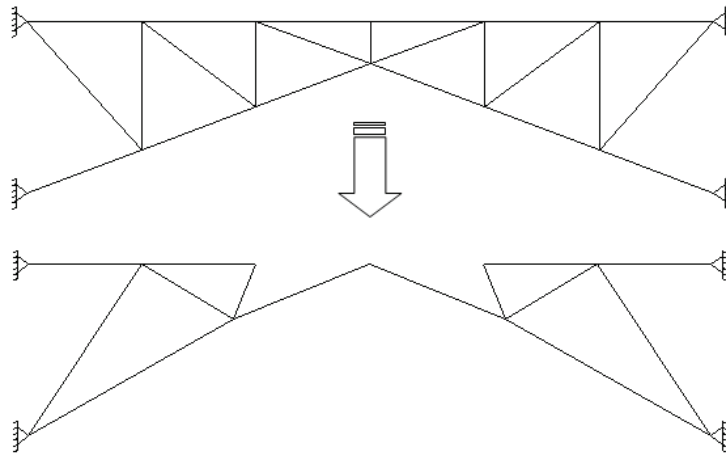


Weight optimization of steel trusses by a genetic algorithm

-Size, shape and topology optimization according to Eurocode



Max Hultman

Department of Structural Engineering
Lund Institute of Technology
Lund University, 2010

Department of Structural Engineering
Lund Institute of Technology
Box 118
S-221 00 LUND
Sweden

Avdelningen för Konstruktionsteknik
Lunds Tekniska Högskola
Box 118
221 00 LUND

Weight optimization of steel trusses by a genetic algorithm – Size, shape and topology optimization according to Eurocode

Viktoptimering av stålfackverk med en genetisk algoritm
– Dimensions-, form- och topologioptimering enligt Eurokod

Max Hultman

2010

Rapport TVBK-5176
ISSN 0349-4969
ISRN: LUTVDG/TVBK-10/5176+65p

Master's thesis
Supervisor: Daniel Honfi, Ph.D. student at Department of Structural Engineering
February 2010

Abstract

There are benefits of weight optimized structures in many engineering fields. In civil engineering it can for instance be associated with cheaper structural parts and easier transportation. In this study a genetic optimization algorithm for weight minimization of steel trusses has been developed in MATLAB. Constraints regarding material strength and buckling stability are taken from “Eurocode 3: Design of steel structures” and implemented in the algorithm.

Genetic programming is an effective search technique based on natural selection. The basic idea is to combine good solutions to a certain problem over many generations to gradually improve the result. All solutions are initially created randomly, and they are individually represented by a binary string with some similarities to natural chromosomes, hence the name *genetic* programming.

As contrary to most previous studies on the subject, a simultaneous optimization of size, shape and topology has been performed. This means a huge amount of computer effort and time is needed to solve a problem, due to the enormous search space that arises. To deal with this, a *reduced method for topology optimization* is proposed in order to reduce the complexity of a problem. The reduced method should be generally preferred over the widely used *ground structure method* for structures of 6-64 nodes. The importance of minimizing the number of possibilities is also emphasized in order to generate reasonable optimization problems.

Three examples run on a 2GHz PC are presented in this study; two versions of a cantilever truss, commonly known as the benchmark problem, and a 24-meter bridge truss. The result of the benchmark problem with 50.8 mm (2 in) displacement limit was 2327 kg (5130 lb) and there are reasons to believe this is a close to optimal solution. The two other examples are slightly more extensive, and the results from those examples show some margins, indicating non-optimal solutions. This is most certainly a result of insufficient computer power. This type of optimization should probably be carried out using parallel computing if the problem isn't very simple.

1	Introduction	1
1.1	Background.....	1
1.2	Previous work	1
1.3	Aim	2
1.4	Hypothesis	2
1.5	Disposition.....	2
2	Theory	3
2.1	Trusses.....	3
2.2	Structural optimization	3
2.3	Genetic algorithms.....	5
2.3.1	A glance back	5
2.3.2	The GA principle.....	5
2.3.3	Representation.....	6
2.3.4	Fitness evaluation.....	6
2.3.5	Selection	6
2.3.6	Cross-over	7
2.3.6.1	Mutation.....	7
2.3.6.2	Elitism	7
2.3.7	The schemata.....	8
2.3.8	Population size	8
2.3.9	Genetic Algorithms in MATLAB	8
2.4	Axially loaded bars according to Eurocode 3 [9].....	9
2.4.1	Tension.....	9
2.4.2	Compression.....	10
2.4.3	Cross-section classification	10
2.4.4	Buckling resistance of steel bars [9]	11
2.4.4.1	Buckling reduction factor, χ [9].....	12
2.4.4.2	Effective cross-sectional area, A_{eff} [10].....	12
2.4.4.3	Critical load, N_{cr}	13
3	Proposed algorithm	15
3.1	General.....	15
3.2	Constraints	15
3.2.1	Constraint 1: Fabricational	15
3.2.2	Constraint 2: Basic nodes	15
3.2.3	Constraint 3: Stability.....	15
3.2.4	Constraint 4: Nodal displacements.....	15
3.2.5	Constraint 5: Constructability	16
3.3	Constraint management	16
3.3.1	Topology optimization	17
3.3.1.1	Zero-bars	17
3.4	Reduced method for topology optimization	18
3.4.1	Basic structure and “free” elements	19
3.4.2	Advantages with the reduced method	23
3.4.3	Limitations	24
3.5	Effects of simultaneous size- shape- and topology optimization	28
4	Results	29
4.1.1	Benchmark problem	29
4.1.1.1	Same truss, tougher limits on displacement.....	32
4.1.2	A bridge truss	34

5	Conclusions	37
5.1	Proposal for further work	37
6	References	39
7	Appendices	43
A)	Optimization algorithm m-files	43
B)	List of available profiles [28]	57

1 Introduction

1.1 Background

According to Coello Coello et al. [6], Galileo Galilei seems to be the first scientist to study structural optimization in his work on the bending of beams. This discipline has evolved over time to become an engineering area called structural optimization. The increasing interest in this area the last few decades is due to the availability of cheap and powerful computers, along with rapid developments in methods of structural analysis and optimization [27]. Weight optimization of structures plays a major role in many engineering fields. In some aspects it can be associated with cost optimization, since it obviously leads to an optimal material usage. In civil engineering, weight optimized structures are convenient since the transportation and construction work in connection with the buildup is simplified. Another advantage of having a weight optimized structure is that a minimum share of the load capacity is engaged by the structure itself. Structural optimization is also important in the aircraft and car industry where a lighter structure means a better fuel economy.

An efficient optimization technique is the use of genetic algorithms. GA, as it is most commonly referred to, is a type of evolutionary programming¹ and probably the best-known today [2]. It simulates the evolutionary principle of survival of the fittest by combining the best solutions to a problem in many generations to gradually improve the result. The initial population of solutions is created randomly, and as the evolution goes, the best individuals are combined in each generation until an optimal solution converges [13].

1.2 Previous work

The use of genetic algorithms in search for an optimal design of trusses has been described in several scientific reports over the last two decades. In most of these studies though, the optimization doesn't refer to size, shape and topology simultaneously. Most commonly the topology of the truss (i.e. the inner connectivity of the members) is fixed, e.g. Kaveh & Kalatjari (2004) [17]; Guerlement et al. (2001) [14]; Soh & Yang (1998) [27]; Galante (1996) [11]; and Rudnik et al. (1994) [6].

One common way of dealing with truss topology optimization is the *ground structure method*, used for instance by Hajela & Lee (1994) [15] and Deb & Gulati (2001) [8]. In this method, a highly connected ground structure with many nodes and elements is slowly reduced till only the necessary elements are remaining [8, 23].

¹Evolutionary programming or *Genetic Programming* is not the only search method that mimics the nature in some way; actually there is quite a few, each one with more or less far-fetched analogy with nature. To mention some, there are *Ant Colony Optimization*, *Particle Swarm Optimization* and *Simulated Annealing*. The common feature of all these methods is that they all start out with randomly created solutions that are later developed until an optimal solution converges.

Ant Colony Optimization imitates the foraging behavior of ants and the way in which they end up finding the shortest way between a food source and the ant hill through the use of pheromone traces [15].

Particle Swarm Optimization is a search method inspired by swarming animals, like fish or birds. Potential solutions are represented by randomly initialized particles in the search space, and as the calculations proceeds, these particles will "swarm" towards the best solution [21].

Simulated Annealing is inspired by statistical thermo dynamics and is developed to simulate the behavior of atoms and molecules during the annealing process. As the temperature slowly decreases, the material is finding its lowest energy level through atomic rearrangement. Through several repetitions of simulated annealing, the solution converges, and an optimal structure is obtained [20]. The reader is referred to the references for further explanations.

In some of the latter studies on truss optimization with GA, the focus has been on developing a highly efficient genetic algorithm that finds an optimal solution by as few calculations as possible, e.g. the adaptive approach, presented by Togan & Daloglu 2006 [29]; or the directed mutation, presented by Li & Ye 2006 [22]. Nor in these studies, size, shape and topology have been taken into account simultaneously.

1.3 Aim

The aim of this work is to develop a genetic algorithm that optimizes planar steel trusses with respect to minimum weight. The optimization refers to the three design categories; size, shape and topology. The requirement is that the algorithm only proposes trusses that consists of elements taken from an available profiles list, and that it satisfies the relevant constraints given in Eurocode 3: Design of steel structures. As to the author's knowledge, no similar work has been done previously.

1.4 Hypothesis

The proposed algorithm will return optimal or near optimal solutions to almost any given weight optimization problem concerning steel trusses, but it may demand more time and calculations to get there than earlier presented *improved* genetic algorithms, due to the fact that the focus is on developing a well functioning complete optimizing algorithm instead of an efficient one.

1.5 Disposition

This work is divided into seven chapters, beginning with an introduction in the first one above. This is followed by a presentation of the theory in the second chapter. That is, a brief presentation of what trusses are, an introduction to structural optimization, and a description of what genetic algorithms are and how they work. The second chapter also contains the relevant rules from Eurocode 3: Design of steel structures. This section and its subsections contain many formulas that might be complicated to follow. The formulas are only presented to show what it is that goes into the algorithm, and they don't need any deeper examination.

The third chapter holds a description of the proposed algorithm, along with an examination of a new method for topology optimization that is developed in connection with this work. The third chapter ends with a short discussion on the effects of optimizing size, shape and topology simultaneously.

The fourth chapter contains results from structures that have been optimized with the proposed algorithm. The obtained results are also compared with previously presented results on the very same structures.

The fifth chapter holds the conclusions on this work, along with a proposal for further work. The discussion is also found in this chapter, even if chapter four also holds some discussion. All the references are listed alphabetically in the sixth chapter.

The seventh and last chapter holds the appendices. The proposed algorithm m-files are listed in appendix A), and the used list of steel profiles given in appendix B).

2 Theory

2.1 Trusses

A truss is a structure of assembled bars, often arranged in a triangular shape. Theoretically, the bars in a truss are assumed to be connected to each other by friction-free joints. In real-life trusses though, the joints are more or less stiff due to welding or screwing the bars together. Even with some stiffness in the connections, a model with friction-free joints can accurately be used if the centre of gravity axis of each bar meets in the point where you put the joint in the model [16]; see figure 1:

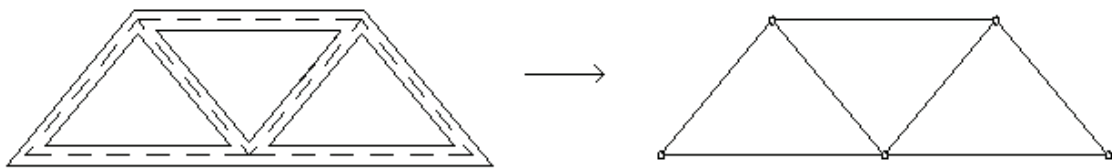


Figure 1: A truss structure with its corresponding theoretical model. The circles that denoted the nodes are commonly left out.

As long as the load is applied in some of the nodes, the bars will only be subjected to compressive or tensile normal forces. This is one part of the explanation to why trusses are so light compared to their load capacity; bar effect is more efficient than beam effect [16]. The other part is that the triangle is the simplest *stable* structure that extends in two dimensions [30].

Due to their efficiency, trusses are desirable in long span structures with high demands in stiffness and strength [16]. Typical scopes of uses are bridges, long-span roof structures and transmission towers. Some well known examples of truss structures are the Eiffel tower in Paris, the Harbor Bridge in Sydney and the Oresund Bridge (a cable-stayed truss bridge) between Copenhagen and Malmoe.

2.2 Structural optimization

The term *optimal structure* is very vague. This is because a structure can be optimal in different aspects. These different aspects are called objectives, and may for instance be the weight, cost or stiffness of the structure. A numerical evaluation of a certain objective is possible through an *objective function*, f , which determines the goodness of the structure in terms of weight, cost or stiffness [4]. Of course, the optimization has to be done within some constraints; otherwise it's a problem without a well defined solution [4]. Firstly, there are *design constraints*, like a limited geometrical extension or limited availability of different structural parts. Secondly, there are *behavioral constraints* [4] on the structure that denotes the structural response under a certain load condition. Here may, for instance, limits on displacements, stresses, forces and dynamic response be sorted. Finally, there is one obvious demand that is valid for all structures, and it is *kinematical stability*, otherwise they are mechanisms [30]. This can be seen as a behavioral constraint. Structures that lie within the constraints are called *feasible* solutions to the optimization problem.

A general expression for structural optimization is given for instance by Christensen & Klarbring (2008) [4]:

$$(SO) \begin{cases} \text{minimize } f(x, y) \text{ with respect to } x \text{ and } y \\ \text{subject to } \begin{cases} \text{behavioral constrains on } y \\ \text{design constrains on } x \\ \text{stability constraint} \end{cases} \end{cases}$$

where f is the objective function;
 x is a function or vector representing the design variables, and;
 y is a function or vector representing the state variables, i.e. the response of the structure.

Optimization can be done with respect to two or more different objective functions. This is referred to as multi-objective optimization [5] (also called multi-criterion or vector optimization [4, 5]). One example of this is Galante's (1996) [11] attempt to find a minimal weight of a truss using as few different profiles as possible. In multi-objective optimization, one general objective function can be put together by weighted parts of the involved objective functions. Hence, by changing the weights, different optima are obtained [4]. Other methods for dealing with multi-objective optimization are also possible.

When it comes to trusses, the optimization can be divided into three categories; sizing, shape and topology optimization;

Sizing optimization refers to finding the optimal cross section area of each member of the structure; *shape optimization* means optimizing the outer shape of the structure; and *topology optimization* describes the search for the best inner connectivity of the members [4].

One way of optimizing these three parameters is to take them into consideration one at a time, starting with the topology optimization, a so called multi-level optimization technique (also called layered optimization [18]). It is obvious though, that this approach doesn't always provide the best global solution, since the problems aren't linearly separable [8]. One of the strengths of a genetic algorithm is that a simultaneous optimization of all three parameters can be done, see section 2.3.

2.3 Genetic algorithms

2.3.1 A glance back

Professor John Holland is commonly known as the father of the genetic algorithm technique. He consolidated the technique in his book *Adaptation of Natural and Artificial Systems* in 1975 [7, 13, 24]. At this time though, the idea of mimicking the evolution in programming had been around for a while. In Germany, for instance, Ingo Rechenberg and Hans-Paul Schwefel developed the *Evolutionsstrategie* (eng. Evolution Strategy) in the 1960s. At the same time, similar work was conducted in the USA under the name *Genetic Programming*. These early proposals involved mutation and selection, but not recombination, which is the key feature of GAs by Professor Holland [24]. Even though this new technique gave some promising results, it didn't gain much interest at the time, probably due to the lack of computational power [24]. GAs fell more or less into oblivion for the next ten years till 1985, when the first international conference on GA was held. Up until then the technique had mainly been used by Professor Holland and his students [7]. The conference shed new light on genetic programming, and with 32 times more powerful computers than in 1975, it got a warmer welcome. Over the next decade the number of scientific publications on genetic algorithms grew at approximately 40 % each year till 1995 when it peaked [1]. The main part of these publications was different implementations of GAs [1].

When it comes to structural optimization, David E. Goldberg (an inquiring student of Professor Holland's [13]) seems to be the first one to suggest the use of GAs [5, 6, 17]. In 1986, he and a graduate student of his used the GA technique to minimize the weight of a ten-bar aluminum truss [13]. This structure is commonly used as a benchmark problem in structural optimization and in section 4.1.1, a steel version of it is being optimized according to Eurocode.

2.3.2 The GA principle

Genetic Algorithms have three characteristic operators, namely *selection*, *crossover* and *mutation*. In each iteration, or *generation*, these operators are applied on a population of possible solutions, or *individuals* in order to improve their *fitness*. Each individual is represented by a *string*, and as we will see, these strings remind very much of the natural chromosomes, hence the name genetic algorithms [7]. Initially, the population is created randomly, and the breeding continues until a stopping criterion is reached, e.g. the exceeding of a certain number of generations, or the absence of further improvements among the individuals. In the following sections, a more detailed review of the different GA operators is given.

There are many advantages with the GA technique, primarily its simplicity and broad applicability. It can easily be modified to work on a wide range of problems [26], as contrary to traditional search methods that are specified on a certain type of problem [7]. The technique is relatively robust as well; it does not tend to get stuck in local optimums as other techniques may do [7, 26]. Furthermore, due to the use of function evaluations rather than derivatives, it can handle discrete variables and is able to work in highly complex search spaces [26]. On the negative side, it may require many function evaluations, and it sometimes suffers from premature convergence; the individuals get very similar to each other early in the process [26]. Finally, as we shall see, there are many different options and alternatives in Genetic Algorithms, and it may be hard to find the right settings to achieve high efficiency.

2.3.3 Representation

Just like the chromosome, the string has different segments, or *genes*, that correspond to different features of the solution [24]. In biological terms, the total information stored in a string is called the *genotype* of an individual, the genetic information. The outward appearance of an individual is called the *phenotype*, and between the two a transformation exist; a *genotype-phenotype mapping* [24, 25].

In the traditional GA, the string consists of a fixed-length binary string [26]. The number of genes is dependent on the number of variables that need representation. For instance, the following string consists of five genes, g_1 - g_5 , representing five problem variables:

$$\underbrace{11010110010101}_{g_1} \underbrace{11000100110001}_{g_2} \underbrace{11000100110001}_{g_3} \underbrace{11000100110001}_{g_4} \underbrace{11000100110001}_{g_5}$$

The number of bits required in a certain gene is calculated as [7]:

$$l_i = \log_2 \left(\frac{x_i^{\min} - x_i^{\max}}{\varepsilon_i} \right)$$

where x_i^{\min} is the lower bound of variable i ;
 x_i^{\max} is the upper bound of variable i ;
 ε_i is the desired precision in variable i

More bits mean more possible combinations, and with more possible combinations, more information can be stored. For instance can the 3 bits in g_5 be combined in $2^3 = 8$ different ways, from 000 to 111. The 8 bits in g_1 on the other hand, can be combined in 256 ways.

The phenotype of the individual is obtained through the genotype-phenotype mapping. The string is divided into genes, which are read and translated individually. The mapping is done according to a “record of phenotyping parameters” [26], a template that shows how the information in the genes should be used.

