preface

foreword

In one and a half year – starting from August 2005 – the idea of creating a structural design optimisation tool grew to the realisation of a script, written in VBA and using AutoCAD 2007 as a visualisation tool, which has the ability to optimise the allocation of a variable number of columns in a structural transfer zone. An extended preliminary study is executed in (computational) structural optimisation – book two –, and aspects and characteristics of transfer zones, modular dimensions, and structural grid layouts are discussed – book one. Based on the knowledge of optimisation methods gained from the preliminary study phase and the research stage, a design tool, intended to be used in the conceptual and preliminary design stage, is written, resulting in a VBA-script of over 4000 lines. A test structure is presented in the addendum, including the determination of the optimal GA operator parameters.

This book is part of an integral report containing all findings of the Master's thesis on structural optimisation for the Delft University of Technology, Faculty of Civil Engineering and Geosciences, department of Building Engineering, and the Structural Design Lab (SDL). Two books including an addendum form the complete and final report of this Master's thesis.

book one: **transfer zones in multi-use buildings and the development and analysis of the optimisation tool**
book two: **background studies on structural optimisation**
addendum: **determination of the optimal GA operator parameters and the presentation of test results based on example structures**

The reader of the complete report will find that the emphasis is mainly on the study into genetic algorithms and the onset of the design optimisation tool.

[It needs to be mentioned that in this Master's thesis the adjective 'structural' will be used to name aspects considering building engineering and structural engineering matters, rather than structural aspects in general]

Delft, January 2007

R.J. (Roel) van de Straat

graduation committee

name:	**prof. dipl.-ing. J.N.J.A. Vamberský** – chairman of the committee
university:	Delft University of Technology
	Faculty of Civil Engineering and Geosciences
address:	Stevinweg 1
	2628 CN Delft, room 1.35 St-II
	The Netherlands
telephone:	+31152783174 (secretariat)
e-mail:	j.n.j.a.vambersky@citg.tudelft.nl

name:	**ir. J.L. Coenders** – daily supervisor
university:	Delft University of Technology
	Faculty of Civil Engineering and Geosciences
address:	Stevinweg 1
	2628 CN Delft, room 1.58 St-II
	The Netherlands
telephone:	+31152785711
e-mail:	j.l.coenders@citg.tudelft.nl

name:	**ir. J.W. Welleman**
university:	Delft University of Technology
	Faculty of Civil Engineering and Geosciences
address:	Stevinweg 1
	2628 CN Delft, room 6.15
	The Netherlands
telephone:	+31152784856
e-mail:	j.w.welleman@citg.tudelft.nl

name:	**ir. S. Boer**
organisation:	ONL [Oosterhuis_Lénárd] (till March 2006) and Mecanoo architecten
home address:	Goudsesingel 464
	3011 KP Rotterdam
telephone:	+31648082965
e-mail:	sanderboer@myrealbox.com

consultant

name:	**dipl. inform. F. Scheurer**
university:	Eidgenössische Technische Hochschule Zürich
	Faculty of Architecture, Chair of CAAD
address:	Hoenggerberg HIL E15.1
	CH-8093 Zürich
telephone:	+41446334025
e-mail:	scheurer@arch.ethz.ch

acknowledgements

Many people helped in the realisation of this Master's thesis, and it is therefore underlined that, without their input and effort, the challenge of writing this report would be far greater. That is the reason why here those people are thanked wholeheartedly.

The graduation committee
in general is thanked for the positive input, the advice, and the criticism given as a group. Unanimity, consistency, and perseverance are characteristics the committee cannot be denied.
In particular, **prof. dipl.-ing. J.N.J.A. Vamberský**, for his ability to give an answer to a question just by listening, his great passion for the building engineering practise (and the way this is transferred to his students), and the eagerness to comprehend new techniques and methods shown by his students. **Ir. J.W. Welleman**, for his enormous ardency and enthusiasm, especially after having spent a considerable amount of time at home due to illness. Extensive collaboration on the Discrete Element Method led to the implementation of this method in this Master's thesis. **Ir. J.L. Coenders**, for the introduction to and shared knowledge of computational design and structural optimisation. Due to his work for the Structural Design Lab the addition of the name of this group on the front page of this Master's thesis, really means something in the computational and building engineering practise. **Ir. S. Boer**, as the only committee member not associated with the Delft University of Technology, for dealing with the extra time he put into this project, and for his insights in the architectural practise. Something not only used for this Master's thesis, but which will also be used for the upcoming Master's study Architecture.

Dipl. inform. F. Scheurer
is willingly thanked for being the non-official member of the graduation committee and for sharing his knowledge on computational design and especially on genetic algorithms. Based on his papers, and visits to Delft, better insights were received into this complex matter.

Ir. A. Habraken
of Arup Amsterdam who was open for an invitation from him to the Arup office in Amsterdam and could answer various structural related questions on the Groningen Twister project, and thereby supplied better understanding in the design of column supported slabs.

ir. S. de Groot and ir. E. Quispel
for their preliminary insights on the subject of stacking different functional layouts within a multi-use building and their willingness to share their work.

H. Ohmori, prof. dr. J. Strasky, and ir. R. Uytenhaak
who supplied images and text files after requests via e-mail.

ir. R. Pijpers
who, with enormous patience, was able to install AutoCAD 2007 on the desktop at the workspace, which was of essential importance for finishing this Master's thesis.

the graduation students of room 0.72
for their tips and tricks on how to graduate, the laughs that were shared during the complete Master's thesis period, and for the many final presentations and accompanying drinks from those who already graduated.

the fellow board members and friends of the Stichting Dispuut Utiliteitsbouw
who made the finishing of the 'U-profiel' of May 2006 a lot more bearable, who opened up the continent of Asia for an unforgettable study tour to Dubai and the east of China at end of the winter of 2006, where, visiting Shanghai, the research stage of this Master's thesis already could be presented to Chinese fellow students. And of course for their friendship, during the whole combined period of working on the Master's thesis and the board membership.

my parents, my brother, and his girlfriend
for being the biggest support, a graduate student could wish for. Enough said.

And of course everyone else, who, although not contributing directly to the content of this Master's thesis, showed that there is more to life than columns, structural grids, and genetic algorithms, but who contributed to the social and emotional evolution of the author.

summary

general introduction

As more functional requirements are implemented in one building, more pressure on the structural designers, both architects and building engineers, will be transferred. Especially in dense urban areas, multi-use buildings usually have several different layouts, for instance for an underground parking, a ground floor, office floors, and mechanical floors. One of the main problems here is the transfer of loads between the different structural grid systems of the different functions, and as a result, numerous structures are built and will be built with inefficient transfer zones considering the volume of material needed for a floor or the amount of reinforcement or prestressing needed. In lieu of making compromises in the design of structural grid systems per function in order to vertically align the grid lines or grid bands, a structural transfer zone in an intervening floor can be designed. By designing an optimal allocation of vertical oriented structural elements, loads can be transferred between the different structural grid systems, without adversely affecting the functionality and usability of the intervening floor.

structural optimisation design tool

The Master's thesis 'Optimisation of structural transfer zones in multi-use buildings' deals with the development of a computational structural design tool based on an artificial intelligence method, that can determine the optimal solution for the allocation of columns between two structural grid systems. Based on genetic algorithms as an optimisation technique and by using basic and simple rules, the design tool can be used for the determination of size, angle, and placement of load bearing columns of the intervening floor. This tool, implemented in VBA and using AutoCAD as a visualisation tool, allows the user to generate and optimise the configuration of the load bearing elements for an arbitrary design, with the rules following from the demands of several aspects of a structural and functional design. With the addition of non-structural criteria (such as costs, aesthetics and construction) the engineer can complete the design, and create an optimised design that is based on the integration of the load bearing elements into a functionally efficient building floor, rather than being only based on the stress-weight optimisation of the individual components.

