, N_2, Q_2, M_2\}. \tag{8}$$

The vector of displacements per truss element is given with the following expression:

$$\{u^T\}=\{u_1, w_1, \varphi_1, u_2, w_2, \varphi_2\}. \tag{9}$$

5 MULTI-OBJECTIVE OPTIMIZATION MODEL

The search goal is to find an optimal distribution of trusses that comprise topology of the structure in respect to minimal mass m and minimal deflection δ. The results will be obtained as a Pareto optimal front of truss structure designs. The optimisation problem is formulated with the following expression:

$$\left.\begin{array}{l} min\left[m\left(F,d_{max},E,n_{max},NoNx,m',BC\right)\right] \\ min\left[\delta\left(F,d_{max},E,n_{max},NoNx,m',BC\right)\right] \\ \textit{subject to:} \\ \delta \le \delta_{max} \\ l \ge l_{min} \\ n^t \le n_{min}^{t-1} \\ \begin{cases} \sigma \le \sigma_T & \sigma \ge 0 \\ |\sigma| < \sigma_B & \sigma < 0 \end{cases} \\ x \in \left[x_{min},x_{max}\right] = x_D \\ y \in \left[y_{min},y_{max}\right] = y_D \end{array}\right\} . \quad (10)$$

The overall mass of individual solutions m' is also being accounted for as an increase to the imposed loads. Empirically within this research it was determined that an addition of m' to the imposed loads pushes the search towards the optimum much earlier in the course of evolution.

The optimization parameters, variables and constraints are given as follows:

- F – load vector – fixed node/nodes only,
- d_{max} – allowed truss cross-section diameter,
- E – Young's modulus,
- n_{max} – allowed number of nodes,
- $NoNx$ – predefined number of fixed nodes,
- BC – boundary conditions – type of supports at particular fixed nodes, load distribution.

Problem variables:

- x and y coordinates of each node considered,
- d – truss cross-section diameter,
- l – length of respective truss,
- m' – mass of individual truss design added to the overall loads.

Constraints within the search domain are defined as follows:

- δ_{max} – allowable nodal deflection,
- l_{min} – minimally allowable truss length,
- n^t – dynamical and recursive constraint which proscribes that each population at step t is allowed to have the number of trusses less than or equal to the least number of trusses found in feasible solutions of the previous $t-1$ step,
- σ_T – allowable tensile stress in trusses, and
- σ_B – the Euler buckling stress for trusses.

It is assumed that compression stress state is calculated less than zero, and that the buckling strength will be entered as a parameter in its absolute value. Design search space is defined within 2-D bounding box.

6 CONTROL PARAMETERS, CONSTRAINT HANDLING AND ALGORITHM

Control parameters of the search algorithm for the evolution of truss girder designs are given as follows:

- population size: $\mu = 60$,
- offspring population size: $\mu = \lambda$,
- crossover probability for both genotype levels: $p_c = 1.0$,
- static mutation probability: $p''_m = 0.02$,
- search halt criteria: either user defined or predefined by $N \le 5000$,
- bit-strings lengths in genotype (Table 1): $l_{i1} = l_{i2} = l_{i3} = 9$.

Both feasible and unfeasible solutions enter a constrain-domination [2] and [3] process meaning that Pareto ranking is performed over all solutions to maintain diversity within the population. The feasible solutions are Pareto ranked over objective functions and the unfeasible ones are ranked according to their constraint violations. When comparing feasible and unfeasible solutions, the feasible always dominate the unfeasible ones. Constraint violation measure $\Omega_i(a_i)$ of ith solution $a_i \in P(t)$ from population P at iteration step t is derived as the summation of product between normalized violations $\omega_j(a_i)$ and the corresponding weighting factor R_j (R_j is normalized over the summation of all weighting factors). The expression for $\Omega_i(a_i)$ is given as follows:

$$\Omega_i\left(a_i\right) = \frac{\sum_{j=1}^{5} R_j \omega_j\left(a_i\right)}{\sum_{k=1}^{5} R_k}. \quad (11)$$