2.3.4 Fitness evaluation

The fitness of an individual is determined by the objective function of the phenotype [26]. In minimizing problems, a low fitness value is desirable, and vice versa. The problem is that the strings sometimes may represent invalid solutions to the problem, even if their fitness is very good. This obviously creates a problem for the GA, but it is taken into consideration by complementing the objective function with a record of *constraints*. There are some sophisticated methods to handle constraints in GA, but the most common method is to use a *penalty function* [7]; if a constraint is violated, a numerical *penalty* is assigned to the fitness value, making it less attractive.

2.3.5 Selection

For the reproduction, individuals with good fitness are chosen to form a mating pool. There exist many different ways to choose individuals for the mating pool, but the main idea is that the better the fitness is, the higher the probability is to be chosen [7, 24, 26]. The mating pool has the same size as the population, but good individuals are more frequent due to duplication. A popular selection method is the *tournament selection* [7, 25]. In this method, small “tournaments” between randomly selected individuals are held, simply meaning that the individual with the best fitness in the group is selected. With a population size of N , N

tournaments are held to fill the mating pool. This way, no copy of the worst individual is selected [25].

2.3.6 Cross-over

With the hope of finding better solutions, the strings in the mating pool are crossed over with each other with the intention of creating a better population. Just as in the selection, there are different cross-over operators, but the main idea is that two random individuals from the mating pool are chosen as parents, and some portion of their strings are switched to create two children [7]. Three usual cross-over methods are given below [7, 26]:

- **Single point cross-over:** The two parent strings are cut at a random spot, and the pieces are put together to make two children:

$$\begin{array}{ccc} 11111111|1111 & \Rightarrow & 111111110000 \\ 00000000|0000 & & 000000001111 \\ \textit{Parents} & & \textit{Children} \end{array}$$

- **Two point cross-over:** The parent strings are cut twice at two random spots to create the children:

$$\begin{array}{ccc} 11|111|1111111 & \Rightarrow & 001110000000 \\ 00|000|0000000 & & 110001111111 \\ \textit{Parents} & & \textit{Children} \end{array}$$

- **Uniform cross-over:** Each bit in the child strings are copied from either one of the parents at a 50 % probability:

$$\begin{array}{ccc} 111111111111 & \Rightarrow & 100101101000 \\ 000000000000 & & 011010010111 \\ \textit{Parents} & & \textit{Children} \end{array}$$

2.3.6.1 Mutation

In the creation of new children, there is always a small probability for each bit in the string to change from 0 to 1 or vice versa. If so, the child is *mutated*:

$$\begin{array}{ccc} 000000000000 & \Rightarrow & 001000000000 \\ \textit{Mutation} & & \end{array}$$

The purpose of this feature is to maintain the diversity amongst the individuals [7], and to prevent the algorithm from getting stuck in a local minimum [26]. The mutation probability should not be too high since in that case the GA turns into random search [26].

2.3.6.2 Elitism

Elitism means that the best or a few of the best individuals are copied into the new generation directly as they are. This ensures that a good solution doesn't get destroyed or unfavorably mutated in the cross-over phase, which significantly improves the performance of the GA [26].

2.3.7 The schemata

So far, the GA search has been described as a chain of operators that repeatedly combines high fitness individuals in the quest for even fitter ones. This may seem like a convenient explanation to the efficiency of GA's, but a very important feature is yet to be described; *the schemata*. The schemata (singular; schema) are templates that describe subsection similarities amongst the strings [26]. A schema consists of 1, 0 and *, where * represents either 1 or 0, for instance;

```
0110*****
```

is a template that represents all thirty-bit strings that starts with 0110.

Schemata are used to look after the similarities amongst individuals of high fitness, since this works as a guide in the search [13]. By doing so, the optimal solution doesn't necessarily have to be a result of evaluating every conceivable combination of bits. Instead, highly fit schemata ("part-solutions") can be sampled and combined to form individuals of potentially better fitness. This reduction of the problem's complexity is actually the main explanation to why GAs work so well [7, 13]. Since the "part-solutions" are compared with building blocks, it is referred to as *The Building Block Hypothesis* [7, 13, 26]. While exceptionally good schemata or *Building Blocks* are kept to propagate in future generations, random cross-over operators with many cross-over points should be avoided, otherwise the building blocks are more likely to be disrupted and the performance is decreased [26].

The ideal is short, low defining schemata of high fitness since they are more likely to survive cross-over and mutation [13]. With this in mind, it is understandable why a small alphabet (like the binary alphabet of zeros and ones) is preferred; Low defining building blocks of high fitness can be spotted more easily. With the binary alphabet, a schemata of three defined bits represents one of eight "part solutions". But to the contrary, if the chromosome was coded with the real numbers 0-9, a schema with three defined numbers would represent one of a *thousand* "part solutions", and the probability of finding such similarities is significantly lower.

2.3.8 Population size

The size of the population should be chosen according to the complexity of the problem. In highly complex problems, the gene pool needs to be extensive enough so that the whole search space can be explored [26]. But of course, with a bigger population the computational time and effort is increased, so the upper limit of the population size should be determined by available computer power and time. In the GA literature there are some proposals on how to choose the size of the population, for instance with respect to the string length (binary coded GA) or the nonlinearity of the problem [7]. In the binary case, a good take-off point is to have a population size in the same order as the length of the strings [7].

2.3.9 Genetic Algorithms in MATLAB

The "Genetic Algorithm and Direct Search Toolbox" in MATLAB enables the use of GA on a wide range of problems. The toolbox includes many different options, e.g. different selection, crossover and mutation operators, and has a built in graphical interface. Due to fact that it is written in open MATLAB language, the user is free to inspect and modify the algorithms, or create own, custom functions [12].

To apply the GA toolbox on an optimization problem, the MATLAB functions has to be implemented with a problem specific representation, genotype/phenotype mapping, fitness evaluation and penalty function. Other than that, the possibilities are practically endless [12].

2.4 Axially loaded bars according to Eurocode 3 [9]

2.4.1 Tension

The basic criterion for a steel bar subjected to tensile stress is that it at each cross-section satisfies:

$$\frac{N_{Ed}}{N_{t,Rd}} \leq 1.0;$$

where N_{Ed} is the design value of the tension force;

$$N_{t,Rd} = N_{pl,Rd} \text{ if no holes are present.}$$

$N_{pl,Rd}$ is the design plastic resistance, and it is calculated as:

$$N_{pl,Rd} = \frac{A \cdot f_y}{\gamma_{M0}};$$

where A is the area of the gross cross-section;

f_y is the yield strength of the steel;

γ_{M0} is a partial factor for resistance of cross sections. $\gamma_{M0} = 1.0$ is recommended for buildings.

The yield strength of steel is dependant on its quality. In this work, five different steel qualities are used, namely S 235, S 275, S 355, S 420 and S 460. The numbers represent the yield strength², f_y in N/mm^2 . If the yield strength is exceeded in any of the members, plastic deformation or even fractures will occur in the truss.

² If the nominal thickness of the element is greater than 40 mm, the yield strength should be reduced according to EC3 [9], but that is not necessary in this case since 40 mm is the thickest element used, see Appendix B).

2.4.2 Compression

In the case of compressive stress, buckling effects have to be taken into consideration. Buckling is a sudden failure of a structural element under compressive stress. Buckling occurs at a level of stress that is less of what the material itself can withstand, and is therefore primarily dependant on the geometrical properties of the element. For bars with closed cross-sections, two types of buckling are treated in the Eurocode, namely *flexural* buckling and *local* buckling. This is described in section 2.4.4, after a description of the effects of different cross-sectional properties in section 2.4.3.

2.4.3 Cross-section classification

Different cross-sections have different local buckling resistance, depending on the inner width - to - thickness ratio. Local buckling can be compared with the collapse of an empty soda can under axial compression (as contrary to flexural buckling that can be compared with the collapse of long, raw spaghetti). To cope with the varying local stability among the cross sections, they are divided into four different cross-sectional classes. The way in which the classification is done depends on what kind of profile it is. In this work, square, hot finished hollow profiles are used, and in that case, the different cross-sectional classes are calculated as (with c and t as in figure 2 below):

class 1 if $\frac{c}{t} \leq 33\varepsilon$;

class 2 if $\frac{c}{t} \leq 38\varepsilon$;

class 3 if $\frac{c}{t} \leq 42\varepsilon$;

and class 4 if it fails to satisfy the limit for class 3;

where ε is equal to $\sqrt{235/f_y}$ with f_y in N/mm^2 .

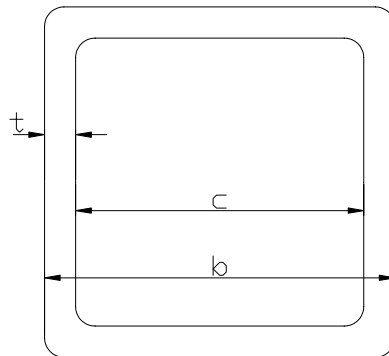


Figure 2: Designations of a square hollow profile used in this work

The different classes represent to which extent a cross-section's local buckling resistance limits it's over all capacity. They are described in Eurocode as follows [9]:

- “Class 1 cross-sections are those which can form a plastic hinge with the rotation capacity required from plastic analysis without reduction of the resistance.”

- “Class 2 cross-sections are those which can develop their plastic moment resistance, but have limited rotation capacity because of local buckling.”
- “Class 3 cross-sections are those in which the stress in the extreme compression fibre of the steel member assuming an elastic distribution of stresses can reach yield strength, but local buckling is liable to prevent development of the plastic moment resistance.”
- “Class 4 cross sections are those in which local buckling will occur before attainment of yield stress in one or more parts of the cross section.”

Since the bars in a truss are only subjected to uniform normal stresses, the only thing that matters is whether or not a cross section is class 1, 2 or 3, i.e. if the yield stress can be reached or not. To deal with the reduced stress capacity of class 4 cross-sections, their cross-sectional area is reduced in the calculations, see section 2.4.4.2.

2.4.4 Buckling resistance of steel bars [9]

Buckling failure is dependent on the slenderness of the bar. A slender bar subjected to a compressive normal stress is much more inclined to buckle than a compact one, subjected to the same stress. The slenderness itself is dependent on the cross-sectional properties, the length of the bar and the end support conditions. To avoid buckling in any member of the truss, the maximum allowed compressive stress must be limited with respect to these parameters. The following criterion must be fulfilled for all the members to assure that buckling is unlikely:

$$\frac{N_{Ed}}{N_{b,Rd}} \leq 1.0$$

where N_{Ed} is the design value of the compression force;
 $N_{b,Rd}$ is the design buckling resistance of the compression member.

$N_{b,Rd}$ should be taken as:

$$N_{b,Rd} = \frac{\chi A f_y}{\gamma_{M1}} \text{ for class 1, 2 and 3 cross-sections;}$$

$$N_{b,Rd} = \frac{\chi A_{eff} f_y}{\gamma_{M1}} \text{ for class 4 cross-section}$$

where χ is the reduction factor for the relevant buckling mode, see section 2.4.4.1;
 A is the cross-sectional area;
 A_{eff} is the effective cross-sectional area, see section 2.4.4.2;
 γ_{M1} is a partial factor for instability resistance (the recommendation is $\gamma_{M1} = 1.0$ for buildings).

2.4.4.1 Buckling reduction factor, χ [9]

The factor χ determines how much of the compressive stress capacity of a bar can be used before it is assumed to buckle.

$$\chi = \frac{1}{\Phi + \sqrt{\Phi^2 - \bar{\lambda}^2}}, \text{ but } \chi \leq 1.0$$

where $\Phi = 0.5 \left[1 + \alpha(\bar{\lambda} - 0.2) + \bar{\lambda}^2 \right]$

$$\bar{\lambda} = \sqrt{\frac{Af_y}{N_{cr}}} \text{ for class 1, 2 and 3 cross-sections}$$

$$\bar{\lambda} = \sqrt{\frac{A_{eff} f_y}{N_{cr}}} \text{ for class 4 cross-sections}$$

α is an imperfection factor. For hot finished, hollow sections, $\alpha = 0.21$ for steel quality S 235 – S 420, and $\alpha = 0.13$ for S 460

N_{cr} is the critical axial force for the relevant buckling mode, see section 2.4.4.3

If $\bar{\lambda} \leq 0.2$, or $\frac{N_{ed}}{N_{cr}} \leq 0.04$ the buckling effects may be ignored, i.e. $\chi = 1.0$.

2.4.4.2 Effective cross-sectional area, A_{eff} [10]

If a compressed cross-section is class 4, the cross-sectional area should be reduced in the calculations as (due to the risk of local buckling):

$$A_{eff} = \rho A$$

where A is the cross-sectional area;

$$\rho = \frac{\bar{\lambda}_p - 0.055(3 + \psi)}{\bar{\lambda}_p^2} \leq 1.0$$

where ψ is 1.0 for uniform stress distribution, as assumed here;

$$\bar{\lambda}_p = \frac{c/t}{28.4\epsilon\sqrt{k_\sigma}}$$

where c is the inner width of the cross-section, see figure 2;

t is the thickness of the cross-section, see figure 2;

k_σ is a buckling factor related to ψ , in this case is $k_\sigma = 4.0$;

2.4.4.3 Critical load, N_{cr}

When a bar buckles, it does it in different shapes, or *modes*, depending on the end support conditions. Leonhard Euler (1707-1783) derived the critical load for four different conditions. In the case of trusses, both ends are assumed to be hinged, meaning they are free to rotate. The buckling mode will consequently be the shape of a bow, i.e. Euler buckling mode 2, see figure 3 b):

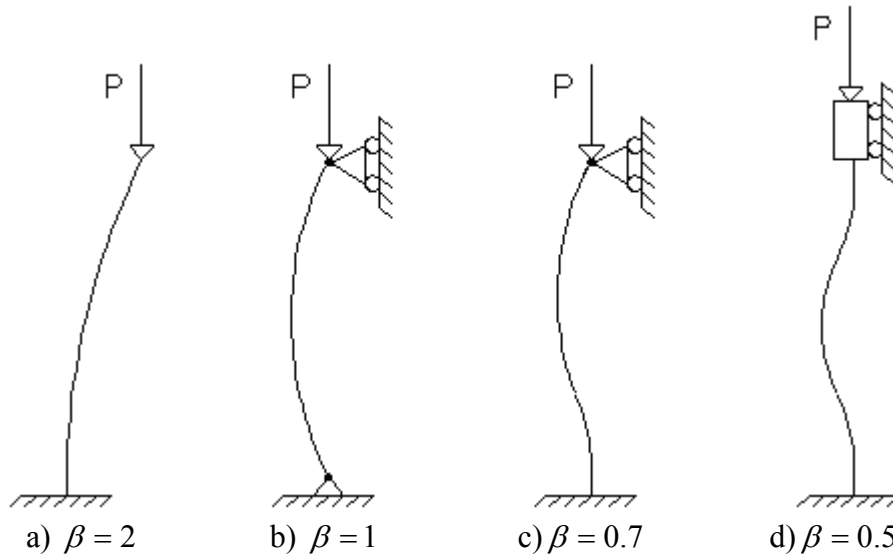


Figure 3: Euler's four derived buckling modes.

The critical load is defined as:

$$N_{cr} = \frac{\pi^2 EI}{(\beta L)^2}$$

where

- β is the effective buckling length, see figure 3;
- E is the modulus of elasticity, 210 GPa for steel;
- I is the moment of inertia for the cross-section;
- L is the length of the element

3 Proposed algorithm

3.1 General

The proposed optimization algorithm is a bit-string encoded genetic algorithm, designed to generate feasible planar steel trusses with minimum weight according to Eurocode 3. It handles size, shape and topology optimization simultaneously. All the elements in a generated truss are chosen from the table in appendix B), and the positions of the nodes are chosen with the precision of one tenth of a metre. The calculations are idealized; neither the dead weight of the structure, nor the three-dimensional stability is taken into consideration. Figure 14 shows a flowchart of the algorithm, and the complete MATLAB files are enclosed in appendix A).

3.2 Constraints

Besides the general limitations given in section 2.4, the following five constraints are implemented in the algorithm.

3.2.1 Constraint 1: Fabricational

The first constraint is that a feasible truss must only consist of elements of available dimensions; otherwise the algorithm would not have any practical application. The available profiles list is taken from Budapest University of Technology and Economics [28], and refers to hot finished, hollow square sections, see figure 2. The list of available dimensions is enclosed in Appendix B).

Hot finished profiles are created by letting hot steel material pass through rolls that gives the bar its intended shape and dimensions. Afterwards it is left to cool down, and depending on the element thickness, the different parts might cool down at a different rate, creating built-in stresses in the element. This is taken into account when determining the buckling resistance, see section 2.4.4. When the algorithm is creating a truss, it will only pick elements from the table in Appendix B), which means that this constraint will automatically be satisfied. The list of available profiles is very detailed and contains profiles in the range from 45x45 to 700x700 mm. This enables the algorithm to find very precise solutions.

3.2.2 Constraint 2: Basic nodes

At the beginning of the algorithm the user is asked to specify the coordinates of all the basic nodes, i.e. nodes where there is either a support or a load. A generated truss must have all of the basic nodes to be feasible. This constraint will also automatically be satisfied.

3.2.3 Constraint 3: Stability

A generated truss must not be a mechanism; it has to be kinematically stable. A way to check if a structure is stable is to calculate the determinant of its stiffness matrix. If it turns out to be zero, the structure is not stable, but all other values of the determinant say it is.

3.2.4 Constraint 4: Nodal displacements

Displacement restrictions are often crucial in structural engineering. The structure is not allowed to deflect more than a certain limit when it is in use. Normally the limit is related to the span width of the structure, e.g. $\delta_{\max} = L/300$. The maximum allowed deflection is often chosen in the interval $L/500 \leq \delta_{\max} \leq L/150$. In this case the maximum allowed deflection put

to $L_x/250$ in the y-direction (vertically) and $L_y/250$ in the x-direction (horizontally) as default values, which are normally used limits. In cases with special demands on the displacement, it can of course be changed.

3.2.5 Constraint 5: Constructability

Even if a generated truss consists of elements that are taken from an available profiles list, and the deflection and the element stresses are within the limits given in EC3, it is not necessarily true that it is feasible. The algorithm has to be given some additional *constructability constraints*. This is taken into consideration by not allowing two or more elements to have both their nodes in common and two nodes cannot exist in the very same place. Furthermore, to avoid having infinite elements stuck in any of the nodes, elements are not allowed to start and end up in the very same node. A violation of any of these constraints will result in penalty.

3.3 Constraint management

The penalty function or *fitness function* works in two different ways, depending on which constraint is violated. Firstly, each truss is checked according to constraints three and five, namely stability and constructability. A violation here indicates that the truss is not feasible and it is assigned with a large *constant* penalty and excluded from further calculations³. On the other hand, if the truss passes this first test, element stresses and the deflection are calculated in a FEM routine⁴. Here, the stress limits due to material strength and buckling resistance according to EC3, are calculated for each element as well. If the stress is violating the EC3 limit in any element, a penalty is assigned. The size of the penalty is in this case *proportional* to the violation. The same principle applies on the deflection.