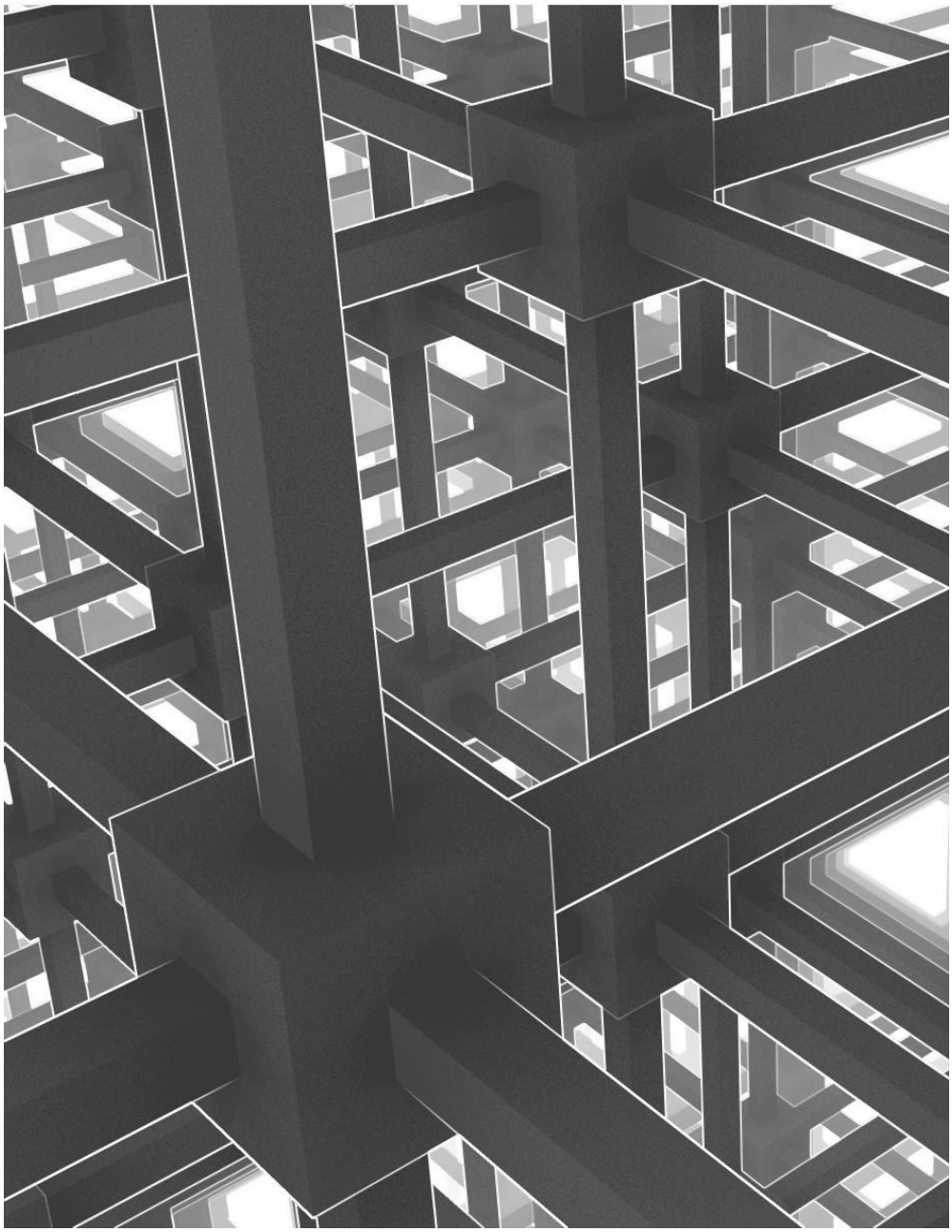
features of the tool

In the AutoCAD environment, the user needs to give the starting and end point of the structural grid line graphically in both the bottom and top layer of the intervening floor to determine the possible location of the columns. Subsequently, the vertical point load, the horizontal point loads, and the moments in the centre of gravity acting on the load transferring structure need to be given numerically. Based on the user input, the tool then randomly generates several solutions in what is called the first generation. By determining the fitness, or in other words the overall compliance with the prescribed desired conditions, the genetic algorithm will use the best solutions to create a new population. This process, including several other genetic algorithm operators, will be repeated until the fittest or best solution per population remains unaltered during a number of generations.

concluding remarks

This Master's thesis shows the capability of an artificial intelligence based design optimisation tool in a predefined setting of the allocation problem of columns in a structural transfer zone. At the same moment, it is made clear that progress can be made for the presented design tool and in scripting design tools for the building practise in general. This also means that it can be expected that tools similar to the tool presented in the Master's thesis will be used more often in the near future. This, however, does not mean that the structural engineer will lose his or her position, as hand calculations and logical interpretations of the result of the design tools will always have to be made.

table of contents



Cubic Space Division, M.C. Escher, litho, 1925, 27 x 26,5 cm[1]

One another intersecting rectangular beams divide each other in pieces of equal length, which each are the edge of a cube. So, the space is filled to infinity with cubes of the same volume

[1] Escher [13]

0,10 table of contents

1 introduction

general introduction to the Master's thesis project

The *manual optimisation* and iteration to minimise weight, cost or for ease of construction issues of structures are instances of non-computational optimisation that are widely used by structural engineers. This includes physical modelling, with the hanging models of Antonio Gaudí as a striking example [Figure 1.1]. By making a physical (scale) model and subjecting it to external loads, a designer can learn about the internal forces in the structural elements and remodel the design accordingly and thus working towards an optimum design. Therefore, when using an optimisation technique like physical modelling, an engineer is able to adapt the initial design in such a way that it will meet the non-structural boundary conditions, while corresponding to e.g. compression lines or minimal energy surfaces.

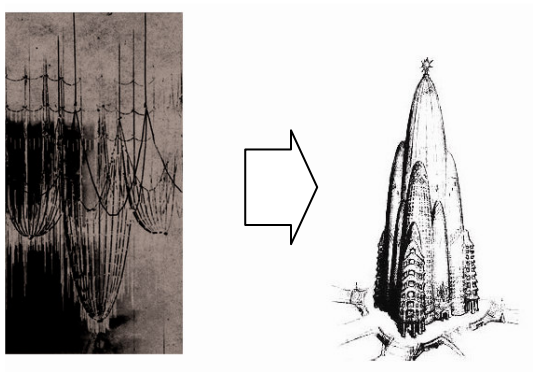


Fig. 1.1. Hanging model by Gaudí [www.gaudiclub.com, August 2005] and a drawing of Hotel New York, design by Gaudí [www.vitruvius.com.br, August 2005] (Model and drawing are not of the same project)

One of the advantages of physical modelling is the insight in the behaviour of the modelled structure[2]. The same can hold true for analysing a virtual model after subjecting it to a *computational* analysis, but often the output will give the user a

[2] Coenders [7]

list containing data without revealing how this data was obtained. But with two disadvantages of physical modelling being two advantages of computational methods (viz. the amount of time to make a model and the simplicity of adjusting the model), together with the accuracy of the latter mentioned, engineers need to comprehend the modelling of a structure based on computational optimisation techniques.

Many *computational optimisation techniques*, based on both 'classical' and 'artificial intelligence' methods can be used by structural engineers in all kinds of fields as well. Nevertheless, for building design only very basic design tools exist based on optimisation. Usually these tools only include a stress versus weight trade-off. One field where an implementation of optimisation-based design tools can give good and maybe surprising results is in the allocation of structural elements in the structural transfer zones of buildings between different layers of functional program and routing. One example of this class of optimisation problems is shown in the Groningen Twister project[3] [see also Section 5.1]. A similar technique that was used for this project, or a technique based on other methods of optimisation can be used for the determination of size, angle, and placement of load bearing columns of an intervening floor, such as a ground floor or a technical floor in a high-rise building. The main reason for using computational methods in dealing with this structural problem is the difficulty that comes with the different structural grids of the different building floors, and that a logical allocation of the load bearing elements might not be evident.

formulation of the general objectives

The goal of the final report is to demonstrate the use of optimisation algorithms in building design for combinations of structural and non-structural criteria as well as the usability of optimisation techniques for combined architectural and structural engineering purposes. This, in particular for the allocation problem of load bearing structural elements of intervening floors as an initiation of the structural design phase, with the use of genetic algorithms as an optimisation technique.

The demonstration will result in the evaluation of a structural design tool that allows the user to generate and optimise the configuration of the load bearing elements, with the rules following from the demands of several aspects of a structural and functional design. With the addition of non-structural criteria (such as costs, aesthetics and construction) the engineer can complete the design, and create an optimised design that is based on the integration of the load bearing elements into a functionally efficient building floor, rather than being only based on the stress-weight optimisation of the individual components.

[3] Scheurer [31]

composition of the report

book one – transfer zones in multi-use buildings and the development and analysis of the optimisation tool

Book one of the Master's thesis deals with the recognition of the problem and a possible solution in the form of a design optimisation tool. As in multi-use buildings, different functions and therefore different structural grid systems are present difficulties can occur when these systems need to be connected. The necessity of transfer zones will be discussed in Chapter 2, along with summaries of two Master's theses of Building Engineering graduates. Chapter 3 deals with the dimensions and different considerations concerning structural grid layout in multi-use buildings. Main chapter of book one is Chapter 4. Here the different characteristics and design aspects of the optimisation tool will be illuminated. In Chapter 5 the validation of the tool will be given on the basis of test runs.

book two – background studies on structural optimisation

The second book of the Master's thesis discusses the concepts of structural optimisation in building engineering, including an enumeration of optimisation methods and an in-depth research on genetic algorithms. In Chapter 2, an introduction into structural optimisation is given and the philosophy behind it is discussed. Chapter 3 and 4 form the main chapters of book two. Here, the optimisation methods and in particular the new artificial intelligence[▹] method genetic algorithms are dealt with. This book concludes with building projects where optimisation played a key role.