Normalizations and weights per constrain violation of ith solution are defined as follows:

$$\begin{cases} R_1 = 100.0 & \omega_1 = 1.0 & \delta_i > \delta_{max} \\ R_2 = 1000.0 & \omega_2 = 1.0 & l_i < l_{min} \\ R_3 = 1.0 & \omega_3 = n_i^t - n_{min}^{t-1} & n_i^t > n_{min}^{t-1} \\ R_4 = 1.0 & \omega_4 = \sigma_i / \sigma_T & \sigma_i > \sigma_T \\ R_5 = 1.0 & \omega_5 = \left|\sigma_i\right| / \sigma_B & \left|\sigma_i\right| > \sigma_B \end{cases} . \quad (12)$$

Violation of minimal length is regarded as a severe constraint violation since it may lead to overlapping of the nodes in the search domain thus resulting in the systems singular stiffens matrix. Secondly the violation of allowed deflection is also significant. These two constraints influence the initial search stages the most thus requiring the formation of so severe weighting factors. The other two constraints begin to significantly influence the search only in the later stages when most of the solutions being generated are meaningful solutions. Therefore, the weighting factor is set to 1.0 for the last three constraint violations. For the recursive constraint the normalized violations are calculated by subtracting the number of trusses of the considered solution and the least number of trusses found in feasible solutions of the previous iteration step. For the last two constraints the normalized violations are calculated by a division of current stresses found in trusses with the corresponding allowed stresses for tension and buckling.

6.1 The Search Algorithm

The pseudo-code of the search algorithm is given in concordance with the general model of evolutionary algorithm [5]:

```
1.  $t \leftarrow 0$;
2.  $P(t) \leftarrow init(\mu, n_{max}, NoNx, BC)$;
3.  $F(t) \leftarrow eval(P(t), \mu, F, d_{max}, E, m', x_D, y_D, \theta_c)$ do;
    A. $decoding\_y(d_{max}, x_D, y_D)$;
    B. $check\_points(P(t))$;
    C. $NodeSort((P(t))$;
    D. $[K] \leftarrow create\_[K](P(t), E)$
    E. $\{F\} \leftarrow create\_\{F\}(P(t), F, m')$;
    F. $\{u\} \leftarrow calculate\_\{u\}((P(t), [K], \{F\})$;
    G. $\{\sigma\} \leftarrow calculate\_\{\sigma\}((P(t), \{u\})$;
    H. $apply\_\Omega((P(t), \theta_c)$;
od
4.  while $(t < n)$ do
    a. $P'(t) \leftarrow cross(P(t), F, \lambda, p_c, n_{max}, NoNx, BC)$;
    b. $P''(t) \leftarrow bit\text{-}flip(P'(t), \lambda, p_m)$;
    c. $P'''(t) \leftarrow node\_mut(P''(t), \lambda, p_m, n_{max}, NoNx)$;
    d. $F(t) \leftarrow eval(P'''(t), \lambda, F, d_{max}, E, m', x_D, y_D, \theta_c)$
       do... od
    e. $P(t+1) \leftarrow NSGA\_II(P(t) + P'''(t), \mu)$;
    f. $t \leftarrow t + 1$;
od
```

A random creation of the initial population composed of μ chromosomes refers to step 2 of the pseudo-code. Populations of free and fixed nodes are created separately and then joined together. The evaluation considers decoding (*A*) form integer to real values (as given by Eq. (1)), displacing overlapping nodes (*B*), applying *NodeSort* (see Section 3.3), calculating displacements and stresses in trusses (steps *D*-*G*) as explained in Section 4. Step *H* concludes the evaluation by applying a constraint check using the Eqs. (11) and (12) to obtain constraint violation measure. For the sake of convenience all the relevant constraint parameters are denoted as θ_c. Step 4 onwards denotes the iterative *while-do* loop which lasts until the halting condition is satisfied. The crossover in step *a*. which produces offspring population of size λ is defined as given by the Eq. (5). To stress out the difference in respect to the parent population, the offspring population generated at step *a* is denoted with $P'(t)$. The mutation of offspring's involving steps *b* and *c* is defined by expressions for mutation probability calculation (Eq. (3)) for bit-flip mutation to form $P''(t)$ and nodal mutation (Eq. (4)) to form $P'''(t)$, respectively. The evaluation procedure at step *d* is comprised of the same subroutines as in the initial in steps *D*-*G*. The difference is that the evaluation is being applied to offspring population $P'''(t)$. Finally, the NSGA-II (*e*) creates a new population of size μ involving the Pareto based ranking.