³ There are two reasons to this exclusion; firstly, no computational effort is wasted on a non-feasible truss, and secondly, if the calculations would continue there is a big chance the algorithm would get stuck in the FEM routine due to the possibly very odd properties of the truss.

⁴ The FEM routine is mainly put together by MATLAB scripts taken from CALFEM – A Finite Element Toolbox [3].

3.3.1 Topology optimization

The proposed algorithm contains two options for the topology optimization; *Ground structure method* and *reduced method*. The ground structure method is commonly used in search for an optimal topology. In this method, bars are initially put in every possible position, and gradually the unnecessary bars (and nodes) are removed until an optimal structure remains. The removal of bars is possible through the “zero-bars” described in section 3.3.1.1. An example of a ground structure with 6 nodes is shown in figure 4:

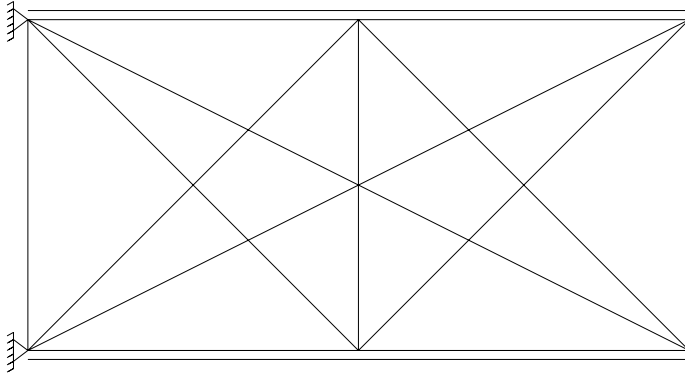


Figure 4: An example of a ground structure. The number of elements in a completely connected ground structure equals $\binom{nn}{2}$ where nn is the number of nodes in the structure.

The second option, *the reduced method*, is developed in connection with this work as an alternative to the ground structure method. With an available profiles list as extensive as the one used here, the ground structure method may be inappropriate if the number of elements in the structure is high. Also, with a very complex structure, the FEM-routine consumes more time per iteration. A complete description of the technique is given in section 3.4.

3.3.1.1 Zero-bars

Since the length of the binary chromosome string has to stay the same for all individuals for the crossover to function properly, the elements that are removed still have to have the same representation on the chromosome. This is solved by replacing an unnecessary bar with a “zero-bar”, a bar with infinitesimal stiffness and mass in the structure. Thirty⁵ zero-bars are added to the available profiles list, to generate a reasonable probability (about 12 percent) that “no bar” is put in a certain place.

⁵ The number of real profiles is 226, which are represented by 8 bits, but since 8 bits can be combined in 256 ways, the number of zero-bars is set to 30 to make a grand total of 256 available profiles.

3.4 Reduced method for topology optimization

The motivation to develop an alternative to the ground structure method comes from the fact that it isn't suitable for bigger structures when the list of available profiles is very extensive or, for that matter, continuous. The number of possibilities gets unnecessarily high even at relatively simple structures with few nodes, which slows down the algorithm. The explanation is of course that it always involves the maximum number of elements, of which the majority often are superfluous. A good effect of the ground structure method is on the other hand that the stability amongst all initial structures is guaranteed.

The main purpose with the reduced method is to work with fewer elements to reduce the number of unnecessary possibilities for the algorithm. The main concept is that some of the elements are arranged in a simple *basic structure* that guarantees stability amongst all individuals. To this structure there are a number of "free" bars with variable topology added, see section 3.4.1. Just like in the ground structure method, unnecessary nodes and elements can be removed thanks to the "zero-bars" described in section 3.3.1.1.

3.4.1 Basic structure and “free” elements

In the reduced method, the main part of the elements is assembled automatically in a certain pattern to form a *basic structure* based on triangles. The topology of the basic structure has no representation on the chromosome string. Instead, it is individually determined with respect to the horizontal position of the nodes. To demonstrate how the basic structure is obtained, consider the four supports, five loads and five arbitrarily positioned nodes in figure 5:

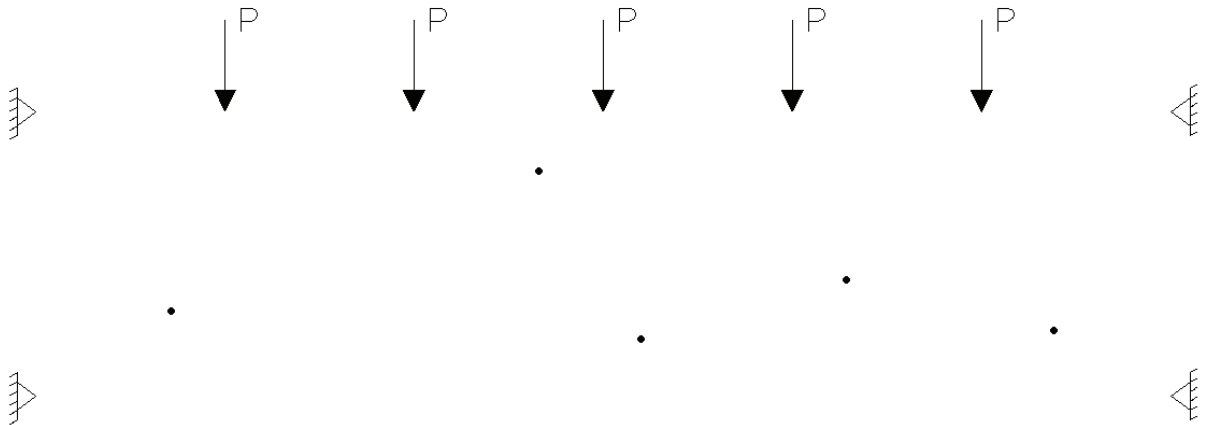


Figure 5: Nine *basic* nodes and five arbitrarily positioned nodes.

The first step is to sort all nodes with respect to their horizontal position and connect the first three with bars, see figure 6:

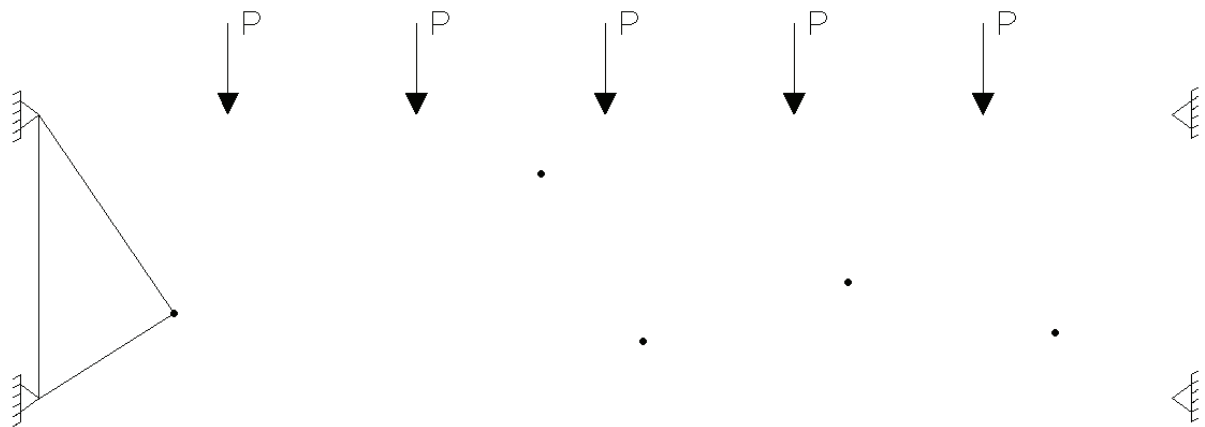


Figure 6: The first three nodes are connected.

In the following steps, the next node is connected with the two previous ones until the last node is connected, see figure 7:

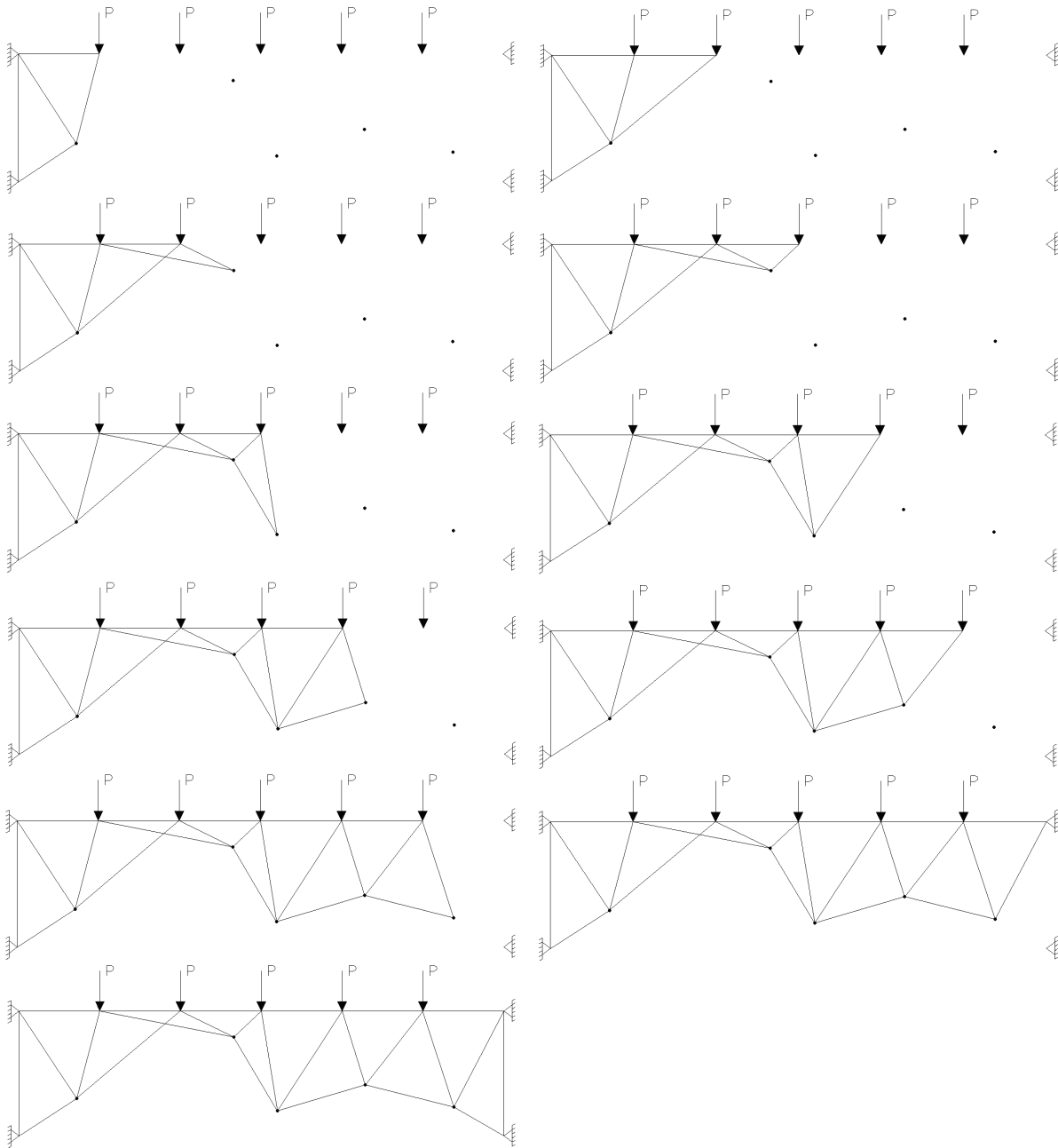


Figure 7: The step-by-step creation of the basic structure.

This way, the elements in the basic structure are always assembled in the characteristic triangular pattern that guarantees stability. The rest of the elements that are used can be said to be free, and can attain any topological position. The topology of these elements is represented on the chromosome string. Their purpose is to fill out the possible shortcomings of the basic structure, thus obviously not all topologies can be obtained with it alone.

The number of elements that is needed to fulfil the basic structure is dependent on the number of nodes in the structure. If the number of elements is chosen as;

$$neft = 2nn - 3$$

where $neft$ is the number of elements;
 nn is the number of nodes,

the basic structure adds up evenly.

Since the maximal number of elements in a structure is $\binom{nn}{2}$, the number of free elements can be chosen in the interval

$$0 \leq nevt \leq \binom{nn}{2} - neft$$

where $nevt$ is the number of free elements (elements with variable topology).

The number of free bars should be chosen with respect to the complexity of the problem. With too many, the algorithm becomes inefficient since the risk of imbrications (and therefore big penalties) gets very substantial. With too few elements, there is a risk of never finding the optimal topology. The numbers given in table 1 are a proposed guideline. They correspond to about one tenth of $\binom{nn}{2} - neft$.

Table 1: Proposed number of free elements

Number of nodes	Number of free elements
6	1
7	1
8	2
9	2
10	3
11	4
12	5
13	6
14	7
15	8
16	9
17	10
18	12
19	14
20	15

Figure 8 shows the resulting basic structure in figure 7 with six free elements added:

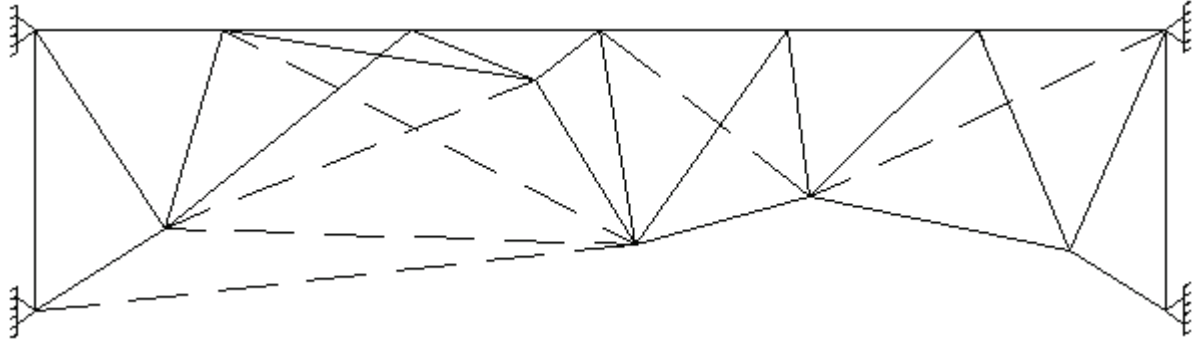


Figure 8: The resulting basic structure in figure 7 with six free elements. The free elements are represented by dotted lines.

The reason why the basic structure is necessary is that it guarantees a certain quality amongst the individuals. This is especially important in the initial population. If the topology of all elements would be chosen randomly, the probability of finding a stable structure would be very low, and the probability of finding a structure of reasonable effectiveness would practically be equal to zero, see the example in figure 9:

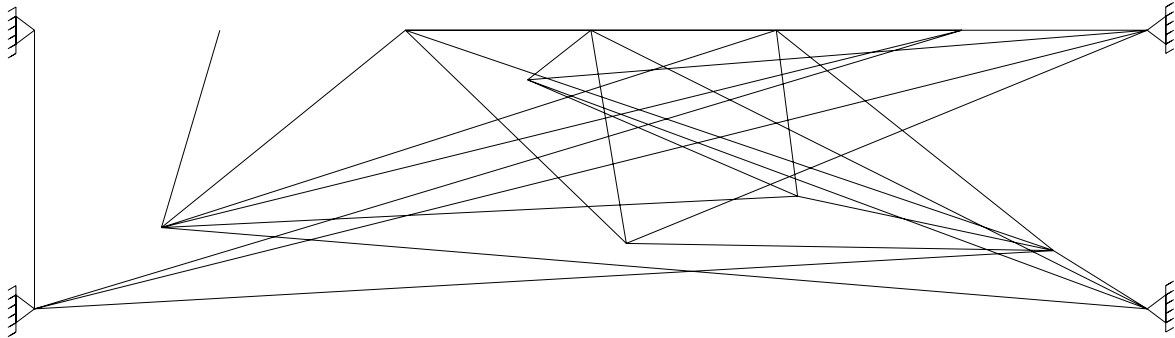


Figure 9: An example of a structure with randomly chosen topology. With structures like this in the initial population, the algorithm would have a very tough job finding the optimal solution.

Additionally, due to the constant penalty given to non stable, or non constructible structures, most of the trusses would come out “equally bad”, leaving the algorithm with no clue on where to go.

3.4.2 Advantages with the reduced method

By reducing the number of possible solutions, the chromosome string doesn't have to contain as many bits, meaning that the search space gets narrowed down. Figure 10 shows the difference in bit string length between the ground structure method and the reduced method if the number of free elements is chosen according to table 1. The different lines represent different numbers of available profiles. As seen in the figure, the reduced method has a bigger advantage the more different profiles there is. Normally in papers on structural optimization, the number of different profiles is 32.

Bear in mind that if the chromosome string length is reduced with a single bit, the search space is divided in half.

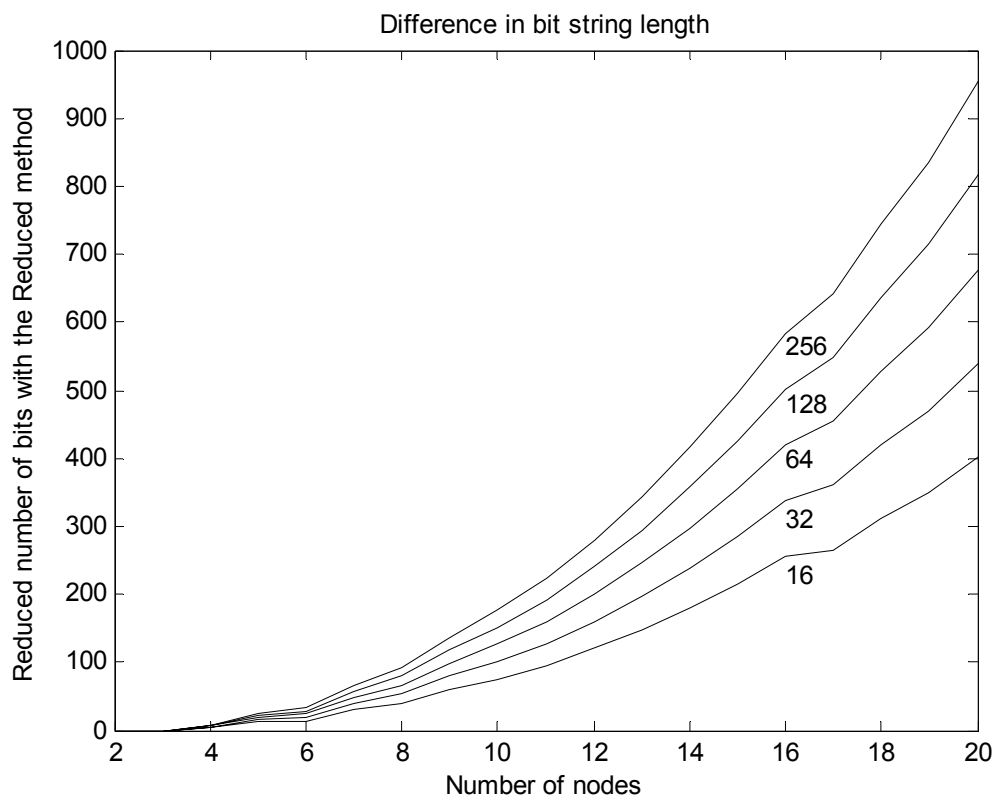


Figure 10: Reduced number of bits in the chromosome string as a function of the number of nodes in the structure. This means that in ten-node structures with 256 different profiles available, the size of the search space with the ground structure method is ten to the power of 52 times bigger than it is with the reduced method (!).