To conclude the introduction Q.Q. Liang [23] can be quoted by stating that the "challenge in structural optimisation is to transform it from an exotic and fruitless academic exercise into a rational and efficient design tool for practicing building engineers". The work presented in this Master's thesis is to answer this challenge.

Fig. 1.2. 'Columns' on a predefined grid [www.uni-kl.de, September 2005]

[▹] Traditional artificial intelligence has been centered around the idea of representation of the world. Toward the end of the 1980s, an exciting new field appeared in computational intelligence; 'embodied cognitive science', also known as 'new artificial intelligence'. It was suggested that the discussion about thinking, logic, and problem solving was based on assumptions that come from our own introspection, from how we tend to see ourselves. Next, it was suggested that these assumptions need to be dropped, that we so away with thinking and with what people call high-level cognition and focus on the interaction with the real world. Intelligence must have body, hence 'embodied intelligence'.
From 'Understanding Intelligence' by R. Pfeifer [29]

0,14 introduction

2 concepts of structural optimisation in building engineering

A general introduction into structural optimisation is given in Section 2.1 and deals with the definitions of some basic terms and some general concepts of structural optimisation in building engineering. In Section 2.2 three examples of optimal structures and structural elements are given. Section 2.3 deals with the philosophy behind structural optimisation.

2.1 general introduction into structural optimisation

This section deals with the basic concepts and ideas concerning *structural optimisation*. Main part of this section is subsection 2.1.2, where an introduction into optimisation is given. In the first subsection, some definitions are written down in order to clarify the term 'structural optimisation'.

2.1.1 definitions of basic terms in structural optimisation[4, 5, 6]

First of all, the first term of *structural optimisation* is defined

> **struc·tur·al** *adj* relating to or having or characterized by structure; affecting or involved in structure or construction

As stated in the introduction of this Master's thesis the adjective 'structural' will be used to name aspects considering building engineering and structural engineering matters, rather than structural aspects in general.

Secondly, the term optimisation is given a closer look. On the page after the title page, it is written *'Nil Satis Nisi Optimum'*. Which in English means; 'Nothing Satisfies But The Optimum', where that *optimum* is the best next thing after *perfection* and in some cases can be perfection.

[4] Koenen [22]
[5] Longman Dictionaries [24]
[6] Lexico Publishing Group [38]

The terms optimisation and perfection are often mixed up in spoken and written language, which is actually not that strange as both terms stand for the best or most suitable condition.

> **op·ti·mal** *adj* the best or most suitable; extreme; highest; most favourable or most desirable possible under a restriction expressed or implied
>
> **op·ti·mise** *v* the way that something is performed, executed, carried out or used as effective as possible; trying to make optimal; trying to reach the extreme; make optimal; get the most out of; use best
>
> **op·ti·mum** *n* the best or most suitable for a particular purpose; the best possible situation; conditions; amount of time etc. for something to happen under certain restrictions; the highest achievable; the point at which the condition, degree, or amount of something is the most favourable
>
> **per·fec·tion** *n* the quality or condition of being perfect; the highest degree of proficiency, skill, or excellence, as in some art; faultlessness; in a way that nothing is wrong

As one looks closer to these definitions the difference between optimum and perfection reveals itself in the condition that optimal solutions/situations/conditions are at their best under *certain restrictions*. These boundary conditions make that one or more parameters of the function to be optimised possibly (or probably) cannot reach their best value. But all together the non-optimal individual parameters form the optimised solution. To clarify this, consider a function that contains both the summed area of the cross section of structural elements and the summed load bearing capacity of the structural elements. It needs no explanation that the optimised values for both parameters cannot occur as the minimum amount of material and the maximum bearing capacity are contradictory. This was also stated by Harald Kloft of the Technische Universität Kaiserslautern during the Free Form Design Colloquium in Delft in September 2006. He explained optimisation as 'bringing different parameters in balance, which is not always the perfect solution'.

Or in other words, the optimal solution might be the perfect solution, but when parameters are contradictory, the optimal solution can be far from it. And so, as throughout the text the words 'optimisation' and 'optimum structure' have been used extensively the optimal shape in the context of a structure subjected to multiple loads and support conditions is the shape which best satisfied the constraints, with the degree of satisfaction not necessarily the same for all the constraints.

1,0 Chapter 2 - concepts of structural optmisation

2.1.2 introduction into structural optimisation

general concepts of optimisation problems

According to R.T. Haftka and M.P. Kamat[7] the definition of an optimal design is 'the best feasible design according to a pre-selected quantitative measure of effectiveness'. The effectiveness is dependant, of course, of the boundary conditions that have to be met. The boundary conditions, in their turn, depend on the design that a designer has in mind, with e.g. minimum cost, maximum stiffness or minimum structural depth.

Structural optimisation is a fusion in the areas of engineering, mathematics, (computational) science and technology based on a thick layer of structural design capacities of the designer that has the goal of achieving the best performance for a structure, be it a single beam, a space frame of a complete building. Because of the mathematical complexities the topic of structural optimisation remained to be of more academic interest until last thirty years[8]. Since then there has been a re-focusing on the topic. The two push factors for this change are the availability of high-performance computing power at low cost, and the rapid improvement in algorithms used for design optimisation where thousands of design variables and constraints may need to be handled.

One of the simplest illustrations in studying optimality is the determination of the optimal cross section of a simply supported beam[9]. The ability of a given volume to carry a load in the situation of a beam resting on simple supports with a point load at the mid-span, as shown in the Figure 2.1, is used as a medium to present part of the message as to why structural optimisation is worthy of studying.

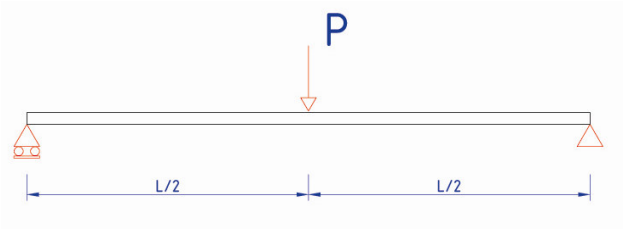


Fig. 2.1. A simply supported beam with a point load *P* at the mid-span

In order to maximise the bearing capacity of the beam the engineer can select from a wide range of beam cross sections. This example is to demonstrate how significant the choice of cross section is.

[7] Haftka [14] [15]
[8] Burns [6]
[9] Kirsch [20]

The design criterion in this case is that the maximum stress in the beam shall not exceed some material limiting value, σ_{max}. The relationship between the stress σ inside the beam and the bending moment M is given by

$$\sigma = \frac{My}{I} \tag{2.1}$$

where y is the distance from the neutral plane at which the stress is calculated. For the simply supported beam the maximum moment is at the mid-span of the beam and is given by

$$M_{max} = \frac{PL}{4} \tag{2.2}$$

where P is the load at the mid-span and L is the length of the beam.

The maximum load capacity can thus be obtained as

$$P_{max} = \frac{4M_{max}}{L} = \frac{4\sigma_{max}}{L}\left(\frac{I}{y}\right) \tag{2.3}$$

The maximum load is obtained by maximising the value of (I/y), which is determined once the shape of the cross section is given. There is an infinite range of ways the material could be arranged and for simplicity reasons four are presented in Figure 2.2.