7 TEST EXAMPLE

Test example involves a multi-objective optimisation case with the boundary conditions selected as shown in Fig. 1. The input parameters are: load F = 2 t (~20 kN), maximal truss thickness d_{max} = 50 mm, Young's modulus of elasticity for steel E = 210 MPa, maximal number of nodes involved n_{max} = 13, number of fixed nodes $NoNx$ = 3. In respect to formalism in Eq. (11) the optimisation problem is given by the following Eq. 13:

$$\left.\begin{array}{l} min\left[m\left(F, d_{max}, E, n_{max}, NoNx, m', BC\right)\right] \\ min\left[\delta\left(F, d_{max}, E, n_{max}, NoNx, m', BC\right)\right] \\ \text{subject to}: \\ \delta \leq 0.015\,\text{m} \\ l \geq 0.250\,\text{m} \\ n^t \leq n_{min}^{t-1} \\ \begin{cases} \sigma \leq 100.0\ \text{N/mm}^2 & \sigma \geq 0 \\ |\sigma| < \sigma_B & \sigma < 0 \end{cases} \\ x \in [0, 15.0\ \text{m}] \\ y \in [0, 7.5\ \text{m}] \end{array}\right\}. \quad (13)$$

8 DISCUSSION OF RESULTS

The scatter plot $m - \delta$ (Fig. 3) shows the the Pareto optimal front formation during 5000 iterations recorded for every 100th step of truss design evolution. The distinctive points on the Pareto curve which define the span (designs 1 and 3) are shown in Figs. 4 and 5, and the knee solution (design 2) is shown in Fig. 6. Crossed-out points within the scatter plot denote the infeasible solutions which violate the constraints according to Eqs. (11) to (13) taken without $n_{t-1} \leq n_t$. Corresponding to labelling defined in the scatter plot $m - \delta$ (Fig. 3) and Figs. 4 to 6 the truss cross-section diameters and positions of nodes for designs 1, 2 and 3 of non-dominated set are shown in Tables 3 and 4.

The optimisation of the averages of objectives with standard deviation calculated at generation 5,000 during 10,000 evolution runs for distinctive points on the Pareto curve are shown in Table 5.

The algorithm score presented in Table 5 states high repeatability of the results in respect to the overall objectives. The only significant dispersions are noted over δ for Design@1 and Design@3. However, a 1/10 and 3/10 of a millimetre are more than acceptable for the proposed search domain as defined in Eq. (13). The repeatability achieved over the design topology is shown in Fig. 7. The picture shows the spread of free nodes in knee solutions recorded for 5,000th iteration through 10,000 evolution runs.

Table 3. *Truss cross-section diameters*

Compression	*d* [mm]				
Design@	1	2	3	4	
1	45.4	43.5	42.0	44.0	
2	45.0	47.5	46.7	44.0	
3	50.0	50.0	49.9	50.0	
Tension	*d* [mm]				
Design@	5	6	7	8	9
1	22.4	24.2	20.3	22.7	22.5
2	29.1	33.9	31.6	33.1	28.9
3	49.9	49.9	49.9	49.9	49.5

Table 4. *Nodal positions*

	Free nodes starting with top left node [m]					
Design@	*x*	*y*	*x*	*y*	*x*	*y*
1	3.4	4.2	7.8	5.2	12.0	4.0
2	2.4	4.9	7.5	6.6	12.6	4.6
3	2.6	5.6	7.6	7.5	12.4	5.6
	Fixed nodes starting with bottom left node [m]					
1, 2, 3	0.0	0.0	7.5	0.0	15.0	0.0

Table 5. *Averages of objectives with standard deviations*

Design@	*m* [t]		δ [mm]	
	$\overline{m}$	S_N	$\overline{\delta}$	S_N
1	0.33	0.01	2.33	0.34
2	0.48	0.03	1.26	0.11
3	0.88	0.04	0.83	0.05