3.4.3 Limitations

There are limitations to this method, though. If too many “free” elements are involved, it demands more extensive representation, which increases the number of possibilities. For instance, if half of $\binom{nn}{2} - n_{eft}$ are involved, the diagram would look like in figure 11:

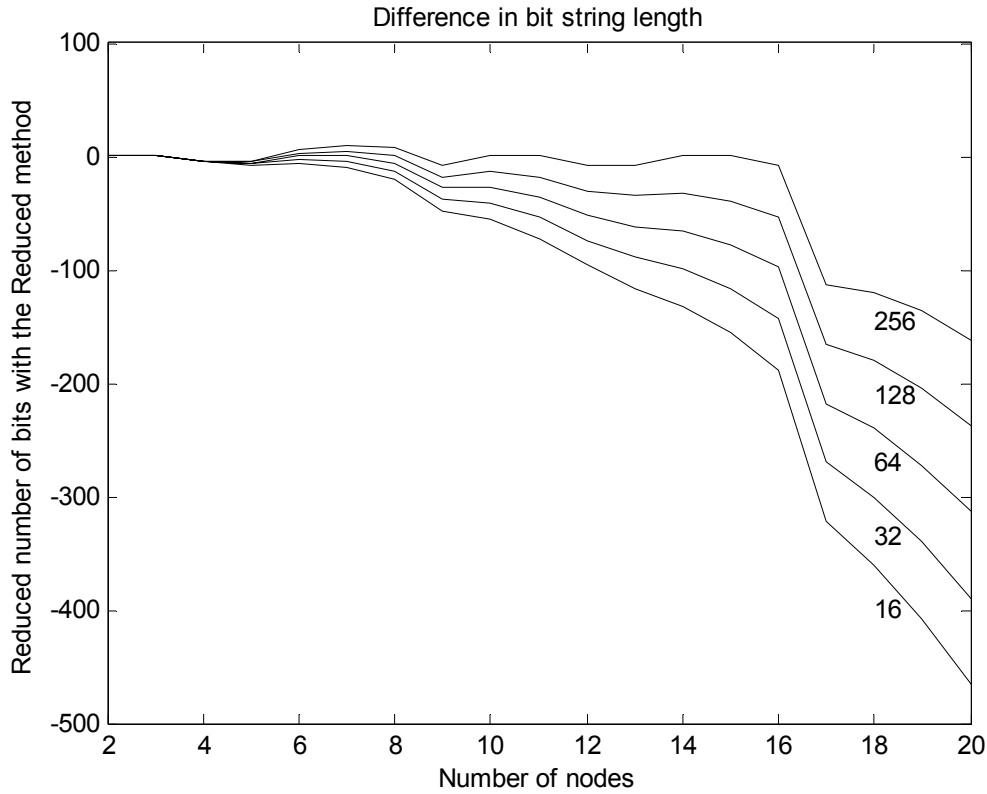


Figure 11: With too many free elements, the reduced method demands longer bit strings for representation.

The limit to when the reduced method gives shorter representation in the range of 2-20 nodes depends on how many different profiles there are available, but approximately the values in table 2 denotes the limits:

Table 2: The maximum number of free elements in the range of 2-20 nodes.

Number of different profiles in use	Maximum number of free elements
16	$0.25 \times \left(\binom{nn}{2} - neft \right)$
32	$0.30 \times \left(\binom{nn}{2} - neft \right)$
64	$0.35 \times \left(\binom{nn}{2} - neft \right)$
128	$0.40 \times \left(\binom{nn}{2} - neft \right)$
256	$0.40 \times \left(\binom{nn}{2} - neft \right)$

In the range of 2-100 nodes with $0.1 \times \left(\binom{nn}{2} - neft \right)$ free elements, the diagram looks like in figure 12:

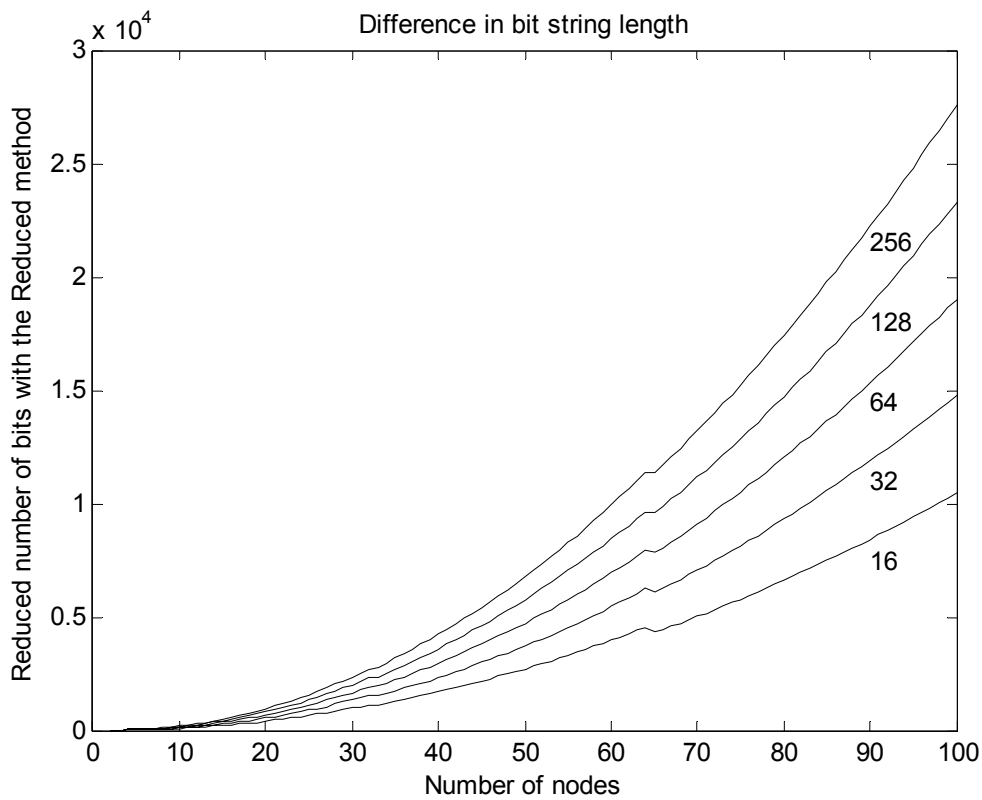


Figure 12: With one tenth of the available free elements involved, the reduced method has the upper hand up to 100 nodes as well.

But, if the number of free nodes would increase to $0.33 \times \left(\binom{nn}{2} - neft \right)$, the advantage disappears, at least in the case of up to 64 different profiles, see figure 13:

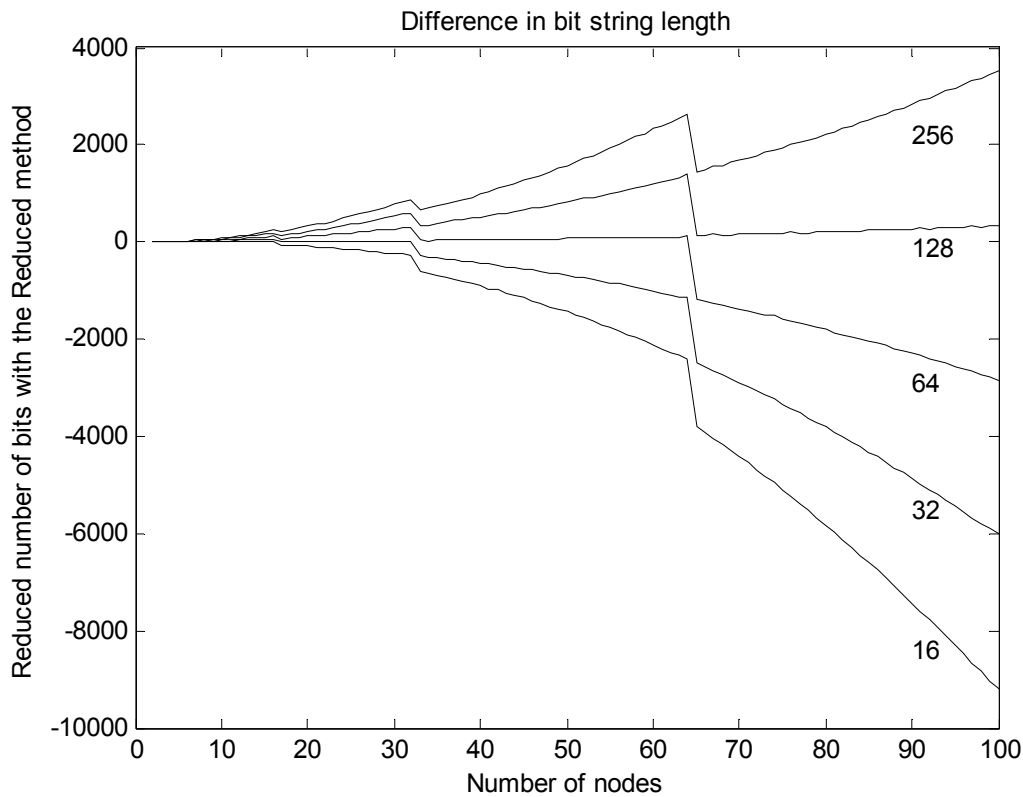


Figure 13: If too many free elements are involved, the extra need to express their topology demands longer bit strings with the reduced method. The clear dips in the diagram that occurs at eight, sixteen, thirty-two and sixty-four nodes are caused by the increased number of bits that are needed to express more possible topologies.

This indicates that there seems to be an upper limit to when the reduced method should be preferred over the ground structure method, as suggestion 32 or 64 nodes. To optimize a structure with more than 32 nodes in terms of size and topology with the current technique would on the other hand be an overwhelming task for a normal computer, due to the almost infinite amount of possibilities.

Additionally, when working with small structures of three to five nodes, the reduced method isn't favourable, even if the search space is smaller. To be sure that important topologies aren't impossible to express, there should always be at least one or two free elements available. In smaller structures, the number of unique topologies is very small, and therefore will the risk of imbrications be very obvious. This lowers the average fitness of the population and stalls the algorithm. Thus, in the range of 3-5 nodes, the ground structure method should be considered.

As a conclusion, the reduced method should be considered in the range of six to about 32 or 64 nodes, or even higher if the number of different profiles is very high.

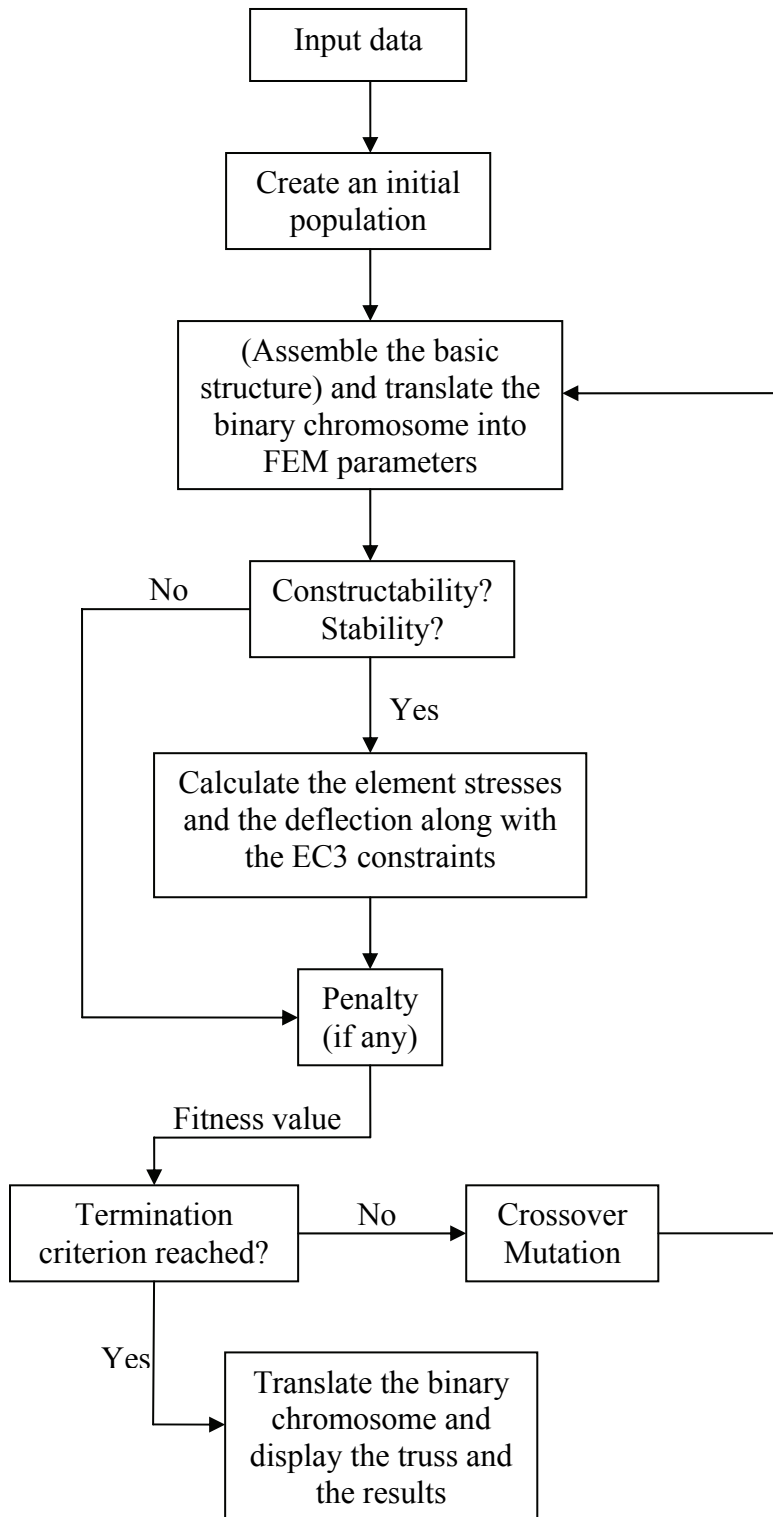


Figure 14: A flowchart of the optimization algorithm.

3.5 Effects of simultaneous size- shape- and topology optimization

Even if the reduced method for topology optimization offers a relief to the number of possibilities, simultaneous optimization of size- shape- and topology still means an extreme number of possibilities, even for quite simple structures. To ensure that premature convergence doesn't occur, a very large population is therefore needed, and as a consequence the calculations are very time consuming when working on a regular PC (~2 GHz processor).

An effective way of reducing the complexity of the problem is to start out as simple as possible, in other words with as few nodes and elements as possible. In this way the number of possibilities is kept at a minimum, meaning smaller populations and less computer effort is needed. With limited computer power available this is absolutely necessary. The problem is to know what to begin with. It might be appropriate to do two runs on an optimization. The first run will probably not provide a global minimum, but gives a hint on the number of nodes and elements to start out with. This possible reduction of complexity increases the probability that a global minimum is found in the second run.

By doing so, there is a risk that the global best solution is excluded. But in return, a reasonable calculation time is obtained.

4 Results

4.1.1 Benchmark problem

As mentioned before, this ten-bar truss is often used as a benchmark problem in structural optimization. This structure appears in most of the papers on the subject. The shape of it (the position of the nodes) is rarely altered though. Simultaneous optimization of size, shape and topology is even rarer, which makes it a bit interesting.

The truss has two vertical supports with a distance of 9.144 metres (360 inches) and two loads of 445.374 kN (100 kips) at 9.144 and 18.288 metres from the lower support, see figure 15.

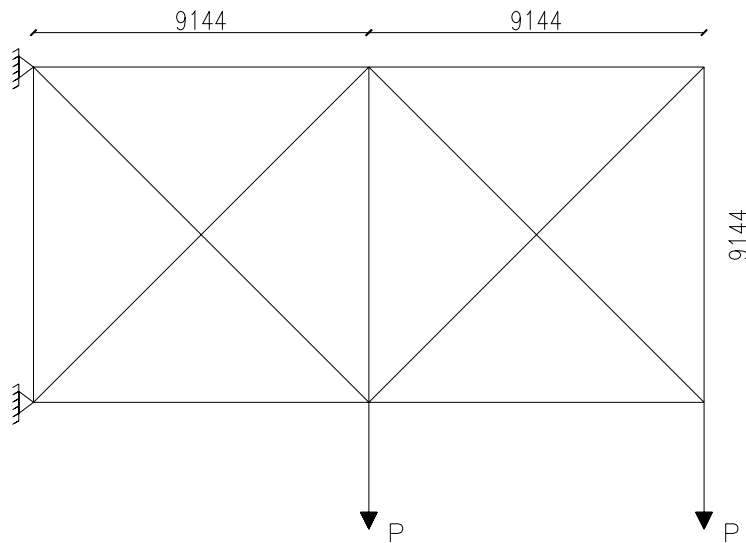


Figure 15: The ten-bar truss.

Most commonly, aluminium alloy is used, with $E = 68.95 \text{ GPa}$ (10^4 ksi), $\rho = 2768 \text{ kg/m}^3$ (0.1 lb/in^3) and the element stresses are limited to 172.37 MPa (25 ksi) in both tension and compression, i.e. buckling is ignored. The displacements are limited to 50.8 mm (2 in) both horizontally and vertically. Some good results with these parameters are⁶:

1. **2222.22 kg⁷** (4899.15 lbs) by Deb and Gulati (2001) [8]. Size and topology optimization by a genetic algorithm.
2. **2241.97 kg** (4942.7 lbs) by Hajela and Lee (1995) [15]. Size and topology optimization by a genetic algorithm.
3. **2295.59 kg** (5060.9 lbs) by Li, Huang and Liu (2006) [21]. Size optimization by a particle swarm optimizer.
4. **2301.09 kg** (5073.03 lbs) by Kripakaran, Gupta and Baugh Jr. (2007) [19]. Size optimization by a hybrid search method.
5. **2322.08 kg** (5119.3 lbs) by Galante (1996) [11]. Size and shape optimization by a genetic algorithm.

⁶ The best solution presented in the literature is 1982.13 kg (4369.84 lbs), found in Wu and Chow (1995) [31]. Here, the same conditions are used, but a control reveals that the displacement limit is violated ($\delta_{\max} = 2.61 \text{ in}$).

⁷ Deb and Gulati presents an even better solution as well (2146.24 kg [4731.65 lbs]), but in this truss there is an undesirable overlap.

With these results in the background, a steel version of the truss was optimized with respect to size, shape and topology according to EC3. Looking at the material properties, steel has 3.05 times higher modulus of elasticity, but only 2.84 times higher density, giving it a small advantage in that aspect. On the other hand, EC3 involves buckling stability. If buckling is not taken into account, the calculated trusses are not suitable for real constructions since they are likely to collapse [11].