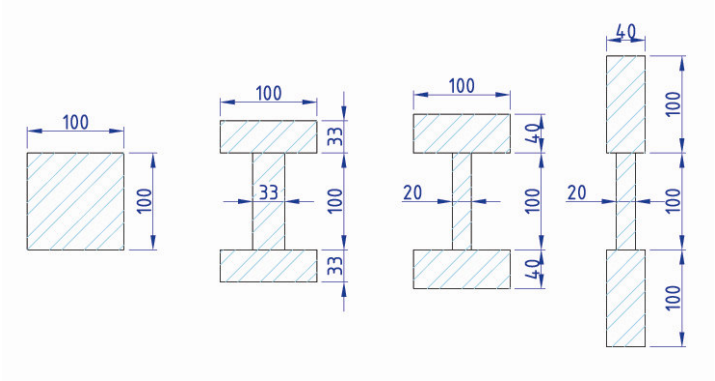


Fig. 2.2. Four possible cross sections for a simply supported beam with equal cross sectional area (A = 10.000)

Using the equation for the maximum bearing capacity, it is not too difficult to evaluate P_{max} for each section and these are presented in Table 2.1.

cross section	P_{max}	P_{max} / $P_{max}(a)$
(a)	$0{,}67\sigma_{max}h^3/L$	1
(b)	$1{,}59\ \sigma_{max}h^3/L$	2,38
(c)	$1{,}86\sigma_{max}h^3/L$	2,80
(d)	$2{,}34\sigma_{max}h^3/L$	3,53

Table 2.1. Maximum load capacities of the beam for various cross sections

This should be convincing as to why sufficient analysis in the structural design practise should be undertaken to ensure that material usage is at a minimum. So it could be concluded that the keywords in structural optimisation are 'the efficient use of materials'. However, manufacturing costs have not been included and it could be that to make section (d) the sum of manufacturing costs plus material costs is greater than for section (a).

Besides, it has to be stated that whilst the stress in the outer fibre of the chosen cross section at the load application point must be equal to or less than the limiting value, and for design purposes we set it equal, the rest of the material is at a much lower stress value. This can be stated because the stress is linear about the middle neutral plane and the bending moment varies linearly from the mid-span value of $PL/4$ to zero at the ends. As a result of this, only approximately 3% of the material of the beam has a greater stress than 75% of the maximum stress[10]. In other words, the bulk of the material, 97%, is at a stress lower than 75% of the maximum. It is this type of problem that provides the real challenge for structural optimisation. The question of how to generate a shape of a structure that makes the best use of the material introduces the concept of *fully stressed design* where, ideally, all the material is at the same absolute stress [Michell structures, see also Section 2.2.3].

[10] Burns [6]

It could be stated that one of the most important fields of structural optimisation is the minimum weight design of structures [Figure 2.3]. This was first recognized by the aerospace industry, where the minimum weight of a structure is of a greater importance than the initial cost considerations, although this ultimately results in lower costs for this industry, and it should be noticed that the weight of structural elements of the complete structure often contributes to the costs of a design. As a result of this, many books are written on structural optimisation regarding minimum weight. In other industries dealing with mechanical, automotive and building engineering systems the cost considerations are of primary interest.

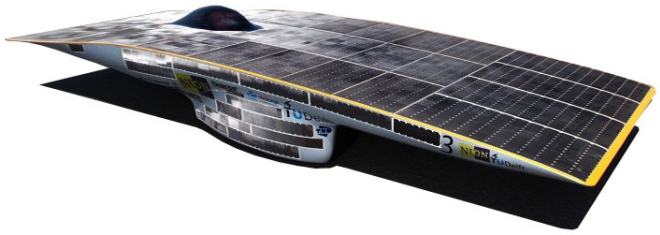


Fig. 2.3. The weight of the Nuna 3 Solar Vehicle was less than that of its predecessors (< 200 kg) [www.kennislink.nl, November 2005]

Before the development of high-speed computation, most of the solutions of structural analysis problems were based on formulations employing differential equations, which were solved analytically. The notion of optimality requires that an *objective function* [see also Section 2.2.1] is given that needs to be minimised of maximised. For example, the objective function for a simply supported beam could be the total mass of that beam [as seen in the first part of this section]. An optimisation problem also requires a number of *constraints*. For the design of the simply supported beam, a typical constraint could be the maximum displacement of the beam. The class of structural optimisation problems which seeks an optimal structural function is called *distributed parameter* structural optimisation[11]. These problems are solved by analytical methods and are often formulated by functions describing continuous distribution of material over the structure, also called *continuous problems*. Another class of structural optimisation problems is called *discretised* structural optimisation where models of structures are involved in a number of variables that vary continuously. The finite element method (FEM) is a method that discretises the structure so that the unknowns of the analysis are discrete values of quantities, such as displacements and stresses at the nodes of the finite element model, rather than functions [for background information on the Finite Element Method, see Appendix A]. However, the greatest drawback of a finite element model and applying the optimisation procedure to the model is that it may

[11] Liang [23]

be distorted to produce elements of unacceptable shapes. In distributed parameter structural optimisation, the optimum solution can be realised by changing parameters which are usually called *design variables* and may be continuous or discrete[12]. For example, the optimisation procedure considering the minimum weight of an I-shaped beam is a continuous problem. But to minimise the manufacturing costs the choice of standard elements is introduced. So, after the optimum design is obtained, the values of the optimum design variables are adjusted to the nearest available discrete value.

Dealing with *multiple objective functions* is complicated and is usually avoided. There are two standard devices for reducing the number of objectives to one[13]. The first is the generation of a *composite* objective function. For example, if the mass of the three bar truss of Figure 2.4 is denoted as m and the stresses in the three bars as σ_i then a composite objective function f could be

$$f = \alpha_0 m + \alpha_1 \sigma_1 + \alpha_2 \sigma_2 + \alpha_3 \sigma_3 \qquad 2.4$$

where the α_i are appreciation factors selected to reflect the relative importance of the four objective functions.

The second device used to reduce the number of objective functions to one is to select the most important one as the only objective function, and to impose limits on the other objective functions. Thus the three bar truss design problem in Figure 2.4 can be formulated as the minimisation of the mass subject to upper limits on the value of the three stresses. The constraints that determine the optimum solution are called active constraints. All other constraints are inactive constraints.

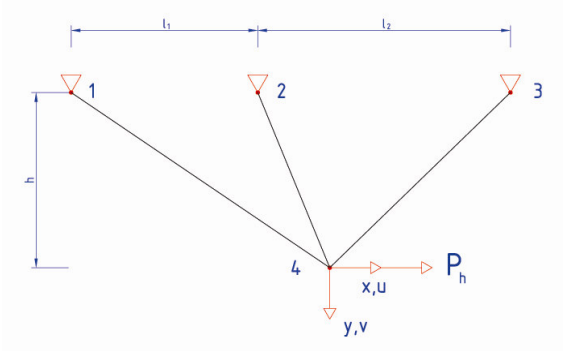


Fig. 2.4. Three bar truss example

[12] Universität Stuttgart [40]
[13] Kirsch [20]

Dealing with optimisation, of course the problem can be stated as a maximisation problem instead of a minimisation problem. Especially when it is considered that minimisation is the negative of a maximisation, also known as duality. This is illustrated in Figure 2.5, where the function

$$y = -x^2 + 4 \qquad 2.5$$

can be used as a function that needs to be maximised, whereas the negative function of this function

$$y = x^2 - 4 \qquad 2.6$$

is a function that needs to be minimised.

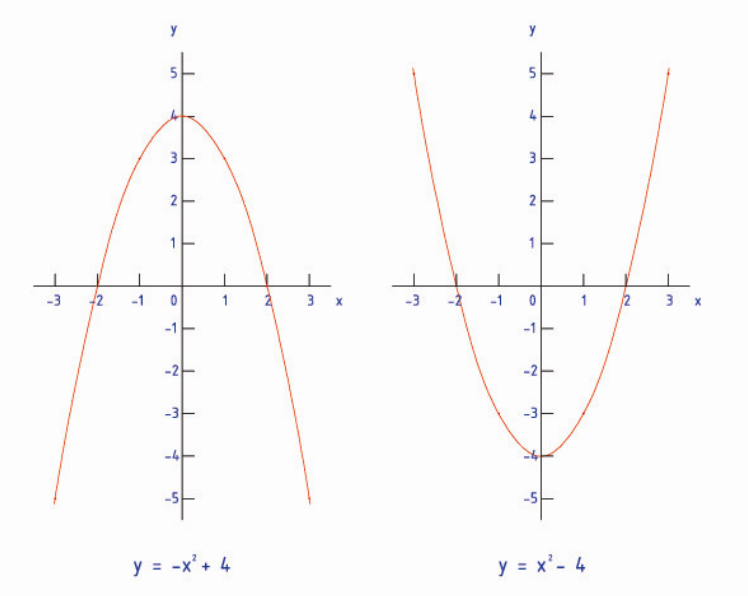


Fig. 2.5. The functions $y = -x^2 + 4$ and $y = x^2 - 4$ are each other negatives.

As stated earlier, many applications in structural optimisation involve design variables that are not continuous in nature and are meaningful only at discrete values, such as commercially available hot-rolled steel sections. Many times selection of solution method depends on the type of discrete design variables, and objective functions for the problem. It is important to note that the discrete variable optimisation problems usually require considerable more computational effort compared to the continuous variable problems. This is true even though the number of feasible points with discrete variables is finite and they are infinite with continuous variables[14].