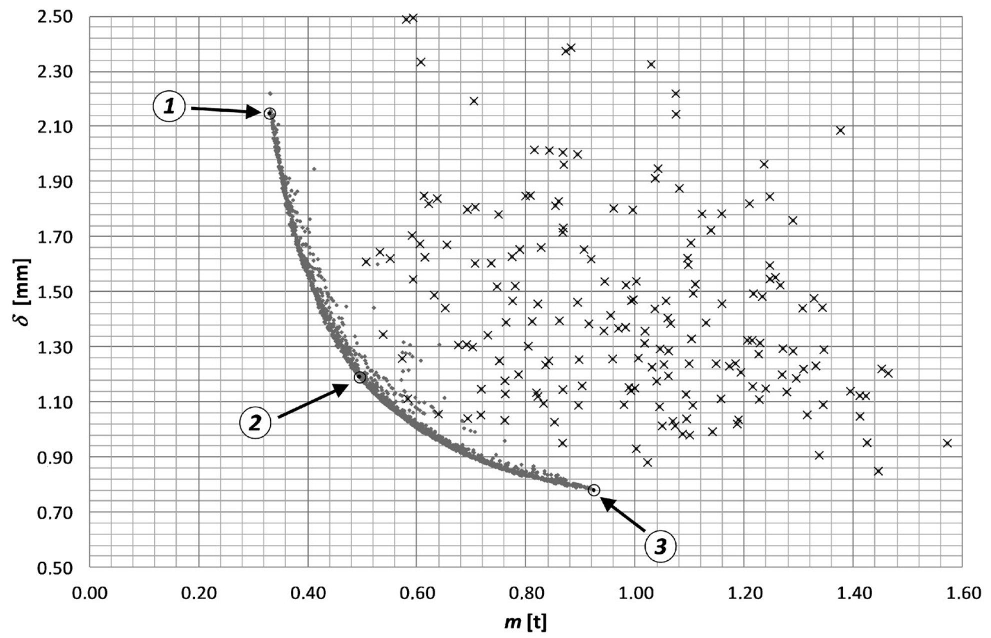

Fig. 3. *Scatter plot m – δ showing the Pareto optimal front formation recorded during 5000 iterations*