Two runs were made. In the first run, six nodes and two free elements were used. With steel quality S235, an initial population of 850 and a maximum of 1 500 generations, the result was a structure of five nodes and one free element. The weight is 3 165.22 kg (6 978.11 lbs), and there are many indications that it was not a global minimum. For instance, the maximal displacement was only 91 % of the limit.

Based on the first run, a second run was made with only five nodes and one free element. By this reduction, the number of possibilities decreased with a factor of 35 billions (!), which drastically reduces the workload. Since the list of available profiles is very high, the reduced method for topology optimization was used here as well. With a population size of 850 and a maximum of 2 500 generations, the following structure was obtained after 495 generations:

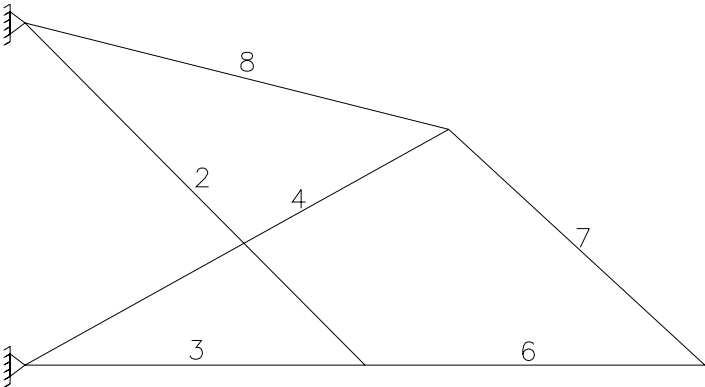


Figure 16: The calculated truss in the second run. The reason why element number one and five don't show is that they weren't needed.

Table 3: Numerical results for the second run of the benchmark problem.

Element no.	Dimensions (b*b*t) (mm)	Start coordinates x,y (m)	End coordinates x,y (m)	Element stresses (MPa)	Element stresses in % of EC3 limit
2	120x120x6	0,9.144	9.144,0	230.2098	97.96
3	300x300x6	0,0	9.144,0	-132.131	78.66
4	250x250x5	0,0	11.4,6.3	-94.1561	97.64
6	200x200x5.6	9.144,0	18.288,0	-111.8235	93.53
7	160x160x6	11.4,6.3	18.288,0	178.5449	75.98
8	150x150x8.8	11.4,6.3	0,9.144	184.7091	78.60

The vertical displacement limit turned out to be critical. $\delta_{\max} = 50.72$ mm for the right load node, or 99.85 % of the limit. The weight of the truss is 2 327.05 kg (5 130.26 lbs), almost in the same class as the lightest aluminum trusses *without* buckling constraints.

The fact that the deflection of the right load node is 99.85 % of the limit, and that element 2 is used to 97.96 % indicates that any change may lead to a violation of some constraint, which gives reason to believe that this is at least a near optimal solution.

In earlier attempts to find an optimal topology to this problem, the shape has been left unaltered, e.g. Deb and Gulati (2001) [8] and Hajela and Lee (1995) [15]. In both those cases, the resulting structures are like in figure 17. This is very similar to the calculated truss in figure 16, which means that no big surprises came from doing a complete optimization.

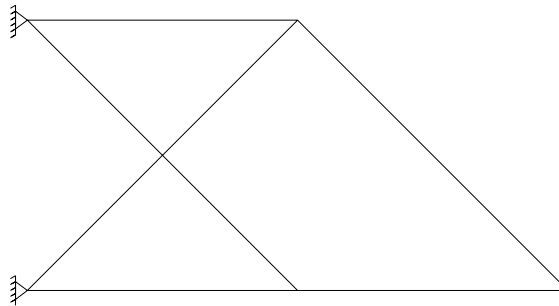


Figure 17: The resulting truss from two earlier attempts to find the optimal topology. Note the similarities between the truss in figure 14 and this one.

As a closure to this section the very same truss was optimized with six nodes fixed as in figure 13. Without variable shape, the number of possibilities is reduced with a factor of one billion, so only one run was made with two free elements. With a population size of 950 and a maximum of 2 500 generations, a structure with a weight of 2 396 kg (5 282 lbs) was found after 585 generations. This is just 3 % more than the result from the more extensive run. The topology was the same as in figure 17. Here it is debatable whether or not the 3 % is worth the extra effort and time.

4.1.1.1 Same truss, tougher limits on displacement

Guerlement, et al. (2001) [14] performed a sizing optimization by a genetic algorithm on this truss in S235 steel, according to EC3. They put the modulus of elasticity to 206 GPa and the displacement limit to rigorous 3.39 cm (1.333 in) over all. The elements were linked into five groups of assumed equal size. Circular hollow profiles were used. Their result under this conditions was 4 761.48 kg (10 497.26 lbs).

This is obviously a better reference for the proposed algorithm, so the truss was optimized once again with new material properties and displacement limit. As in the first example, two runs were made. Steel quality S235 was used. The first run with a population size of 850, a maximum of 1000 generations gave a truss of 3 695 kg (8 147 lb) with six nodes and one free element. This was not much of a reduction, possibly because of the tough limit. But still, one free element less is a one million times reduction of the possibilities.

Based on the first run, the population size was put to 950, the maximal number of generations was put to 500 and the number of free elements was put to one. The result after 123 generations is given in figure 18 and table 4 below:

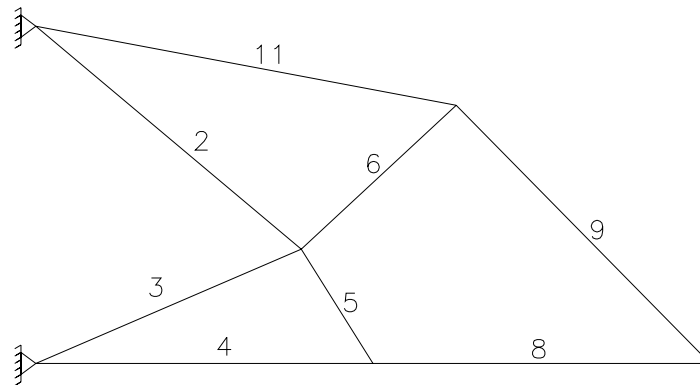


Figure 18: The shape and topology of the truss with stricter displacement limit. It reminds of the shape and topology in the first example, but in this one there is a node in the middle of the cross. This actually increases the buckling stability in element two, three, five and six since the crack length is reduced.

Table 4: Numerical results for the truss in figure 18

Element no.	Dimensions (b*b*t) (mm)	Start coordinates x,y (m)	End coordinates x,y (m)	Element stresses (MPa)	Element stresses in % of EC3 limit
2	180x180x7.1	0,9.144	7.2,3.1	152.31	64.81
3	250x250x6.3	0,0	7.2,3.1	-109.67	61.01
4	250x250x6.3	0,0	9.144,0	-116.84	74.19
5	160x160x5	7.2,3.1	9.144,0	169.58	72.16
6	160x160x5	7.2,3.1	11.4,7	-91.19	58.57
8	180x180x7.1	9.144,0	18.288,0	-89.25	91.41
9	180x180x10	11.4,7	18.288,0	91.89	39.10
11	250x250x8.8	11.4,7	0,9.144	91.48	38.93

The displacement is 33.87 mm vertically, or 99.91 % of the limit. Not surprisingly, this was the critical constraint here as well. The weight of the structure is 3 138.78 kg (6 919.82 lbs), which is a 34 % improvement to Guerlement et al. However, there is actually room for further improvements. The displacement of the left load node is only 17 mm, and element five is only used to 72 % of its capacity. A conclusion of this is that a deeper search is probably needed to reach a global optimum.

4.1.2 A bridge truss

The third example is taken from Soh and Yang (1998) [27], who used a genetic algorithm to find the minimum weight of a 24-m spanned bridge truss made of steel, see figure 19:

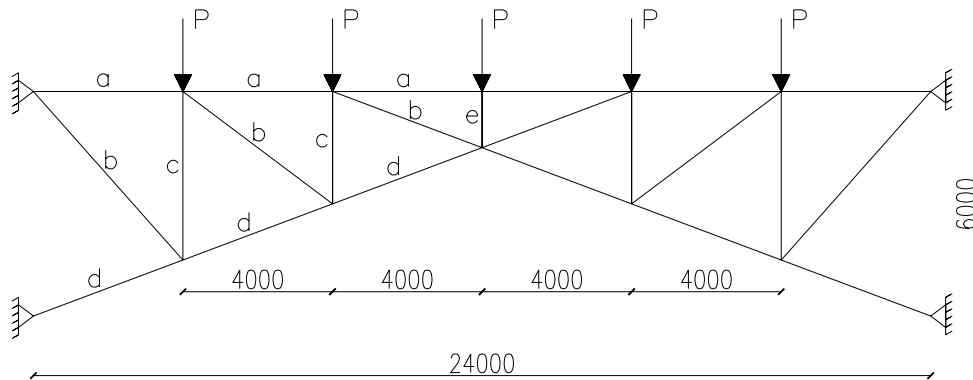


Figure 19: The initial structure to be optimized. The letters a-e represents groups of assumed equal size.

Their optimization was very simplified and referred only to sizing and shape variation⁸. The horizontal and vertical displacement limits were set to 10 mm and 50 mm respectively, and the maximum allowed stress was 140 MPa in both tension and compression.

With an initial population of 40, ran over 70 generations, they got the following result:

Table 5: The cross-sectional areas for each group. The letters refer to figure 19.

Member group	cross-sectional area (mm ²)
a	27.14
b	106.68
c	1433.24
d	5136.46
e	1420
Total weight (kg)	
15 704⁹	

⁸ The calculation was carried out on a 486-PC.

⁹ After going through Soh and Yang's results, it looks as though a mistake has been done. If that is true, the total weight of their truss is 1 273 kg instead of 15 704 kg.

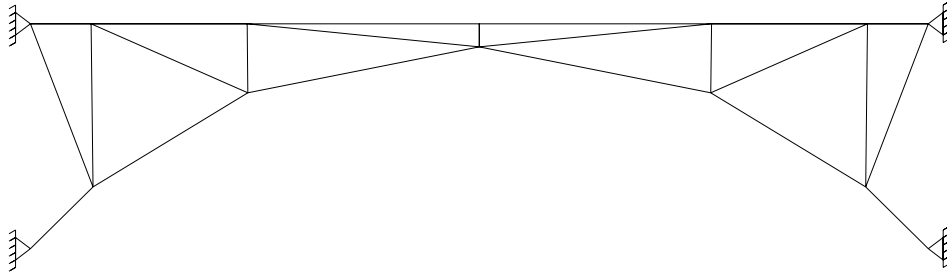


Figure 20: Soh and Yang's calculated truss (1998) [27]. Note that the load nodes have been allowed to attain more favourable positions!

With these results in the background, the same truss was optimized with the proposed algorithm. Instead of having a 140 MPa stress limit, the constraints on buckling stability and yield stress according to EC3 was used. The steel quality was S235, and the same material properties ($E = 210 \text{ GPa}$, $\rho = 7\,850 \text{ kg/m}^3$) and limits on the displacements were applied. The load nodes were not allowed to move.

Due to symmetry, only half the truss was evaluated. Two runs were made here as well. In the first run, eight nodes and two free elements was used. A population size of 1 500 and a maximum number of generations of 150 gave half a bridge truss of six nodes and one free element. The weight it was 1 007 kg (2 220 lb).

With this result as a take-off point, a second run was made with a population size of 1500. This time the number of nodes was set to 6 and the number of free elements to 1 (in the half-model), since more seemed superfluous. The result from this run is declared in figure 21 and table 6 below:

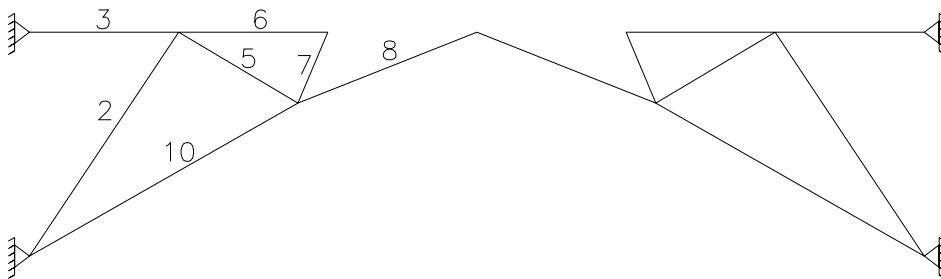


Figure 21 Optimal shape and topology of the bridge truss. Note that the upper horizontal bars were not necessary in the middle. This might look odd, but remember that it is just designed sustain load in the nodes!

Table 6: Numerical results for half the truss. Because of the symmetry, these values translate to the other side as well.

Element no.	Dimensions (b*b*t) (mm)	Start coordinates x,y (m)	End coordinates x,y (m)	Element stresses (MPa)	Element stresses in % of EC3 limit
2	160x160x5	0,0	4,6	-98.9487	81.5075
3	80x80x5	0,6	4,6	231.5789	98.5442
5	50x50x3	4,6	7.2,4.1	191.787	81.6115
6	40x40x3	4,6	8,6	189.6633	80.7078
7	80x80x4	7.2,4.1	8,6	-178.4585	90.4429
8	120x120x4.9	7.2,4.1	12,6	-120.438	94.5043
10	200x200x5	7.2,4.1	0,0	-126.8355	92.0546

The displacements are 8.0237 mm horizontally, and 19.66 mm vertically, which are 80.2366 and 39.32 % of the limits respectively. The overall weight of the structure is $2 \times 617.7226 = 1\,245.7662$ kg, which corresponds to a 92 % improvement of Soh and Yang's result from 1998¹⁰ (?). Obviously, the results can't be compared directly, since a more advanced optimization has been done here. Additionally, higher levels of stress have been allowed, and a much more detailed list of available profiles has been used. Not to mention the difference in computer power. On the other hand, the load nodes weren't allowed to move, making the load scenario more severe in this case.

The calculated weight may seem suspiciously low, but a comparison with the result from the benchmark truss, the weight makes sense. The two structures holds up approximately the same amount of load, but the load scenario in the benchmark case is obviously much more unfavourable since there are only support nodes on one side.

After a look at the displacements and element stresses compared with the limits, it looks as though there are room for further improvements. Maybe not so much in element five and six that principally already are as thin as they can be, but element two has a lot of margin. Element seven, nine and ten could also be a bit thinner without violating any constraint. This is obviously a non-optimal solution, despite the calculation time of 30 hours. A concluding remark is that this problem is an overwhelming task for a 2 GHz PC. With such limited computer power, a better idea would probably be to lock down a few well placed nodes and focus on size and topology optimization to reduce the complexity of the problem.

¹⁰ With 1273 kg, the improvement would be 2.15 %

5 Conclusions

When dealing with simultaneous size- shape and topology optimization, the number of possible solutions reaches extreme levels, which means very big populations and long calculation time. This may suggest that this kind of optimizations should be carried out using parallel computing where the workload is divided to a group of processors. With limited computer power, the number of possibilities should be kept at a corresponding level; otherwise the calculation time will be extreme.

The hypothesis said that the proposed algorithm would return optimal or near optimal solutions to almost any given weight optimization problem concerning steel trusses, but it would take longer time. The part about the time was absolutely true. The other part can be neither validated nor falsified, but with limited computer power it is definitely false, since in that case it is only suitable for simple structures. On the other hand, the shape variation can be left out to create a simpler problem, and according to section 3.4.1 it doesn't necessarily have to be a very big loss. This raises the question whether the shape variation is worth the extra effort; perhaps a clever manual positioning of a few fixed nodes would generate equally good results. It was pointed out in the introduction that in previous studies the optimization has only referred to size and topology or just size, but maybe that is the most effective way to go.

The reduced method for topology optimization offers a relief to the number of possible solutions and should generally be considered in the range of 6-64 nodes, instead of the widely used *ground structure method*. Additionally, it reduces the evaluation time in the FEM routine since it brings simpler structures. A negative effect of this method is that it contributes to the already overwhelming amount of settings of genetic algorithms.

Moreover, optimizations should be carried out with as little "material" as possible, i.e. as few nodes and elements as possible. It might be necessary to do a "pre-run" to get a clue on how many nodes and elements to begin with, in order to include as few possibilities as possible.

Finally, there is no doubt of the efficiency of the genetic algorithm search technique. In the benchmark example, with only five nodes and one free element, the number of possible solutions reaches about 3.9×10^{25} (!). An evaluation of all possibilities one at a time would take a 2 GHz computer about 1.2×10^{17} years, three billion times longer than the age of the Earth, but the GA found a near optimal solution in less than six hours.

5.1 Proposal for further work

The proposal for further work mostly concerns the reduced method for topology optimization proposed in this work. A complete examination of its properties would be in its place, and for instance develop a complete description on how to use it on various types of structures. Its efficiency may also be increased, for instance by implementing a topology schedule that keeps track of engaged topologies. This would avoid imbrications and increase the average fitness of the population.

Finally, to fully enjoy simultaneous optimization of size shape and topology, it is probably appropriate to adjust the algorithm to enable parallel computing.

6 References

1. Alander Jarmo T. (1999). *An Indexed Bibliography of Genetic Algorithm Implementations*. Bibliography. University of Vaasa. Department of Information Technology and Production Economics.
2. Ashlock, Daniel. (2006). *Evolutionary Computation for Modeling and Optimization*. New York: Springer.
3. Austrell, P-E. Dahlblom, O. Lindeman, J. Olsson, A. Olsson, K-G. Persson, K. Petersson, H. Ristinmaa, M. Sandberg, G. Wernberg, P-A. (2004). CALFEM – A FINITE ELEMENT TOOLBOX Version 3.4. Lund: KFS i Lund AB.
4. Christensen, Peter W. Klarbring, Anders. *An Introduction to Structural Optimization*. (2008). Springer Netherlands.
5. Coello Coello, Carlos A. Christiansen, Alan D. (2000). *Multiobjective optimization of trusses using genetic algorithms*. Computers and Structures. Volume 75, issue 6. Pages 647-660.
6. Coello Coello, Carlos A. Rudnik, Michael. Christiansen, Alan D. (1994). *Using genetic algorithms for optimal design of trusses*. Proceedings of the Sixth International Conference on Tools with Artificial Intelligence. Pages 88-94.
7. Deb, Kalyanmoy. (1997). *Genetic Algorithm in Search and Optimization: The Technique and Applications*. Proceedings of the International Workshop on Soft Computing and Intelligent Systems. Pages 58-87.
8. Deb, Kalyanmoy. Gulati, Surendra. (2000). *Design of Truss-Structures for Minimum Weight using Genetic Algorithms*. Finite Elements in Analysis and Design. Volume 37, issue 5. Pages 447-465.
9. EUROPEAN COMMITTEE FOR STANDARDISATION. *Eurocode 3: Design of steel structures – Part 1-1: General rules and rules for buildings*. (2005). Brussels: CEN.
10. EUROPEAN COMMITTEE FOR STANDARDISATION. *Eurocode 3: Design of steel structures – Part 1.5: Plated structural elements* (2003). Brussels: CEN.
11. Galante, Miguel. (1996). *GENETIC ALGORITHMS AS AN APPROACH TO OPTIMIZE REAL-WORLD TRUSSES*. International Journal for Numerical Methods in Engineering. Volume 39, issue 3. Pages 361-382.
12. *Genetic Algorithm and Direct Search Toolbox 2.4.2*. (2009). The MathWorks. Available: www.mathworks.com. Last accessed 2009-10-25.
13. Goldberg, David Edward. (1989). *Genetic algorithms in search, optimization and machine learning*. New York: Addison-Wesley.