[14] Kirsch [20]

In the design process, a conceptual design is created by the designer based on his intuition, creativity and past experience. Structural analysis is then undertaken to evaluate the performance of the design. If the design does not satisfy the performance objectives, a new design is then developed. This process is repeated until the design satisfies the multiple performance objectives [Figure 2.6]. In this process structural optimisation methods can be used to decrease the number of loops.

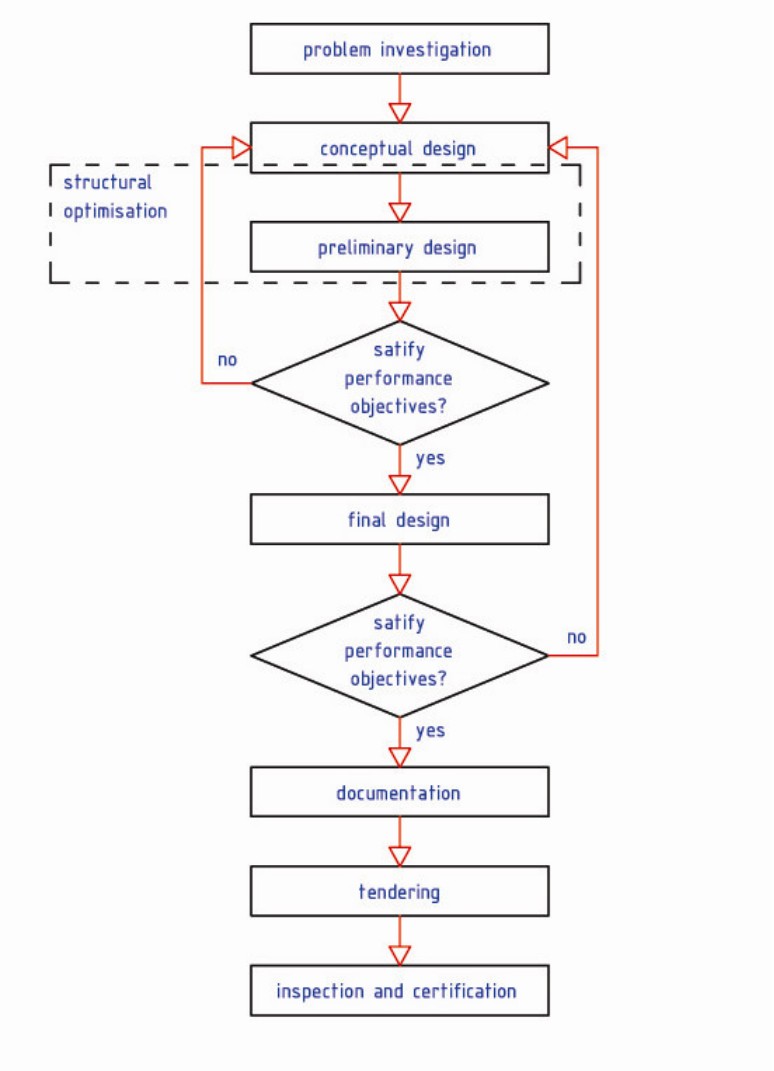


Fig. 2.6. The standard building process and the place where general structural optimisation can be implemented

It is difficult to construct an objective function for a realistic engineering project since it depends on many factors. The constraints can be geometrical restrictions such as the weight and width of the structure or behavioural restrictions such as stresses, displacements, mean compliance, frequency and buckling loads. In order to obtain an optimal structure, a number of alternative structural systems must be invented and evaluated. The invention of structural systems is the most challenging task in structural design since it involves a large number of possibilities for the structural layout. The traditional design process is highly time-consuming and expensive. In the fully optimal design procedure, the designer does not produce the structural system on past experience, but still needs to analyse it of course [Figure 2.7][15]. The real optimisation process will have characteristics of both processes.

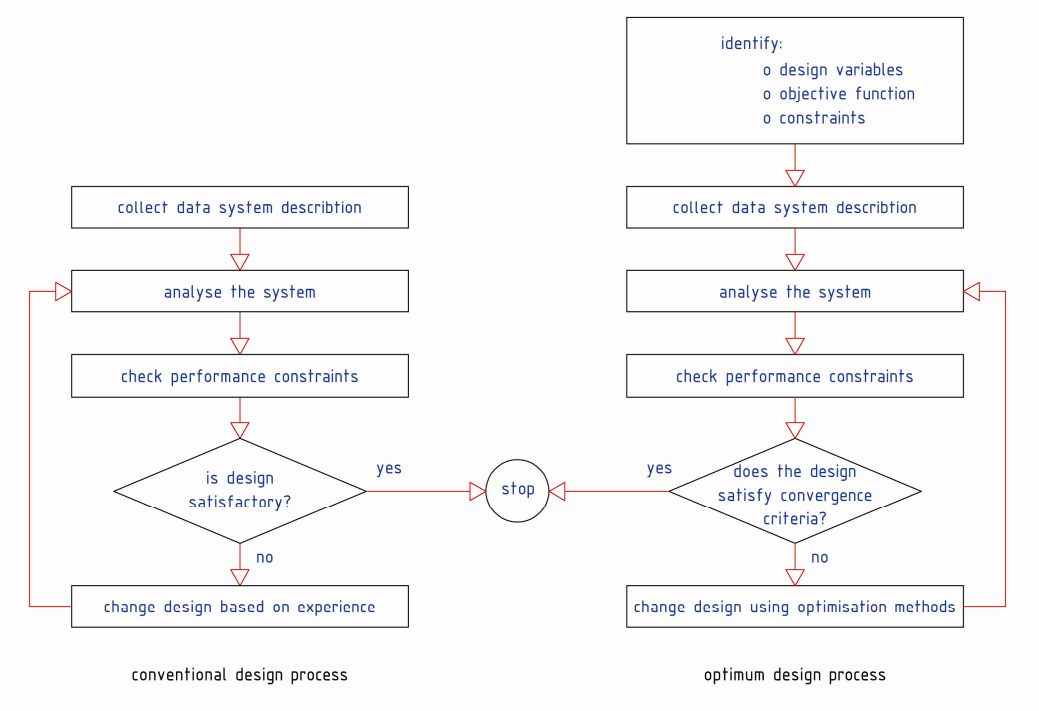


Fig. 2.7. The conventional design process versus the optimum design process for the relevant part of the total building process

[15] Kirsch [20]

The fully optimal design process differs from the conventional design process in the following ways

- there is an additional initial step 0 where the design problem is formulated as an optimisation problem
- in step 5, the design is improved using an optimisation method, rather than based on experience as in the conventional approach
- the stopping criterion is step 4 is based on whether the current design is the best. The conventional approach accepts any feasible design

This makes the optimum design process a more formal process than the conventional design process. The optimum design process takes into account all the constraints simultaneously, and iteratively improves the design, while minimising of maximising the objective function. Besides if problem parameters are changed the conventional process must start over again, demanding repetition of all the calculations, which can be quite tedious.
Once the optimum design problem has been formulated, it is relatively easy to obtain solutions for different conditions and requirements. Formulation of an optimum design problem uses the following steps[16] [see the first box of the optimum design process in Figure 2.7]

1. identification and definition of independent design variables

   as a first step in the problem formulation phase, the variables that describe the design of the system must be identified and defined explicitly. These variables are such that once their values are specified, a design of the system is known. Usually, the design variables are independent of each other. If some of the dependent variables are designated as design variables, then there must be equality constraints between the variables to have a meaningful design of the system. At this stage of the problem formulation phase, it is also necessary to collect all the design parameters and data related to the problem and define them precisely. These include material properties, loading conditions, and the like. Any additional problem variables that depend explicitly or implicitly on the design variables are called *dependent variables* (such as the cross-sectional properties of members, the bending stress, the shear stress, the deflection, etc.).

[16] Kirsch [20]

2. identification and definition of an objective function

   as a second step in the problem formulation phase, an objective function that measures the relative merit of alternate designs needs to be identified and defined. This is a scalar function that depends on the design variables. Once a design is specified, it should be possible to evaluate this function. Various objective functions are the cost of the system, mass of the system, maximum deflection, moment capacity, and so on.

3. identification and definition of constraints

   each design problem has constraints that must be met, such as service, strength, and available resources constraints. Therefore, as the final step in the problem formulation phase, all the constraints on the system need to be identified and defined. Usually constraint expressions involve both the dependent and independent design variables as well as explicit bounds on the variables. Examples of the quantities to be constrained are stress, buckling load, natural frequency, and deflection. An evaluation of the constraint functions requires response analysis of the system for all applied loading cases.

structural optimisation methods

The available methods of structural optimisation may conveniently be subdivided into two distinctly different categories called analytical methods and numerical methods. While analytical methods emphasise the conceptual aspect, numerical methods are concerned mainly with the algorithmical aspect.