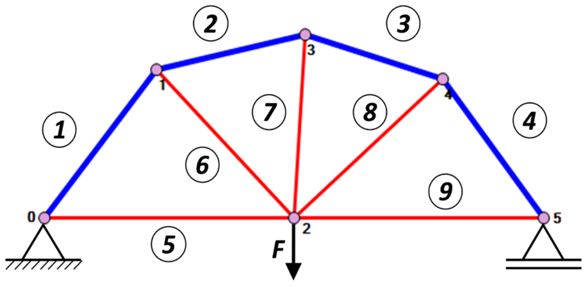

Fig. 4. *Design@1 (lightweight design), m = 0.33 t, δ = 2.15 mm*

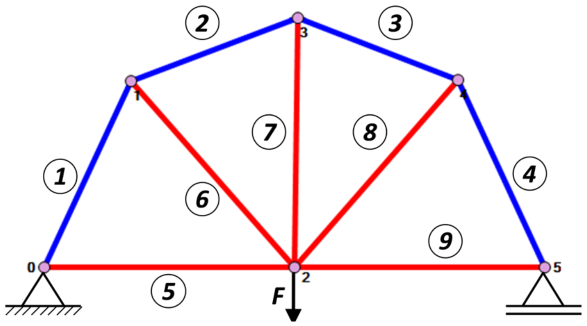

Fig. 5. *Design@3 (heaviest design), m = 0.92 t, δ = 0.78 mm*

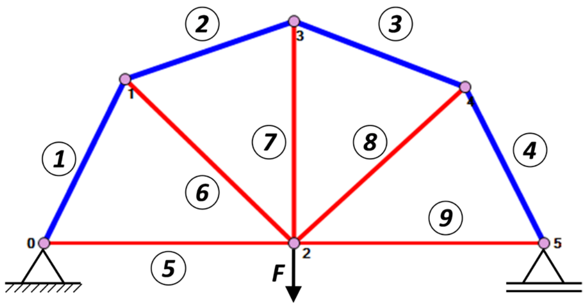

Fig. 6. *Design@2 (knee solution), m = 0.49 t, δ = 1.19 mm*

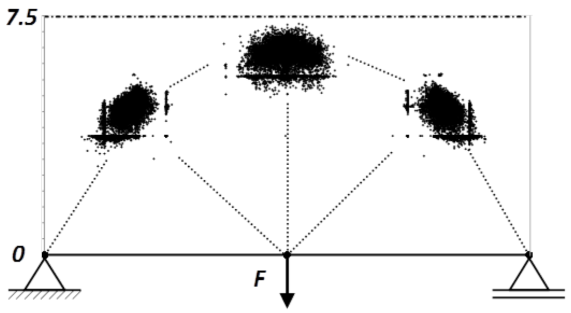

Fig. 7. *Spread of free nodes recorded for the knee solution(Design@2 type)*

9 CONCLUSION AND FURTHER WORK

The test example verified that it is possible to utilize the *NodeSort* phenotype based decoding together with the proposed genotype extensions to include thickness gamete and variable length chromosomes to achieve a multi-objective NSGA-II backed optimisation. By supporting a complete topological search which maintains high results repeatability, an edge over presented methods has been achieved. Further work will address the influences on the optimal solution search in respect to recursive constrain $n^t \leq n_{min}^{t-1}$ and the order of node sorting.

10 REFERENCES

[1] Stanković, T., Marjanović, D., Bojčetić, N., Ščap, D. (2009). Enhancing Evolution of truss structures by using genetic algorithms. *Transactions of FAMENA*, vol. 33, no. 11, p. 1-10.

[2] Deb, K., Agrawal, S., Pratap, A., Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm NSGA-II. *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, p. 182-197, DOI:10.1109/4235.996017.

[3] Deb, K., Agrawal, S., Pratap, A., Meyarivan, T. (2000). A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. *Parallel Problem Solving from Nature VI Conference*.

[4] Goldberg, D.E. (2002). *Design of Innovatinon*. Kluwer Academic Publishers, Norwell.

[5] Bäck, T., Fogel, D.B., Michalewicz, Z. (2000). *Evolutionary Computation, Advanced Algorithms and Operators*. Institute of Physics Publishing, Bristol.

[6] Hasançeb, Q. (2007). Optimization of truss bridges within a specified design domain using evolution strategies. *Engineering Optimization*, vol. 39, no. 6, p. 737-756, DOI:10.1080/03052150701335071.

[7] Coello, C.A.C., Christiansen, A.D. (2000). Multiobjective optimization of trusses using genetic algorithms. *Computers and Structures*, vol. 75, no. 6, p. 647-660, DOI:10.1016/S0045-7949(99)00110-8.

[8] Jakiela, M.J., Chapman, C., Duda, J., Adewuya, A., Saitou, K. (2000). Continuum structural topology design with genetic algorithms. *Computational Methods in Applied Mechanical Engineering*, vol. 186, no. 2, p. 339-356, DOI:10.1016/S0045-7825(99)00390-4.

[9] Hamda, H., Schoenauer, M. (2002). Topological optimum design with evolutionary algorithms. *Journal of Convex Analysis*, vol. 9, no. 2, p. 503-517.

[10] Kim, Y.I., De Weck, O. (2004). Progressive structural topology optimization by variable chromosome length genetic algorithm. *China-Japan-Korea Joint Symposium on Optimization of Structural and Mechanical Systems, M3*, Kanazawa.

[11] Shea, K., Cagan, J. (1998). Generating Structural Essays from Languages of Discrete Structures. Hero, j.s., Sudweeks, F. (Eds.) *Artificial Intelligence in Design,* Klower Academic Publishers, Dordrecht, p. 365-384.

[12] Gero, J.S., Louis, S.L. (1995). Improving pareto optimal designs using genetic algorithms, *Microcomputers in Civil Engineering*, vol. 10, no. 4, p. 241-249, DOI:10.1111/j.1467-8667.1995.tb00286.x.