14. Guerlement, G. Targowski, R. Gutkowski, W Zawidzka, J. Zawidzki J. (2001). *Discrete minimum weight design of steel structures using EC3 code*. Structural and Multidisciplinary Optimization. Volume 22, issue 4. Pages 322-327.
15. Hajela, P. Lee, E. (1995). *GENETIC ALGORITHMS IN TRUSS TOPOLOGICAL OPTIMIZATION*. International Journal of Solids and Structures. Volume 32, issue 22. Pages 3341-3357.
16. Heyden, Susanne. Dahlbom, Ola. Olsson, Anders. Sandberg, Göran. (2005). *INTRODUKTION TILL STRUKTURMEKANIKEN*. Lund: KFS i Lund AB.
17. Kaveh, A. Kalatjari, V. (2004). *Size/geometry optimization of trusses by the force method and genetic algorithm*. ZAMM – Zeitschrift für Angewandte Mathematik und Mechanik. Volume 84, issue 5. Pages 347-357.
18. Kawamura, H. Ohmori, H. Kito N. (2002). *Truss topology optimization by a modified genetic algorithm*. Structural and Multidisciplinary Optimization. Volume 23, issue 6. Pages 467-473.
19. Kripakaran, Prakash. Gupta, Abhinav. Baugh Jr. John W. (2007). *A novel optimization approach for minimum cost design of trusses*. Computers & Structures. Volume 85, issue 23-24. Pages 1782-1794.
20. Lamberti, L. (2008). *An efficient simulated annealing algorithm for design optimization of truss structures*. Computers & Structures. Volume 86, issue 19-20. Pages 1936-1953.
21. Li, Lijuan. Huang, Zhibin. Liu, Feng. (2006). *An Improved Particle Swarm Optimizer for Truss Structure Optimization*. 2006 International Conference on Computational Intelligence and Security. Volume 1. Pages 924-928.
22. Li, Na. Ye, Feng. (2006). *Optimal Design of Discrete Dstructure with Directed Mutation Genetic Algorithms*. 2006 6th World Congress on Intelligent Control and Automation. Volume 1. Pages 3663-3667.
23. Ohsaki, M. Katoh N. (2005) *Topology optimization of trusses with stress and local constraints on nodal stability and member intersection*. Structural and Multidisciplinary Optimization. Volume 29, issue 3. Pages 190-197.
24. Reeves, Colin R. Rowe Jonathan E. (2002). *Genetic Algorithms: Principles and Perspectives – A Guide to GA Theory*. New York: Springer US.
25. Rothlauf, Franz. (2006). *Representations for Genetic and Evolutionary Algorithms*. Berlin: Springer.
26. Sivanandam S. N. Deepa S. N. (2008). *Introduction to Genetic Algorithms*. Berlin: Springer.
27. Soh, Che-Kiong. Yang, Jiaping. (1998). *Optimal Layout of Bridge Trusses by Genetic Algorithms*. Computer-Aided Civil and Infrastructure Engineering. Volume 13, issue 4. Pages 247-254.

28. *Square hollow profiles*. Department of Structural Engineering. Budapest University of Technology and Economics. Available: www.hsz.bme.hu. Last accessed 2009-10-29.

29. Toğan, Vedat. Daloglu, Ayşe T. (2006). *Optimization of 3d trusses with adaptive approach in genetic algorithms*. Engineering Structures. Volume 28, issue 7. Pages 1019-1027.

30. Tsai, Lung-Wen. *Mechanism design: enumeration of kinematic structures according to function*. (2001). New York: CRC Press.

31. Wu, Shyue-Jian. Chow Pei-Tse. (1995). *STEADY-STATE GENETIC ALGORITHMS FOR DISCRETE OPTIMIZATION OF TRUSSES*. Computers & Structures. Volume 56, issue 6. Pages 979-991.

7 Appendices

A) Optimization algorithm m-files

```
function GAmodule

%-----
% PURPOSE:
%   To calculate a weight optimized steel truss of the given conditions
%   according to Eurocode 3: Design of steel structures.
%-----

%-----
% Genetic algorithm settings:
Generations=500;
Mutation=0.05;
%-----

format short g
close all
clc

sq=input('Please specify which steel quality You intend to use [N/mm^2]
(235/275/355/420/460)! ');
nb=input('Please specify the number of support nodes! ');
nl=input('Please specify the number of nodes affected by a load! ');
nn=input('Please specify the total number of nodes! ');

fixedcoord=zeros(nl+nb,2);
for i=1:nb
    fixedcoord(i,1)=input(['Please specify the X-coordinate of support node
',num2str(i),'! (dm) ']);
    fixedcoord(i,2)=input(['Please specify the Y-coordinate of support node
',num2str(i),'! (dm) ']);
end

for i=nb+1:nb+nl
    fixedcoord(i,1)=input(['Please specify the X-coordinate of load node
',num2str(i-nb),'! (dm) ']);
    fixedcoord(i,2)=input(['Please specify the Y-coordinate of load node
',num2str(i-nb),'! (dm) ']);
end

xspan=max(fixedcoord(:,1))-min(fixedcoord(:,1));
yspan=max(fixedcoord(:,2))-min(fixedcoord(:,2));

if nn<nb+nl
    nn=nb+nl;
end

f=zeros(2*nn,1);
for i=nb+1:nb+nl
```

```

    f(2*i-1)=input(['For load node ',num2str(i-nb),' please specify the X-
resultant of the load! (N) ']);
    f(2*i)=input(['For load node ',num2str(i-nb),' please specify the Y-
resultant of the load! (N) ']);
end

```

```

Dof=[1:2:2*nn-1;2:2:2*nn]';
bc=[1:2*nb;zeros(1,2*nb)]';

```

```

Top=input('Please select method for topology optimization! (Press g for
ground structure method, r for reduced method) ','s');

```

```

if Top=='g'
    neft=factorial(nn)/(2*factorial(nn-2));
    nevt=0;
    A=[];B=[];TOPGSM=[];
    for i=1:nn-1
        A=[A,i*ones(1,nn-i)];
        B=[B,(i+1:nn)];
    end
    for i=1:neft
        TOPGSM=[TOPGSM;i Dof(A(i),:) Dof(B(i),:)];
    end
else
    TOPGSM=0;
    neft=2*nn-3;
    maxnevt=factorial(nn)/(2*factorial(nn-2))-neft;
    nevt=input(['Please specify the number of elements with variable
topology (0-',num2str(maxnevt),' ',num2str(round(maxnevt/10)),'
recommended! ']);
end

```

```

Population=input('Please specify the size of the initial population (150-
1000 recomend)! ');

```

```

disp(' ')
disp('Calculating...')

```

```

tic
% Fabricational dimensions (in SI units):
% Element thickness:
t=1e-3*[2.5 3.0 3.2 4.0 4.9 5.0 2.5 3.0 3.2 4.0 4.9 5.0 6.0 6.3 3.0 3.2 4.0
4.9 5.0 6.0 6.3 8.0 3.0 3.2 3.6 4.0 4.9 5.0 6.0 6.3 7.1 8.0 3.2 3.6 4.0 4.9
5.0 6.0 6.3 7.1 8.0 3.2 3.6 4.0 4.9 5.0 5.6 6.0 6.3 7.1 8.0 3.6 4.0 4.9 5.0
5.6 6.0 6.3 7.1 8.0 3.6 4.0 4.9 5.0 5.6 6.0 6.3 7.1 8.0 10.0 4.0 4.9 5.0
5.6 6.0 6.3 7.1 8.0 8.8 10.0 12.0 12.5 4.9 5.0 5.6 6.0 6.3 7.1 8.0 8.8 10.0
12.0 12.5 4.9 5.0 5.6 6.0 6.3 7.1 8.0 8.8 10.0 12.0 12.5 16.0 5.0 5.6 6.0
6.3 7.1 8.0 8.8 10.0 12.0 12.5 14.2 16.0 5.0 5.6 6.0 6.3 7.1 8.0 8.8 10.0
12.0 12.5 14.2 16.0 5.0 5.6 6.0 6.3 7.1 8.0 8.8 10.0 12.0 12.5 14.2 16.0
5.0 5.6 6.0 6.3 7.1 8.0 8.8 10.0 12.0 12.5 14.2 16.0 6.0 6.3 7.1 8.0 8.8
10.0 12.0 12.5 14.2 16.0 6.0 6.3 7.1 8.0 8.8 10.0 12.0 12.5 14.2 16.0 8.0
8.8 10.0 12.0 12.5 14.2 16.0 8.0 8.8 10.0 12.0 12.5 14.2 16.0 20.0 19.0
22.0 25.0 22.0 25.0 12.0 16.0 19.0 22.0 25.0 28.0 32.0 12.0 16.0 19.0 22.0
25.0 28.0 32.0 36.0 16.0 19.0 22.0 25.0 28.0 32.0 36.0 40.0 25.0 28.0 32.0
36.0 40.0 25.0 28.0 32.0 36.0 40.0];

```

```

% Cross-sectional width:
w=1e-3*[40 40 40 40 40 40 50 50 50 50 50 50 50 50 60 60 60 60 60 60 60 60 60
70 70 70 70 70 70 70 70 70 70 70 76.2 76.2 76.2 76.2 76.2 76.2 76.2 76.2 76.2
80 80 80 80 80 80 80 80 80 80 80 90 90 90 90 90 90 90 90 90 90 100 100 100 100
100 100 100 100 100 100 120 120 120 120 120 120 120 120 120 120 120 120 120 120 140
140 140 140 140 140 140 140 140 140 140 140 150 150 150 150 150 150 150 150 150 150
150 150 150 160 160 160 160 160 160 160 160 160 160 160 160 160 160 160 180 180 180 180
180 180 180 180 180 180 180 180 180 180 200 200 200 200 200 200 200 200 200 200 200
200 250 250 250 250 250 250 250 250 250 250 250 250 250 250 250 250 250 260 260 260 260
260 260 260 260 300 300 300 300 300 300 300 300 300 300 300 300 300 300 350 350 350 350
350 350 400 400 400 400 400 400 400 400 400 350 350 350 400 400 400 450 450 450 450
450 450 450 500 500 500 500 500 500 500 500 500 550 550 550 550 550 550 550 550
600 600 600 600 600 700 700 700 700 700];

% Moment of inertia:
I=1e-8*[8.54 9.78 10.20 11.80 13.20 13.40 17.50 20.20 21.20 25.00 28.50
28.90 32.00 32.80 36.20 38.20 45.40 52.50 53.30 59.90 61.60 69.70 59.00
62.30 68.60 74.70 87.20 88.50 101.00 104.00 112.00 120.00 81.50 89.90 98.00
115.00 117.00 133.00 138.00 149.00 160.00 95.00 105.00 114.00 135.00 137.00
149.00 156.00 162.00 176.00 189.00 152.00 166.00 196.00 200.00 218.00
230.00 238.00 260.00 281.00 212.00 232.00 275.00 279.00 306.00 323.00
336.00 367.00 400.00 462.00 410.00 489.00 498.00 547.00 579.00 603.00
663.00 726.00 779.00 852.00 958.00 982.00 793.00 807.00 891.00 944.00
984.00 1086.00 1195.00 1287.00 1416.00 1609.00 1653.00 984.00 1002.00
1106.00 1174.00 1223.00 1352.00 1491.00 1608.00 1773.00 2023.00 2080.00
2430.00 1225.00 1353.00 1437.00 1499.00 1659.00 1831.00 1978.00 2186.00
2502.00 2576.00 2809.00 3028.00 1765.00 1954.00 2077.00 2168.00 2404.00
2661.00 2880.00 3193.00 3677.00 3790.00 4154.00 4504.00 2445.00 2710.00
2883.00 3011.00 3345.00 3709.00 4021.00 4471.00 5171.00 5336.00 5872.00
6394.00 4861.00 5399.00 5752.00 6014.00 6701.00 7455.00 8107.00 9055.00
10556.00 10915.00 12094.00 13267.00 6491.00 6788.00 7567.00 8423.00 9164.00
10242.00 11954.00 12365.00 13714.00 15061.00 10080.00 10547.00 11775.00
13128.00 14305.00 16026.00 18777.00 19442.00 21637.00 23850.00 21129.00
23055.00 25884.00 30435.00 31541.00 35211.00 38942.00 31857.00 34798.00
39128.00 46130.00 47839.00 53526.00 59344.00 71535.00 43360.00 48360.00
52890.00 74710.00 82150.00 65430.00 84070.00 97060.00 109200.00 120600.00
131200.00 144100.00 90750.00 117100.00 135500.00 153000.00 169400.00
184900.00 204000.00 221500.00 157700.00 183000.00 207100.00 230000.00
251600.00 278600.00 303500.00 326500.00 303400.00 332700.00 369400.00
403700.00 435500.00 494100.00 543500.00 606200.00 665400.00 721200.00];

% c:
c=w-2*t;
% Cross-sectional area:
A=w.^2.-c.^2;

% available=[A(1:3:150)' c(1:3:150)' t(1:3:150)' I(1:3:150)'];
available=[A' c' t' I'];
zerobar=[eps*ones(30,1) zeros(30,3)];
available=[zerobar;available];

% Number of available elements:
na=size(available,1);

% Number of bits required in the different segments of the bit string:
numbitEp=ceil(log2(na));

if nn == nb+nl
    numbitCoordY=0;
    numbitCoordX=0;

```

```

else
    numbitCoordX=ceil(log2(xspan));
    numbitCoordY=ceil(log2(yspan));
end

if Top == 'g'
    numbitEdof=0;
else
    numbitEdof=ceil(log2(nn));
end

% Creating an initial population:
Initpop=round(rand(Population,(nn-nb-
nl)*(numbitCoordX+numbitCoordY)+(neft+nevt)*numbitEp+2*nevt*numbitEdof));

options = gaoptimset('PopulationType', 'bitString',...
    'FitnessLimit',0,...
    'InitialPopulation', Initpop,...
    'PlotFcns', {@gaplotbestf2},...
    'Generations', Generations,...
    'PopulationSize', Population,...
    'StallGenLimit', Inf,...
    'StallTimeLimit', Inf,...
    'SelectionFcn', @selectiontournament,...
    'FitnessScalingFcn', @fitscalingrank,...
    'EliteCount', 2,...
    'CrossoverFraction', 0.9,...
    'CrossoverFcn', @crossoverscattered,...
    'MutationFcn', {@mutationuniform, Mutation}...
);

E=210e9;
[x,fval]=ga(@ (x)
penalty2(x,available,bc,Dof,E,f,fixedcoord,na,nb,neft,nevt,nl,nn,numbitCoordX,numbitCoordY,numbitEdof,numbitEp,sq,TOPGSM,xspan,yspan),(nn-nb-nl)*(numbitCoordX+numbitCoordY)+(neft+nevt)*numbitEp+2*nevt*numbitEdof,options);
Time=toc;

[Coord,Edof,Ep]=bintranslate(available,x,Dof,fixedcoord,na,nb,neft,nevt,nl,nn,numbitCoordX,numbitCoordY,numbitEdof,numbitEp,xspan,yspan);
if Edof==0;
    Edof=TOPGSM;
end
[Ed,Ex,Ey,v1,v2,w,wrong]=FEM2(bc,Coord,Dof,E,Edof,Ep,f,neft,nevt,sq,TOPGSM);

if wrong==1
    disp('Unable to find a feasible solution!')
else

    % Removing zero-bars from the FEM parameters before the plot:
    for i=1:neft+nevt
        if v2(i,1)==0

```

```

        P(i)=0;
    else
        P(i)=i;
    end
end
J=find(P);

Ex2=zeros(length(J),2);Ey2=zeros(length(J),2);Ed2=zeros(length(J),4);Edof2=
zeros(length(J),5);Ep2=zeros(length(J),4);v22=zeros(length(J),2);
for i=1:length(J)
    Ex2(i,:)=Ex(J(i),:);
    Ey2(i,:)=Ey(J(i),:);
    Ed2(i,:)=Ed(J(i),:);
    Edof2(i,:)=Edof(J(i),:);
    Ep2(i,:)=Ep(J(i),:);
    v22(i,:)=v2(J(i),:);
end

% Displaying the calculated truss:
figure
eldraw2(Ex2,Ey2,[1 3 1],Edof2)

axis([0 xspan/10+1 -1 yspan/10+1]);
legend(['Elapsed time: ',num2str(Time),' seconds; Fitness:
',num2str(fval),'; Actual weight: ',num2str(w),' Lbs.'])

figure
eldraw2(Ex2,Ey2,[1 3 1],Edof2)
axis([0 xspan/10+1 -1 yspan/10+1]);
[sfac]=scalfact2(Ex2,Ey2,Ed2,0.1);
eldisp2(Ex2,Ey2,Ed2,[2 1 1],sfac)
pltscalb2(sfac,[5e-2 10 8]);

% Printing the results:
disp(['Element no. ',' ', 'Dimensions in mm (w*w*t)', ' ', ' ', 'Start
coordinates(m)', ' ', ' ', 'End coordinates (m)', ' ', ' ', 'Stress in % of EC3'])
for i=1:length(J)
    disp([' ', num2str(J(i)), '
', num2str((Ep2(i,2)+2*Ep2(i,3))*1000), 'x', num2str((Ep2(i,2)+2*Ep2(i,3))*100
0), 'x', num2str(Ep2(i,3)*1000), '
', num2str(Ex2(i,1)), ' ', num2str(Ey2(i,1)), '
', num2str(Ex2(i,2)), ' ', num2str(Ey2(i,2)), '
', num2str(v22(i,1)/v22(i,2)*100)])
end
disp(' ')
disp(' ')
disp('Element stresses in MPa:')
for i=1:length(J)
    disp([num2str(J(i)), ' ', num2str((v22(i,1)/Ep2(i,1))/1e6)])
end
disp(' ')
disp(' ')
disp('Displacements:')
disp(['Horizontally', ' ', num2str(v1(1,1)*1000), ' ', ' ', 'mm', ' ',
', num2str(100*v1(1,1)/v1(1,2)), '% of the limit'])
disp(['Vertically', ' ', num2str(v1(2,1)*1000), ' ', ' ', 'mm', ' ',
', num2str(100*v1(2,1)/v1(2,2)), '% of the limit'])
disp(' ')