Analytical methods are usually employing the mathematical theory of calculus, variational methods, etc., in studies of optimal layout or geometrical forms of simple structural elements, such as beams, columns, and plates. These methods are most suitable for fundamental studies of single structural components, the material layouts of structural components and simple skeletal structures under simple loading conditions, but are usually not intended to handle larger structural systems. Analytical methods cannot be used to deal with the topology optimisation of complex practical problems (problems solved by analytical methods are denoted as *distributed parameter* structural optimisation). The structural design is represented by a number of unknown functions and the goal is to find the form of these functions. The optimal design is theoretically found exactly through the solution of a system of equations expressing the conditions for optimality. An example for this approach is the theory of layout, which seeks the arrangement of structural members that produces a minimum volume structure for specified loads and materials [e.g. Michell structures, see also Section 2.2.3]. Since they are applied without meaningful constraints on the geometric form of the structure, such theorems often yield impractical solutions.

Work on analytical methods, although sometimes lacking the practicality of being applied to realistic structures, is nonetheless of fundamental importance. Analytical solutions, when they can be found, provide valuable insight and the theoretical lower bound optimum against which more practical designs may be judged.

Numerical methods usually employ a branch in the field of numerical mathematics called *mathematical programming*. Since the early nineties developments in this branch are closely related to the rapid growth in computing capacities. In the numerical methods, a near optimal design is automatically generated in an iterative manner. An initial guess is used as a starting point for a systematic search for better designs. The search is terminated when certain criteria are satisfied indicating that the current design is sufficiently close to the optimum. Rapid developments in the programming methods as well as in the application of such methods in design facilitate the solution of realistically large practical design problems. Problems solved by numerical methods are called *finite optimisation problems* or *discrete parameter optimisation problems*. This is due to the fact that they can be formulated by a finite number of variables. Assignment of numerical values to these variables specifies a unique structure. Design optimisation of practical structures is accomplished mainly by the use of finite formulations. An example of a numerical method is the topology optimisation method [Section 2.2.2.].

Some of the mathematical programming methods, such as *linear*, *quadratic*, *dynamic*, and *geometric programming* algorithms, have been developed to deal with specific classes of optimisation problems[17]. Though the history of mathematical programming is relatively short, there has been a large number of algorithms developed for the solution of numerical optimisation problems. However, there is no single best method for all optimisation problems. There is an obvious need, therefore, for familiarity with the basic concepts of numerical optimisation.

Of the engineering disciplines, structural design has probably seen the most widespread development and application of numerical optimisation techniques. Using numerical optimisation as a design tool has several advantages[18]:

- reduction in design time and improving the quality of the design. Optimisation is an effective tool to reach a high quality design much faster. Even in cases where optimisation by itself does not save design time or cost, the final result is a product that is superior;
- the ability of dealing with large numbers and a wide variety of design variables and constraints relative to traditional methods;
- applying systematised logical design procedures may lead to improved, non-traditional and unexpected results, particularly in a new design environment.

[17] Xie [34]
[18] Liang [23]

One of the most effective uses of numerical optimisation is to make early design trade-offs using simplified models. The advantage is that optimal designs can be compared instead of just comparing non-optimal solutions. On the other hand, numerical optimisation has some limitations to be aware of[19]:

- the quality of the result is only as good as the assumed analysis model. That is, optimisation techniques are limited to the range of applicability of the analysis method;
- incomplete problem formulation, such as ignoring an important constraint, may lead to meaningless if not dangerous designs. Furthermore, improper formulation might reduce the real factors of safety that now exists;
- the number of design variables is restricted due to the computational effort involved in solving large problems by many optimisation methods;
- most optimisation algorithms can solve problems with continuous functions. In addition, highly nonlinear problems may converge slowly or not at all;
- in many problems it cannot be guaranteed that the global optimum design will be obtained. Therefore, it may be necessary to restart the optimisation process from several different designs and compare the results.

In summary, optimisation techniques can greatly reduce the design time and yield improved, efficient and economical designs. However, it is important to understand the limitations of these techniques. In addition, it should be recognised that the absolute best design will seldom be achieved. Thus, optimisation methods can be viewed as a valuable and convenient tool to achieve improved designs although some if its theoretical optima are of little value in the design process.

2.2 optimisation of structures and elements in building engineering

In this section, three examples are given of optimisation methods for structural elements [simply supported beams, Section 2.2.1] and structures [trusses and Michell structures, respectively Section 2.2.2 and 2.2.3]

2.2.1 optimisation of simply supported beams[20]

The problem formulation procedure, as presented in Section 2.1.2 is illustrated for a simply supported rectangular and I-shaped beam design problem [Figure 2.8]. The objective is to minimise the total mass or, equivalently, the cross-sectional area A of the beam. Therefore, a feasible design with smaller cross-sectional area is called a 'better design' compared to the one with a larger area. It should be noticed that the given examples are very basic, especially considering the constraints.

[19] Liang [23]
[20] Arora [1]

A spreadsheet program like MS Excel has the capability of solving constrained non-linear optimisation problems, such as the beam problem. The solver of MS Excel is an 'Add- in', which is an optional module for adding optimisation capabilities to the spreadsheet program. Main requirements are that one cell contains the objective function formula, and the independent variables appear somewhere else. In 'what if' scenarios, the design parameters can be altered to check different solutions.

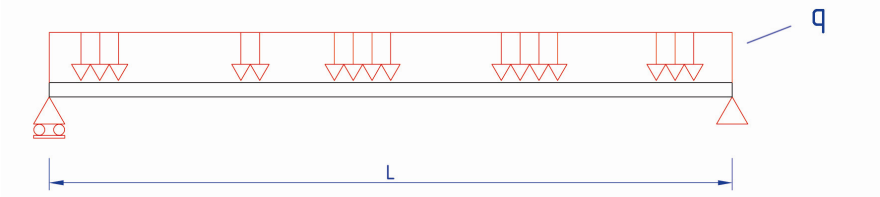


Fig. 2.8. The simply supported beam with live load *q* with a rectangular or I-shaped cross section

simply supported rectangular beam

Mathematically, the beam design optimisation problem is stated as

- compute the design variables h and b to minimise the cross sectional area A
- subject the problem to the constraint on bending stress

independent variables

h = beam height, mm (with a lower and an upper bound)
b = beam thickness, mm (with a lower and an upper bound)

design parameters

L = span, m
σ_y [sigma_y] = yield stress, N/mm^2
q = live load, kN/m

dependent variables

A = cross sectional area, mm^2
I = moment of inertia, mm^4
W = section modulus, mm^3
M = bending moment, kNm
σ [sigma] = bending stress, N/mm^2

constraints

bending stress: $\sigma \leq \sigma_y$

In Figure 2.9 the layout of the MS Excel spreadsheet is given in which the minimum volume of steel is determined dependant of the variables. *lb* and *ub* stand respectively for 'lower bound' and 'upper bound'.

Optimisation of a simply supported rectangular beam

independent variables name	**lb**		**symbol**	**value**	**ub**	**units**
height	100		h	300	300	mm
thickness	50		b	78,61635228	150	mm
parameters name	**symbol**		**value**	**units**		
span length	L		10	m		
yield stress	sigma_y		265	N/mm^2		
live load	q		25	kN/m		
dependent variables name	**symbol**	**formulas**	**result**	**units**		
cross sectional area	A	=h*b	23584,90568	mm^2		
moment of inertia	I	=(1/12)*b*h^3	176886792,6	mm^4		
section modulus	W	=(b*h^2)/6	1179245,284	mm^3		
bending moment	M	=(q*L^2)/8	312,5	kNm		
bending stress	sigma	=(M*10^6)/W	264,9999997	N/mm^2		
objective function name						
volume of steel		=(A/(1*10^6))*L	0,235849057	m^3		
constraints name	**value/eqn**		**</>/=**	**value/eqn**	**name**	
bending stress	265		<	265	allowable bending stress	

Fig. 2.9. Layout of the spreadsheet for the determination of the minimum volume of steel for a simply supported rectangular beam

The result for this calculation is a rectangular beam with a height of 300 mm and a width of almost 79 mm and the volume of steel is 0,236 m^3 [Figure 2.10]. It isn't surprising that the height of the beam yields to the upper bound, because this results in the maximum moment of inertia.