```

```
disp(' ')
disp(['Weight of the structure is ',num2str(w),'Lbs or
',num2str(w*0.45359237),'kg'])
end
%-----end-----
```

```

function
[fitness]=penalty2(chromosome,available,bc,Dof,E,f,fixedcoord,na,nb,neft,ne
vt,nl,nn,numbitCoordX,numbitCoordY,numbitEdof,numbitEp,sq,TOPGSM,xspan,yspa
n)

%-----
% PURPOSE:
%   To assign a penalty weight, proportional to the extent in which the
%   truss is violating the constraints and calculate the fitness of the
%   truss.
%
% INPUT:
%   chromosome   A binary string representing the truss.
%   available    A matrix containing the element properties of available
%               hot finished square profiles. Size(available)=na*4.
%   fixedcoord  A matrix containing the X-and Y-coordinates of the
%               fixed nodes. Size(fixedcoord)=(nb+nl)*2.
%   na          Number of available profiles.
%   nb          Number of basic nodes.
%   nl          Number of nodes affected by a load.
%   nn          Number of nodes.
%   numbitCoordX Number of bits required to represent all possible
%               positions in the X-direction with the current
%               precision.
%   numbitCoordY Number of bits required to represent all possible
%               positions in the Y-direction with the current
%               precision.
%   numbitEdof  Number of bits required to represent all possible
%               topologies of an element.
%   numbitEp    Number of bits required to represent the element
%               properties of an element.
%   sq          Steel quality in Newton per square millimetre.
%
% OUTPUT:
%   fitness     The fitness of the truss.
%-----

[Coord,Edof,Ep]=bintranslate(available,chromosome,Dof,fixedcoord,na,nb,neft
,nevt,nl,nn,numbitCoordX,numbitCoordY,numbitEdof,numbitEp,xspan,yspan);
if Edof==0;
    Edof=TOPGSM;
end
[Ed,Ex,Ey,v1,v2,w,wrong]=FEM2(bc,Coord,Dof,E,Edof,Ep,f,neft,nevt,sq,TOPGSM)
;

P=zeros(3+neft+nevt,1);
if wrong==1
    P(1)=1e20;
    fitness=w+sum(P);
    return
else
    P(1)=0;
    dispfactors=v1(:,1)./v1(:,2);
    stressfactors=v2(:,1)./v2(:,2);
    for i=1:2
        if dispfactors(i)>1
            P(i+1)=dispfactors(i)*1e8;
        else
            P(i+1)=0;
        end
    end
end

```



```
end
for i=1:neft+nevt
    if stressfactors(i)>1
        P(i+3)=stressfactors(i)*1e8;
    else
        P(i+3)=0;
    end
end
fitness=w+sum(P);
end
%-----end-----
```

```

function
[Coord,Edof,Ep]=bintranslate (available,chromosome,Dof,fixedcoord,na,nb,neft
,nevt,nl,nn,numbitCoordX,numbitCoordY,numbitEdof,numbitEp,xspan,yspan)

%-----
% PURPOSE:
%   To translate the binary chromosome string into FEM parameters.
%
% INPUT:
%   available      A matrix containing the element properties of available
%                  hot finished square profiles. Size(available)=na*4.
%   chromosome     A binary string representing the truss.
%   Dof            Degrees of freedom. Dof(i,:) are corresponding with
%                  Coord(i,:). Size(Dof)=nn*2.
%   fixedcoord     A matrix containing the X-and Y-coordinates of the
%                  fixed nodes. Size(fixedcoord)=(nb+nl)*2.
%   na             Number of available profiles.
%   nb             Number of basic nodes.
%   neft           Number of elements with fixed topology.
%   nevt           Number of elements with variable topology.
%   nl             Number of nodes affected by a load.
%   nn             Number of nodes.
%   numbitCoordX   Number of bits required to represent all possible
%                  positions in the X-direction with the current
%                  precision.
%   numbitCoordY   Number of bits required to represent all possible
%                  positions in the Y-direction with the current
%                  precision.
%   numbitEdof     Number of bits required to represent all possible
%                  topologies of an element.
%   numbitEp       Number of bits required to represent the element
%                  properties of an element.
%
% OUTPUT:
%   Coord          The X- and Y-coordinates for each node.
%                  Size(Coord)=nn*2.
%   Edof           Topology matrix. Describes to which degrees of freedom
%                  a bar is connected.
%   Ep             Element properties. [Area, inner width, thickness,
%                  moment of inertia]. Size(Ep)=(neft+nevt)*4.
%-----

format short

% Extracting nodal coordinates
if numbitCoordX==0
    Coord=1e-1*fixedcoord;
else
    chromosome1=chromosome(1:numbitCoordX*(nn-nb-nl));
    chromosome2=chromosome(1+numbitCoordX*(nn-nb-nl):numbitCoordX*(nn-nb-
nl)+numbitCoordY*(nn-nb-nl));

    variablecoord=zeros(nn-(nb+nl),2);
    for i=numbitCoordX:numbitCoordX:numbitCoordX*(nn-nb-nl)
        if round(bin2dec(num2str(chromosome1(i-(numbitCoordX-1):i))))>xspan
            variablecoord(i/numbitCoordX,1)=xspan-1;
        else
            variablecoord(i/numbitCoordX,1)=round(bin2dec(num2str(chromosome1(i-
(numbitCoordX-1):i)))));
        end
    end
end

```

```

        end
    end

    for i=numbitCoordY:numbitCoordY:numbitCoordY*(nn-nb-nl)
        if round(bin2dec(num2str(chromosome2(i-(numbitCoordY-1):i))))>yspan
            variablecoord(i/numbitCoordY,2)=yspan;
        else
            variablecoord(i/numbitCoordY,2)=round(bin2dec(num2str(chromosome2(i-
            (numbitCoordY-1):i)))));
        end
    end

    Coord=1e-1*[fixedcoord;variablecoord];
end

chromosome3=chromosome(1+(numbitCoordX+numbitCoordY)*(nn-nb-
nl):numbitEp*(neft+nevt)+(numbitCoordX+numbitCoordY)*(nn-nb-nl));
chromosome4=chromosome(1+length(chromosome)-
2*nevt*numbitEdof:length(chromosome)-nevt*numbitEdof);
chromosome5=chromosome(1+length(chromosome)-
nevt*numbitEdof:length(chromosome));

if numbitEdof == 0
    Edof=0;
else
    add=0.01*[1:50]';
    Coord1=Coord(:,1)+add(1:size(Coord,1));

    sortcoord=sort(Coord1);

    % Creating a basic structure
    A=zeros(neft,2);B=zeros(neft,2);
    A(neft,:)=Dof(find(Coord1==sortcoord(end-1),1),:);
    B(1,:)=Dof(find(Coord1==sortcoord(2),1),:);
    for i=2:2:neft-1
        A(i-1,:)=Dof(find(Coord1==sortcoord(i/2),1),:);
        A(i,:)=Dof(find(Coord1==sortcoord(i/2),1),:);
        B(i,:)=Dof(find(Coord1==sortcoord((i+4)/2),1),:);
        B(i+1,:)=Dof(find(Coord1==sortcoord((i+4)/2),1),:);
    end

    % Extracting the elements with variable topology
    if nevt==0
        Edof=[(1:neft)' A B];
    else
        C=zeros(nevt,2);D=zeros(nevt,2);
        for i=numbitEdof:numbitEdof:numbitEdof*nevt
            if ceil(bin2dec(num2str(chromosome4(i-(numbitEdof-1):i))))>nn
                C(i/numbitEdof,:)=Dof(nn,:);
            elseif ceil(bin2dec(num2str(chromosome4(i-(numbitEdof-
1):i))))==0
                C(i/numbitEdof,:)=Dof(1,:);
            else
                C(i/numbitEdof,:)=Dof(ceil(bin2dec(num2str(chromosome4(i-
(numbitEdof-1):i))))),:);
            end
            if ceil(bin2dec(num2str(chromosome5(i-(numbitEdof-1):i))))>nn
                D(i/numbitEdof,:)=Dof(nn,:);
            end
        end
    end
end

```

```

elseif ceil(bin2dec(num2str(chromosome5(i-(numbitEdof-
1):i))))==0
    D(i/numbitEdof,:)=Dof(1,:);
else
    D(i/numbitEdof,:)=Dof(ceil(bin2dec(num2str(chromosome5(i-
(numbitEdof-1):i))))),:);
end
end

Edof=[(1:neft)' A B;(1+neft:neft+nevt)' C D];
end
end

% Extracting the element properties
Ep=zeros(neft+nevt,4);
for i=numbitEp:numbitEp:(neft+nevt)*numbitEp
    if ceil(bin2dec(num2str(chromosome3(i-(numbitEp-1):i))))>na
        Ep(i/numbitEp,:)=available(na,:);
    elseif ceil(bin2dec(num2str(chromosome3(i-(numbitEp-1):i))))==0
        Ep(i/numbitEp,:)=available(1,:);
    else
        Ep(i/numbitEp,:)=available(ceil(bin2dec(num2str(chromosome3(i-
(numbitEp-1):i))))),:);
    end
end
end
%-----end-----

```

```

function
[Ed, Ex, Ey, v1, v2, w, wrong]=FEM2(bc, Coord, Dof, E, Edof, Ep, f, neft, nevt, sq, TOPGSM)

%-----
% PURPOSE:
%   To determine feasibility and calculate the element stresses and nodal
%   displacements along with the appropriate EC3 constraints.
%
% INPUT:
%   bc           Boundary conditions. Describes which degrees of freedom
%               that are given a prescribed displacement.
%   Coord       The X- and Y-coordinates for each node. Size(Coord)=nn*2.
%   Dof         Degrees of freedom. Dof(i,:) are corresponding with
%               Coord(i,:). Size(Dof)=nn*2.
%   E           Young's modulus. 210 GPa for steel.
%   Edof       Topology matrix. Describes to which degrees of freedom a
%               bar is connected.
%   Ep         Element properties. [Area, inner width, thickness, moment
%               of inertia]. Size(Ep)=(neft+nevt)*4.
%   f          Load vector. Size(f)=2nn*1.
%   neft       Number of elements with fixed topology.
%   nevt       Number of elements with variable topology.
%   sq         Steel quality in Newton per square millimetre.
%
% OUTPUT:
%   Ed         Element displacement matrix. Size(Ed)=(neft+nevt)*4.
%   Ex         X-coordinates for each element. Size(Ex)=(neft+nevt)*2.
%   Ey         Y-coordinates for each element. Size(Ey)=(neft+nevt)*2.
%   v1         Max displacement and displacement constraint in the X- and
%               Y-direction. Size(v1)=2*2.
%   v2         Level of stress and stress constraint for each element
%               size(v2)=(neft+nevt)*2.
%   w          the weight of the structure in Lbs.
%   wrong      1 or 0. If wrong == 1, the truss is not feasible.
%-----

density=7850;w=0;wrong=0;Ed=0;Ex=0;Ey=0;v1=0;v2=0;
ne=neft+nevt;

% element coordinates:
[Ex, Ey]=coordxtr(Edof, Coord, Dof, 2);

% Checking if double nodes exists:
Ucoord=unique(Coord, 'rows');
if size(Ucoord,1) ~= size(Coord,1)
    wrong=1;
    return
end

% Looking for elements with identical topology, and elements without
% any spatial extension:

if TOPGSM == 0
    PD=Edof(:,2).*Edof(:,4);ND=unique(PD);
    for i=1:ne
        if PD(i)==Edof(i,2)^2
            wrong=1;
            return
        end
    end
end

```

```

        end
    end

    if length(PD) ~= length(ND)
        wrong=1;
        return
    end
end

% Stiffness matrix of the structure:
K=zeros(length(f));
for i=1:ne
    Ke=bar2e(Ex(i,:),Ey(i,:),[E Ep(i,1)]);
    K=assem(Edof(i,:),K,Ke);
end

if rcond(K) == NaN
    wrong=1;
    return
end

% Stability check:
if abs(det(K))<1e-9
    wrong=1;
    return
end

% Calculating the element displacements:
[a,r]=solveq(K,f,bc);
Ed=extract(Edof,a);

% Calculating max displacements and limits and storing the info in v1:
% (All odd-numbered DOFs are x-directions)
Xdispl=[]; Ydispl=[];
for i=1:2:length(a)-1
    Xdispl=[Xdispl a(i)];
end
for i=2:2:length(a)
    Ydispl=[Ydispl a(i)];
end

% Lx (only valid for simple span structures):
Lx=max(max(Ex)); Ly=max(max(Ey)); v1=zeros(2,2);
v1(1,1)=max(abs([max(Xdispl) min(Xdispl)]));
v1(2,1)=max(abs([max(Ydispl) min(Ydispl)]));
% Horizontal limit
v1(1,2)=Ly/250;
% Vertical limit
v1(2,2)=Lx/250;

% Calculating cross-sectional class and effective area if needed:
epsilon=sqrt(235/sq); Ep2=zeros(ne,2); lambdap=zeros(ne,1); ro=zeros(ne,1);
for i=1:ne
    if Ep(i,1) == eps
        Ep2(i,:)=[0 123];
    else
        lambdap(i)=(Ep(i,2)/Ep(i,3))/(28.4*epsilon*2);
    end
end

```

```

        ro(i)=(lambdap(i)-0.055*4)/lambdap(i)^2;
        if ro(i)>1
            ro(i)=1;
        end
        if Ep(i,2)/Ep(i,3) <= 42*epsilon
            Ep2(i,1)=Ep(i,1);
            Ep2(i,2)=123;
        else
            Ep2(i,1)=Ep(i,1)*ro(i);
            Ep2(i,2)=4;
        end
    end
end

% Calculating the weight of the structure:
for i=1:ne
    % Calculating element length:
    Le(i)=sqrt((Ex(i,1)-Ex(i,2))^2+(Ey(i,1)-Ey(i,2))^2);
    w=w+Ep(i,1)*Le(i)*density/0.45359237;
end

% Calculating buckling stability:
if sq == 235 || sq == 275 || sq == 355 || sq == 420
    alpha=0.21;
else alpha=0.13;
end
Ncr=zeros(ne,1); Nbrd=zeros(ne,1); lambda=zeros(ne,1); Phi=zeros(ne,1);
Chi=zeros(ne,1); Ned=zeros(ne,1); v2=zeros(ne,2);
for i=1:ne
    % Critical load:
    Ncr(i)=pi^2*E*Ep(i,4)/Le(i)^2;
    if Ncr(i) == 0
        v2(i,:)=[0 1];
    else
        % Buckling reduction factor Chi:
        lambda(i)=sqrt((Ep2(i,1)*sq*1e6)/Ncr(i));
        Phi(i)=0.5*(1+alpha*(lambda(i)-0.2)+lambda(i)^2);
        Chi(i)=1/(Phi(i)+sqrt(Phi(i)^2-lambda(i)^2));
        if Chi(i)>1
            Chi(i)=1;
        end
        % Calculating element forces (tensile=positive) and constraints:
        Ned(i)=bar2s(Ex(i,:),Ey(i,:),[E Ep(i,1)],Ed(i,:));
        if lambda(i)<=0.2 || abs(Ned(i))/Ncr(i)<=0.04
            Nbrd(i)=Ep(i,1)*sq*1e6;
        else
            Nbrd(i)=Chi(i)*Ep2(i,1)*sq*1e6;
        end
        % Storing current forces and constraints in v2:
        if Ned(i)<0
            v2(i,:)=[Ned(i) -Nbrd(i)];
        else
            v2(i,:)=[Ned(i) Ep(i,1)*sq*1e6];
        end
    end
end
end
%-----end-----

```

B) List of available profiles [28]

Size	Thickness	Mass per Meter	Area of section	Moment of inertia	Radius of Inertia	Elastic section modulus	Plastic section modulus	Moment of gyration
B x B	T	M	A	I	i	W_{el}	W_{pl}	I_t
[mm]	[mm]	[kg/m]	[cm ²]	[cm ⁴]	[cm]	[cm ³]	[cm ³]	[cm ⁴]
40 x 40	2,5	2,89	3,68	8,54	1,52	4,27	5,14	13,60
40 x 40	3,0	3,41	4,34	9,78	1,50	4,89	5,97	15,70
40 x 40	3,2	3,61	4,60	10,20	1,49	5,11	6,28	16,50
40 x 40	4,0	4,39	5,59	11,80	1,45	5,91	7,44	19,50
40 x 40	4,9	5,20	6,62	13,20	1,41	6,62	8,55	22,20
40 x 40	5,0	5,28	6,73	13,40	1,41	6,68	8,66	22,50
50 x 50	2,5	3,68	4,68	17,50	1,93	6,99	8,29	27,50
50 x 50	3,0	4,35	5,54	20,20	1,91	8,08	9,70	32,10
50 x 50	3,2	4,62	5,88	21,20	1,90	8,49	10,20	33,80
50 x 50	4,0	5,64	7,19	25,00	1,86	9,99	12,30	40,40
50 x 50	4,9	6,74	8,58	28,50	1,82	11,40	14,30	46,90
50 x 50	5,0	6,85	8,73	28,90	1,82	11,60	14,50	47,60
50 x 50	6,0	7,99	10,20	32,00	1,77	12,80	16,50	53,60
50 x 50	6,3	8,31	10,60	32,80	1,76	13,10	17,00	55,20
60 x 60	3,0	5,29	6,74	36,20	2,32	12,10	14,30	56,90
60 x 60	3,2	5,62	7,16	38,20	2,31	12,70	15,20	60,20
60 x 60	4,0	6,90	8,79	45,40	2,27	15,10	18,30	72,50
60 x 60	4,9	8,28	10,50	52,50	2,23	17,50	21,60	85,10
60 x 60	5,0	8,42	10,70	53,30	2,23	17,80	21,90	86,40
60 x 60	6,0	9,87	12,60	59,90	2,18	20,00	25,10	98,60
60 x 60	6,3	10,30	13,10	61,60	2,17	20,50	26,00	102,00
60 x 60	8,0	12,50	16,00	69,70	2,09	23,20	30,40	118,00
70 x 70	3,0	6,24	7,94	59,00	2,73	16,90	19,90	92,20
70 x 70	3,2	6,63	8,44	62,30	2,72	17,80	21,00	97,60
70 x 70	3,6	7,40	9,42	68,60	2,70	19,60	23,30	108,00
70 x 70	4,0	8,15	10,40	74,70	2,68	21,30	25,50	118,00
70 x 70	4,9	9,81	12,50	87,20	2,64	24,90	30,30	140,00
70 x 70	5,0	9,99	12,70	88,50	2,64	25,30	30,80	142,00
70 x 70	6,0	11,80	15,00	101,00	2,59	28,70	35,50	163,00
70 x 70	6,3	12,30	15,60	104,00	2,58	29,70	36,90	169,00
70 x 70	7,1	13,60	17,30	112,00	2,54	32,00	40,30	185,00
70 x 70	8,0	15,00	19,20	120,00	2,50	34,20	43,80	200,00
76,2 x 76,2	3,2	7,25	9,23	81,50	2,97	21,40	25,20	127,00
76,2 x 76,2	3,6	8,10	10,30	89,90	2,95	23,60	27,90	141,00
76,2 x 76,2	4,0	8,93	11,40	98,00	2,93	25,70	30,60	154,00
76,2 x 76,2	4,9	10,80	13,70	115,00	2,89	30,20	36,40	183,00
76,2 x 76,2	5,0	11,00	14,00	117,00	2,89	30,60	37,00	186,00
76,2 x 76,2	6,0	12,90	16,50	133,00	2,85	35,00	42,90	215,00
76,2 x 76,2	6,3	13,50	17,20	138,00	2,83	36,20	44,60	223,00
76,2 x 76,2	7,1	15,00	19,10	149,00	2,80	39,10	48,80	244,00
76,2 x 76,2	8,0	16,60	21,10	160,00	2,75	42,10	53,20	265,00
80 x 80	3,2	7,63	9,72	95,00	3,13	23,70	27,90	148,00
80 x 80	3,6	8,53	10,90	105,00	3,11	26,20	31,00	164,00
80 x 80	4,0	9,41	12,00	114,00	3,09	28,60	34,00	180,00
80 x 80	4,9	11,40	14,50	135,00	3,05	33,60	40,40	214,00
80 x 80	5,0	11,60	14,70	137,00	3,05	34,20	41,10	217,00
80 x 80	5,6	12,80	16,30	149,00	3,02	37,20	45,20	238,00
80 x 80	6,0	13,60	17,40	156,00	3,00	39,10	47,80	252,00
80 x 80	6,3	14,20	18,10	162,00	2,99	40,50	49,70	262,00
80 x 80	7,1	15,80	20,20	176,00	2,95	43,90	54,50	286,00
80 x 80	8,0	17,50	22,40	189,00	2,91	47,30	59,50	312,00

90 x 90	3,6	9,66	12,30	152,00	3,52	33,80	39,70	237,00
90 x 90	4,0	10,70	13,60	166,00	3,50	37,00	43,60	260,00
90 x 90	4,9	12,90	16,40	196,00	3,46	43,60	52,10	310,00
90 x 90	5,0	13,10	16,70	200,00	3,45	44,40	53,00	316,00
90 x 90	5,6	14,60	18,60	218,00	3,43	48,50	58,30	347,00
90 x 90	6,0	15,50	19,80	230,00	3,41	51,10	61,80	367,00
90 x 90	6,3	16,20	20,70	238,00	3,40	53,00	64,30	382,00
90 x 90	7,1	18,10	23,00	260,00	3,36	57,70	70,80	419,00
90 x 90	8,0	20,10	25,60	281,00	3,32	62,60	77,60	459,00
100 x 100	3,6	10,80	13,70	212,00	3,92	42,30	49,50	328,00
100 x 100	4,0	11,90	15,20	232,00	3,91	46,40	54,40	361,00
100 x 100	4,9	14,40	18,40	275,00	3,87	55,00	65,20	432,00
100 x 100	5,0	14,70	18,70	279,00	3,86	55,90	66,40	439,00
100 x 100	5,6	16,30	20,80	306,00	3,84	61,20	73,20	484,00
100 x 100	6,0	17,40	22,20	323,00	3,82	64,60	77,60	513,00
100 x 100	6,3	18,20	23,20	336,00	3,80	67,10	80,90	534,00
100 x 100	7,1	20,30	25,80	367,00	3,77	73,40	89,20	589,00
100 x 100	8,0	22,60	28,80	400,00	3,73	79,90	98,20	646,00
100 x 100	10,0	27,40	34,90	462,00	3,64	92,40	116,00	761,00
120 x 120	4,0	14,40	18,40	410,00	4,72	68,40	79,70	635,00
120 x 120	4,9	17,50	22,30	489,00	4,68	81,50	95,80	763,00
120 x 120	5,0	17,80	22,70	498,00	4,68	83,00	97,60	777,00
120 x 120	5,6	19,90	25,30	547,00	4,65	91,20	108,00	858,00
120 x 120	6,0	21,20	27,00	579,00	4,63	96,60	115,00	911,00
120 x 120	6,3	22,20	28,20	603,00	4,62	100,00	120,00	950,00
120 x 120	7,1	24,70	31,50	663,00	4,59	110,00	133,00	1 051,00
120 x 120	8,0	27,60	35,20	726,00	4,55	121,00	146,00	1 160,00
120 x 120	8,8	30,10	38,30	779,00	4,51	130,00	158,00	1 252,00
120 x 120	10,0	33,70	42,90	852,00	4,46	142,00	175,00	1 382,00
120 x 120	12,0	39,50	50,30	958,00	4,36	160,00	201,00	1 578,00
120 x 120	12,5	40,90	52,10	982,00	4,34	164,00	207,00	1 623,00
140 x 140	4,9	20,60	26,20	793,00	5,50	113,00	132,00	1 230,00
140 x 140	5,0	21,00	26,70	807,00	5,50	115,00	135,00	1 253,00
140 x 140	5,6	23,40	29,80	891,00	5,47	127,00	149,00	1 387,00
140 x 140	6,0	24,90	31,80	944,00	5,45	135,00	159,00	1 475,00
140 x 140	6,3	26,10	33,30	984,00	5,44	141,00	166,00	1 540,00
140 x 140	7,1	29,20	37,20	1 086,00	5,40	155,00	184,00	1 709,00
140 x 140	8,0	32,60	41,60	1 195,00	5,36	171,00	204,00	1 892,00
140 x 140	8,8	35,60	45,40	1 287,00	5,33	184,00	221,00	2 048,00
140 x 140	10,0	40,00	50,90	1 416,00	5,27	202,00	246,00	2 272,00
140 x 140	12,0	47,00	59,90	1 609,00	5,18	230,00	284,00	2 616,00
140 x 140	12,5	48,70	62,10	1 653,00	5,16	236,00	293,00	2 696,00
150 x 150	4,9	22,10	28,20	984,00	5,91	131,00	153,00	1 522,00
150 x 150	5,0	22,60	28,70	1 002,00	5,90	134,00	156,00	1 550,00
150 x 150	5,6	25,10	32,00	1 106,00	5,88	147,00	173,00	1 718,00
150 x 150	6,0	26,80	34,20	1 174,00	5,86	156,00	184,00	1 828,00
150 x 150	6,3	28,10	35,80	1 223,00	5,85	163,00	192,00	1 909,00
150 x 150	7,1	31,40	40,00	1 352,00	5,81	180,00	213,00	2 121,00
150 x 150	8,0	35,10	44,80	1 491,00	5,77	199,00	237,00	2 351,00
150 x 150	8,8	38,40	48,90	1 608,00	5,74	214,00	257,00	2 549,00
150 x 150	10,0	43,10	54,90	1 773,00	5,68	236,00	286,00	2 832,00
150 x 150	12,0	50,80	64,70	2 023,00	5,59	270,00	331,00	3 272,00
150 x 150	12,5	52,70	67,10	2 080,00	5,57	277,00	342,00	3 375,00
150 x 150	16,0	65,20	83,00	2 430,00	5,41	324,00	411,00	4 026,00
160 x 160	5,0	24,10	30,70	1 225,00	6,31	153,00	178,00	1 892,00
160 x 160	5,6	26,90	34,20	1 353,00	6,29	169,00	198,00	2 098,00
160 x 160	6,0	28,70	36,60	1 437,00	6,27	180,00	210,00	2 233,00
160 x 160	6,3	30,10	38,30	1 499,00	6,26	187,00	220,00	2 333,00
160 x 160	7,1	33,70	42,90	1 659,00	6,22	207,00	245,00	2 595,00

160 x 160	8,0	37,60	48,00	1 831,00	6,18	229,00	272,00	2 880,00
160 x 160	8,8	41,10	52,40	1 978,00	6,14	247,00	295,00	3 125,00
160 x 160	10,0	46,30	58,90	2 186,00	6,09	273,00	329,00	3 478,00
160 x 160	12,0	54,60	69,50	2 502,00	6,00	313,00	382,00	4 028,00
160 x 160	12,5	56,60	72,10	2 576,00	5,98	322,00	395,00	4 158,00
160 x 160	14,2	63,30	80,70	2 809,00	5,90	351,00	436,00	4 579,00
160 x 160	16,0	70,20	89,40	3 028,00	5,82	379,00	476,00	4 988,00
180 x 180	5,0	27,30	34,70	1 765,00	7,13	196,00	227,00	2 718,00
180 x 180	5,6	30,40	38,70	1 954,00	7,10	217,00	252,00	3 018,00
180 x 180	6,0	32,50	41,40	2 077,00	7,09	231,00	269,00	3 215,00
180 x 180	6,3	34,00	43,30	2 168,00	7,07	241,00	281,00	3 361,00
180 x 180	7,1	38,10	48,60	2 404,00	7,04	267,00	314,00	3 744,00
180 x 180	8,0	42,70	54,40	2 661,00	7,00	296,00	349,00	4 162,00
180 x 180	8,8	46,70	59,40	2 880,00	6,96	320,00	379,00	4 524,00
180 x 180	10,0	52,50	66,90	3 193,00	6,91	355,00	424,00	5 048,00
180 x 180	12,0	62,10	79,10	3 677,00	6,82	409,00	494,00	5 873,00
180 x 180	12,5	64,40	82,10	3 790,00	6,80	421,00	511,00	6 070,00
180 x 180	14,2	72,20	92,00	4 154,00	6,72	462,00	566,00	6 711,00
180 x 180	16,0	80,20	102,00	4 504,00	6,64	500,00	621,00	7 343,00
200 x 200	5,0	30,40	38,70	2 445,00	7,95	245,00	283,00	3 756,00
200 x 200	5,6	33,90	43,20	2 710,00	7,92	271,00	314,00	4 174,00
200 x 200	6,0	36,20	46,20	2 883,00	7,90	288,00	335,00	4 449,00
200 x 200	6,3	38,00	48,40	3 011,00	7,89	301,00	350,00	4 653,00
200 x 200	7,1	42,60	54,20	3 345,00	7,85	335,00	391,00	5 189,00
200 x 200	8,0	47,70	60,80	3 709,00	7,81	371,00	436,00	5 778,00
200 x 200	8,8	52,20	66,50	4 021,00	7,78	402,00	474,00	6 288,00
200 x 200	10,0	58,80	74,90	4 471,00	7,72	447,00	531,00	7 031,00
200 x 200	12,0	69,60	88,70	5 171,00	7,64	517,00	621,00	8 208,00
200 x 200	12,5	72,30	92,10	5 336,00	7,61	534,00	643,00	8 491,00
200 x 200	14,2	81,10	103,00	5 872,00	7,54	587,00	714,00	9 417,00
200 x 200	16,0	90,30	115,00	6 394,00	7,46	639,00	785,00	10 340,00
250 x 250	5,0	38,30	48,70	4 861,00	9,99	389,00	447,00	7 430,00
250 x 250	5,6	42,70	54,40	5 399,00	9,96	432,00	498,00	8 271,00
250 x 250	6,0	45,70	58,20	5 752,00	9,94	460,00	531,00	8 825,00
250 x 250	6,3	47,90	61,00	6 014,00	9,93	481,00	556,00	9 238,00
250 x 250	7,1	53,70	68,40	6 701,00	9,90	536,00	622,00	10 325,00
250 x 250	8,0	60,30	76,80	7 455,00	9,86	596,00	694,00	11 525,00
250 x 250	8,8	66,00	84,10	8 107,00	9,82	649,00	758,00	12 572,00
250 x 250	10,0	74,50	94,90	9 055,00	9,77	724,00	851,00	14 106,00
250 x 250	12,0	88,50	113,00	10 556,00	9,68	844,00	1 000,00	16 567,00
250 x 250	12,5	91,90	117,00	10 915,00	9,66	873,00	1 037,00	17 164,00
250 x 250	14,2	103,00	132,00	12 094,00	9,58	967,00	1 158,00	19 139,00
250 x 250	16,0	115,00	147,00	13 267,00	9,50	1 061,00	1 280,00	21 138,00
260 x 260	6,0	47,60	60,60	6 491,00	10,35	499,00	576,00	9 951,00
260 x 260	6,3	49,90	63,50	6 788,00	10,34	522,00	603,00	10 417,00
260 x 260	7,1	56,00	71,30	7 567,00	10,30	582,00	674,00	11 647,00
260 x 260	8,0	62,80	80,00	8 423,00	10,26	648,00	753,00	13 006,00
260 x 260	8,8	68,80	87,60	9 164,00	10,23	705,00	822,00	14 192,00
260 x 260	10,0	77,70	98,90	10 242,00	10,18	788,00	924,00	15 932,00
260 x 260	12,0	92,20	117,00	11 954,00	10,09	920,00	1 087,00	18 729,00
260 x 260	12,5	95,80	122,00	12 365,00	10,06	951,00	1 127,00	19 409,00
260 x 260	14,2	108,00	137,00	13 714,00	9,99	1 055,00	1 259,00	21 659,00
260 x 260	16,0	120,00	153,00	15 061,00	9,91	1 159,00	1 394,00	23 942,00
300 x 300	6,0	55,10	70,20	10 080,00	12,00	672,00	772,00	15 407,00
300 x 300	6,3	57,80	73,60	10 547,00	12,00	703,00	809,00	16 136,00
300 x 300	7,1	64,90	82,60	11 775,00	11,90	785,00	906,00	18 061,00
300 x 300	8,0	72,80	92,80	13 128,00	11,90	875,00	1 013,00	20 194,00
300 x 300	8,8	79,80	102,00	14 305,00	11,90	954,00	1 107,00	22 060,00
300 x 300	10,0	90,20	115,00	16 026,00	11,80	1 068,00	1 246,00	24 807,00

300 x 300	12,0	107,00	137,00	18 777,00	11,70	1 252,00	1 470,00	29 249,00
300 x 300	12,5	112,00	142,00	19 442,00	11,70	1 296,00	1 525,00	30 333,00
300 x 300	14,2	126,00	160,00	21 637,00	11,60	1 442,00	1 708,00	33 938,00
300 x 300	16,0	141,00	179,00	23 850,00	11,50	1 590,00	1 895,00	37 622,00
350 x 350	8,0	85,40	109,00	21 129,00	13,90	1 207,00	1 392,00	32 384,00
350 x 350	8,8	93,60	119,00	23 055,00	13,90	1 317,00	1 522,00	35 413,00
350 x 350	10,0	106,00	135,00	25 884,00	13,90	1 479,00	1 715,00	39 886,00
350 x 350	12,0	126,00	161,00	30 435,00	13,80	1 739,00	2 030,00	47 154,00
350 x 350	12,5	131,00	167,00	31 541,00	13,70	1 802,00	2 107,00	48 934,00
350 x 350	14,2	146,00	189,00	35 211,00	13,70	2 012,00	2 364,00	54 879,00
350 x 350	16,0	166,00	211,00	38 942,00	13,60	2 225,00	2 630,00	60 990,00
400 x 400	8,0	97,90	125,00	31 857,00	16,00	1 593,00	1 830,00	48 695,00
400 x 400	8,8	107,00	137,00	34 798,00	15,90	1 740,00	2 004,00	53 290,00
400 x 400	10,0	122,00	155,00	39 128,00	15,90	1 956,00	2 260,00	60 092,00
400 x 400	12,0	145,00	185,00	46 130,00	15,80	2 306,00	2 679,00	71 181,00
400 x 400	12,5	151,00	192,00	47 839,00	15,80	2 392,00	2 782,00	73 906,00
400 x 400	14,2	170,00	217,00	53 526,00	15,70	2 676,00	3 127,00	83 026,00
400 x 400	16,0	191,00	243,00	59 344,00	15,60	2 967,00	3 484,00	92 442,00
400 x 400	20,0	235,00	300,00	71 535,00	15,40	3 577,00	4 247,00	112 489,00
350 x 350	19,0	190,00	242,00	43 360,00	13,40	2 478,00	2 967,00	70 930,00
350 x 350	22,0	217,00	276,00	48 360,00	13,20	2 763,00	3 343,00	80 180,00
350 x 350	25,0	242,00	309,00	52 890,00	13,10	3 022,00	3 695,00	88 880,00
400 x 400	22,0	251,00	320,00	74 710,00	15,30	3 735,00	4 476,00	122 400,00
400 x 400	25,0	282,00	359,00	82 150,00	15,10	4 108,00	4 967,00	136 100,00
450 x 450	12,0	162,00	207,00	65 430,00	17,80	2 908,00	3 371,00	102 400,00
450 x 450	16,0	213,00	271,00	84 070,00	17,60	3 736,00	4 376,00	133 500,00
450 x 450	19,0	250,00	318,00	97 060,00	17,50	4 314,00	5 092,00	155 800,00
450 x 450	22,0	286,00	364,00	109 200,00	17,30	4 854,00	5 775,00	177 200,00
450 x 450	25,0	321,00	409,00	120 600,00	17,20	5 359,00	6 427,00	197 700,00
450 x 450	28,0	355,00	452,00	131 200,00	17,00	5 830,00	7 047,00	217 300,00
450 x 450	32,0	399,00	509,00	144 100,00	16,80	6 404,00	7 826,00	242 000,00
500 x 500	12,0	181,00	231,00	90 750,00	19,80	3 630,00	4 196,00	141 500,00
500 x 500	16,0	238,00	303,00	117 100,00	19,60	4 682,00	5 461,00	184 800,00
500 x 500	19,0	280,00	356,00	135 500,00	19,50	5 422,00	6 368,00	216 100,00
500 x 500	22,0	320,00	408,00	153 000,00	19,40	6 120,00	7 239,00	246 200,00
500 x 500	25,0	360,00	459,00	169 400,00	19,20	6 778,00	8 074,00	275 300,00
500 x 500	28,0	399,00	508,00	184 900,00	19,10	7 396,00	8 874,00	303 300,00
500 x 500	32,0	450,00	573,00	204 000,00	18,90	8 161,00	9 886,00	338 900,00
500 x 500	36,0	498,00	635,00	221 500,00	18,70	8 860,00	10 840,00	372 500,00
550 x 550	16,0	263,00	335,00	157 700,00	21,70	5 734,00	6 666,00	247 900,00
550 x 550	19,0	309,00	394,00	183 000,00	21,50	6 656,00	7 787,00	290 200,00
550 x 550	22,0	355,00	452,00	207 100,00	21,40	7 532,00	8 868,00	331 300,00
550 x 550	25,0	399,00	509,00	230 000,00	21,30	8 362,00	9 909,00	371 000,00
550 x 550	28,0	443,00	564,00	251 600,00	21,10	9 149,00	10 910,00	409 400,00
550 x 550	32,0	500,00	637,00	278 600,00	20,90	10 130,00	12 190,00	458 600,00
550 x 550	36,0	555,00	707,00	303 500,00	20,70	11 040,00	13 400,00	505 400,00
550 x 550	40,0	608,00	775,00	326 500,00	20,50	11 870,00	14 540,00	549 800,00

600 x 600	25,0	439,00	559,00	303 400,00	23,30	10 110,00	11 930,00	486 600,00
600 x 600	28,0	487,00	620,00	332 700,00	23,20	11 090,00	13 160,00	537 700,00
600 x 600	32,0	550,00	701,00	369 400,00	23,00	12 310,00	14 730,00	603 400,00
600 x 600	36,0	611,00	779,00	403 700,00	22,80	13 460,00	16 220,00	666 400,00
600 x 600	40,0	671,00	855,00	435 500,00	22,60	14 520,00	17 640,00	726 400,00
700 x 700	25,0	517,00	659,00	494 100,00	27,40	14 110,00	16 540,00	784 900,00
700 x 700	28,0	575,00	732,00	543 500,00	27,20	15 530,00	18 280,00	869 200,00
700 x 700	32,0	651,00	829,00	606 200,00	27,00	17 320,00	20 530,00	978 300,00
700 x 700	36,0	724,00	923,00	665 400,00	26,90	19 010,00	22 690,00	1 083 700,00
700 x 700	40,0	797,00	1 015,00	721 200,00	26,70	20 600,00	24 760,00	1 183 200,00