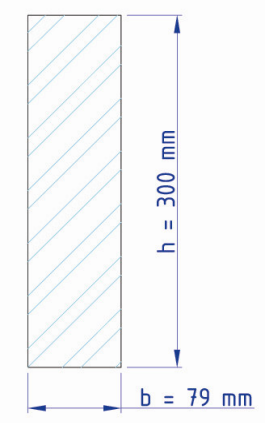


Fig. 2.10. Cross section of the calculated rectangular beam with minimum volume of steel

When dealing with a two-variable design problem the graphical approach can be used. In this approach, each constraint function is plotted on a graph sheet [Figure 2.11]. This results in curves of each constraint that divides the design space into two parts. One side of the curve represents all the design points that satisfy the constraint (feasible design) and the other side represents the designs that violate the constraint (infeasible designs). Next, it is needed to locate the best possible design in the feasible region, also called the constraint set. In order to do that, a few *iso-cost curves* are drawn through the feasible region and a point that gives the least value to the objective function is identified as the optimum solution. The coordinates for the optimum point are read directly from the graph. It can be seen that the results of the graphical approach match the results of the spreadsheet.

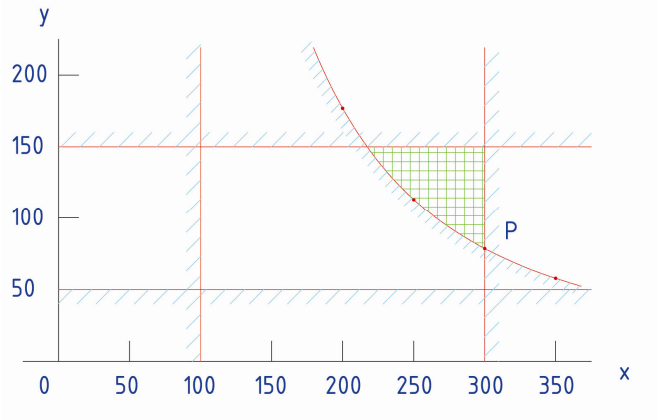


Fig. 2.11. The graphical approach based on the rectangular beam problem. Point P denotes the optimum cross section of the beam

simply supported I-shaped beam

The same procedure can be followed when dealing with an I-shaped beam. In the next example, additional variables, parameters and constraints are given in order to present a more difficult optimisation problem. With the four independent variables (instead of two in the previous example) the graphical approach cannot be used to check the results. This example still is not completely satisfactory for practical use (e.g. the web crippling and flange buckling aren't taken into account), but as stated before, this is merely an example to demonstrate an optimisation problem.

Mathematically, the beam design optimisation problem is stated as

- compute the design variables h, b, tf, tw to minimise the cross sectional area A
- subject the problem to the constraint on bending stress, shear stress and deflection

independent variables

h = beam height, mm (with a lower and an upper bound)
b = flange width, mm (with a lower and an upper bound related to h)
tf = flange thickness, mm (with a lower and an upper bound related to b)
tw = web thickness, mm (with a lower and an upper bound related to h)

design parameters

L = span, m
E = modulus of elasticity, N/mm^2
σ_y [sigma_y] = yield stress, N/mm^2
q = live load, kN/m

dependent variables

A = cross sectional area, mm^2
I = moment of inertia, mm^4
W = section modulus, mm^3
p = uniform load – own weight, kN/m
M = bending moment, kNm
σ [sigma] = bending stress, N/mm^2
V = shear force, kN
τ [tau] = average shear stress, N/mm^2
u = deflection, mm

constraints

bending stress: $\sigma \leq \sigma_y$
shear stress: $\tau \leq 0{,}58\sigma_y$
deflection: $u \leq 0{,}004 \cdot L$

2,0 Chapter 2 – concepts of structural optmisation

In Figure 2.11 the layout of the MS Excel spreadsheet is given in which the minimum volume of steel of the I-shaped beam is determined dependant of the variables.

Optimisation of a simply supported I-beam

independent variables name	**formulas**	**lb**		**symbol**	**value**	**ub**	**formulas**	**units**
web height		0,18		h	0,556426138	0,6		m
flange width	=1/2,5*h	0,2225705		b	0,222570455	0,3091256	=1/1,8*h	m
flange thickness	=1/15*b	0,014838		tf	0,01483803	0,0278213	=1/8*b	m
web thickness	=1/60*h	0,0092738		tw	0,009273769	0,0139107	=1/40*h	m

parameters name	**symbol**	**value**	**units**
span length	L	12	m
modulus of elasticity	E	210000	N/mm^2
yield stress	sigma_y	235	N/mm^2
live load	q	25	kN/m

dependent variables name	**symbol**	**formulas**	**result**	**units**
cross sectional area	A	=h*tw+2*b*tf	0,011765182	m^2
moment of inertia	I	=(1/12)*((tw*h^3)+(2*b*tf^3))+(2*b*tf*((1/2)*(h+tf))^2)	0,000672132	m^4
section modulus	W	0,002415891	0,002415891	m^3
uniform load - own weight	p	=A*7,9	0,092944936	kN/m
bending moment	M	=((q+p)*L^2)/8	451,6730088	kNm
bending stress	sigma	=(M/W)/1000	186,9591806	N/mm^2
shear force	V	=((q+p)*L)/2	150,5576696	kN
average shear stress	tau	=V/(h*tw)/1000	29,17689612	N/mm^2
deflection	u	=(5/384)*(q+p)*(L*1000)^4/(E*(I*10^12))	47,99999956	mm

objective function name				
volume of steel		=A*L	0,141182181	m^3

constraints name	**value/eqn**	**</>/=**	**value/eqn**	**name**
bending stress	186,95918	<	235	allowable bending stress
shear stress	29,176896	<	136,3	allowable shear stress
deflection	48	<	48	allowable deflection

Fig. 2.12. Layout of the spreadsheet for the determination of the minimum volume of steel for a simply supported I-shaped beam

The result for this calculation is a I-shaped beam with a height of 556 mm, a flange width of 223 mm, a flange thickness of 15 mm, and a web thickness of 9 mm and the volume of steel is 0,141 m^3 [Figure 2.13]. In this example, not the bending stress but the deflection is the active constraint.

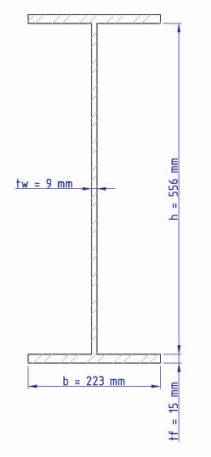


Fig. 2.13. Cross section of the calculated I-shaped beam with minimum volume of steel

For the live load on a roof, normally p_{rep} = 1,0 kN/m² is assumed. So the live load q = 25 kN/m² is rather high. The height of the I-shaped beam is larger than the common $\frac{h}{l}$ ratio of $\frac{1}{40} * l$ for steel profiles, but it is smaller than the $\frac{h}{l}$ ratio of $\frac{1}{20} * l$ for plate girders.

It is important to note here that the constraints can be written explicitly in terms of the design variables *h*, *b*, *tf*, and *tw* by substituting expressions for the dependent variables. However, there are many applications where it is not possible to eliminate the dependent variables, such as stresses and deflections, to obtain explicit expressions for all the functions of the optimisation problem in terms of the design variables. In such cases, the dependent variables must be kept in the problem formulation and treated in the solution process. Such methods have been developed for various applications and are available for use in practical applications.

In order to convert the I-beam problem to a two-variable problem, it can be assumed that the flange width is a fraction of the web height and the flange thickness is proportional to the web thickness, so the problem can be expressed in two variables, the web height and web thickness.

2.2.2 size, shape, and topology optimisation

general introduction into shape, size, and topology optimisation

Applications of numerical methods to truss problems and other discrete models were first described in the early sixties, but only recently have these challenging large-scale problems attracted renewed interest, especially for producing specialised algorithms[21].

In the design of the size, shape, and topology of a structure the interest is in the determination of the optimal placement of a given isotropic material in space, which means, it should be determined which points of space should be material points and which points should remain void (no material). The geometric representation of a structure is thought of as similar to a black-white rendering of an image. In discrete form this then corresponds to a black-white raster representation of the geometry, with pixels given by the finite element discretisation. So, in its most general setting shape, size and topology optimisation of continuum structures should consist of a determination for every point in space if there is material in that point or not. Alternatively, for a FEM discretisation every element is a potential void or structural member. In other words, the ground approach is that

[21] Bendsøe [2]

for an initially chosen layout of nodal points in the truss structure or in the finite element mesh, the optimum structure connecting the imposed boundary conditions and external loads is found as a subset of all the elements of the initially chosen set of connections between the truss nodal points or the initially chosen set of finite elements. The positions of nodal points are not used as design variables, meaning that these points are fixed.

terminology and representation

The three principles size, shape, and topology optimisation can be mentioned as one under the common denominator of *layout optimisation*[22] [Figure 2.14, on the next page]

Size optimisation

The main feature of the size problem is that the domain of the design model and variables is known a priori and is fixed throughout the optimisation process. Only the size of certain element is optimised without changing the shape or topology of the structure. Size optimisation is to find the optimal cross-sectional properties of members in a truss or frame structure or the optimal thickness distribution of a plate structure. It has the goal of maximising the performance of a structure in terms of the weight and overall stiffness or strength while the equilibrium condition and the design constraints are satisfied. The design variable is the cross-sectional area of truss members or the thickness of a plate.

Shape optimisation

The goal in shape optimisation, or geometry optimisation, is to find the optimum geometry of the domain, that is, the shape problem is defined on a domain which is now the design variable. In shape optimisation, the objective is to find the optimal shape of the design domain, which maximises its performance. The shape of the design domain is not fixed but rather is a design variable. In shape optimisation, only the boundaries of the design domain are changed but not the topology of the domain.

Topology optimisation

The purpose of layout optimisation is to find the optimal layout of a structure within a specified region. The only known quantities in the problem are the applied loads, the possible support conditions, the volume of the structure to be constructed and possibly some additional design restrictions such as the location and size of prescribed holes. In this problem the physical size and the shape and connectivity of the structure are unknown.

[22] Bendsøe [3]

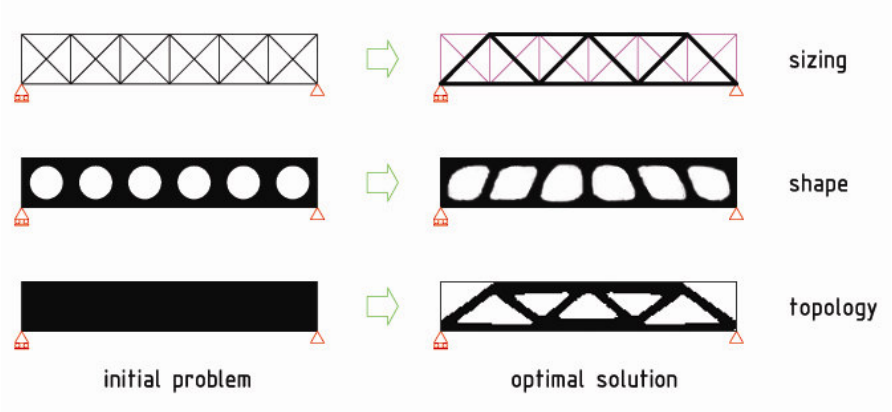


Fig. 2.14. A representation of the size, shape, and topology optimisation method

The topology, shape and size of the structure are not represented by standard parametric functions but by a set of distributed functions defined on a fixed design domain. These functions in turn represent a parameterisation of the rigidity tensor of the continuum and it is the suitable choice of this parameterisation, which leads to the proper design formulations for layout optimisation.

The practical use of topology design to date often has been on the level of a creative sparring partner in the initial phase of a design process. Thus the output of the *homogenisation method*[23], as how use of topology optimisation is also called, has been used to identify potential good designs, the completion of the design being based entirely on traditional skills of the design office. One effect of the topology method that cannot be underestimated is the efficient testing of the appropriateness of the model of loads and supports. As the topology is very sensitive to a proper modelling of the load environment, one can immediately discover discrepancies or inaccuracies in this modelling. The results of using the homogenisation method for optimal topology design tend to favour the use of the sub-optimal microstructures, as these, from a practical point of view, results in more classically useful structures. In the future it will probably implement, for example, production requirements as constraints that will limit the final design. It is natural to integrate the material distribution method and the boundary variations approach into one design tool, employing the topology optimisation techniques as a pre-processor for boundary shape optimisation.

The topology is of great importance for the performance of the structure[24], and it has turned out that the compliance optimised topologies generated using the homogenisation method are very good starting points for optimisation concerning several other criteria such as maximum stress, maximum deflection, etc.

[23] Bendsøe [3]

[24] Universität Stuttgart [40]

In Figure 2.15 a flow of an integrated design system with topology design and boundary shape design modules is given.

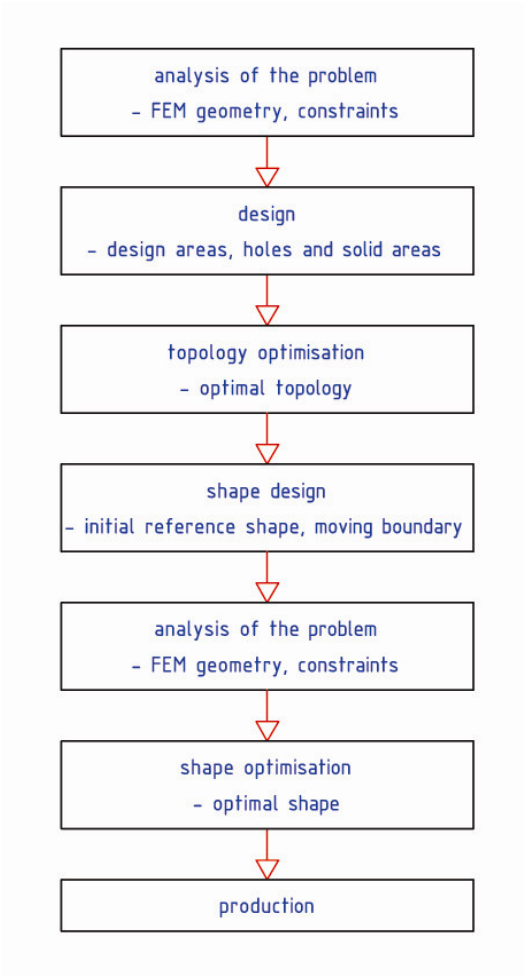


Fig. 2.15. A flowchart of an integrated design system with topology design and boundary shape design modules[25]

A topology optimisation code of Bendsøe [39] can be implemented in Matlab to give an example of the potential of the homogenisation method. The Matlab implementation is given in at http://www.topopt.dtu.dk/ [39].

[25] Bendsøe [2]

evolutionary structural optimisation

A relative new method is the evolutionary structural optimisation (ESO) method[26]. It is a simple concept of slowly removing (or shifting) inefficient material from a structure so that the resulting shape of the structure evolves towards an optimum. The ESO method is based on a simple concept that the step-by-step removal of the inefficient parts from the initial structure leads the structure toward an optimised configuration. There is, however, no consistent rule for determination of the control parameters needed in the evolutionary process of ESO such as, what is called, rejection ratios, evolution ratios and tolerance parameters for convergence. Additionally it has to be pointed out that the operations in the process of the original ESO are only those for removing inefficient parts. It has been found that the evolutionary optimisation method can be effectively used for examination of the structural form, especially in the beginning stage of the design process. The organic form of the structure generated through the usage of the computational morphogenesis scheme has not only a structural rationality but also a fresh appearance not easily acquired only through the usual designing process [see also Section 5.3].
Structural optimisation methods can be enabled to obtain the structural form of which characteristic values are set to be extreme values while the subsidiary conditions imposed on the stress or displacement at specified portions of the structure are satisfied. The building engineering industry has to satisfy all conditions required from the aspects of planning, architectural design, life facilities and other mathematically factors that are hard to prescribe, such as social impact on the human environment. It can be said that all these factors unsuitable for mathematical description have been keeping the building and architectural engineering away from effective applications of optimising methods. However, regarding the conditions required from the planning or the life facility as the constraint conditions, there can be useful tools for building and architectural engineering in the structural optimisation field.

extended evolutionary structural optimisation

Living things (flora and fauna) have been evolving their shapes to survive under various environments they have encountered. They are thought to evolve themselves toward better shapes by removing unnecessary parts, and, on the other hand, by extending necessary parts as well.

[26] Xie [34]

Standing at this point of view, the extended evolutionary structural optimisation[27] (extended ESO) method has been proposed, where two ideas are newly introduced

1. shape control scheme by the contour lines for two dimensional problem and the contour surfaces for three dimensional problem of sensitivity number;
2. bi-directional evolution.

An idea of contour line or contour surface is introduced for the determination of criteria of the boundary regions. Additionally, the bi-directional evolution which is the evolution scheme with not only deleting the concerned regions but also increasing them has been also newly introduced. Consequently, the proposed scheme makes the ordinary ESO method much more powerful.

In the ordinary ESO method, rejection of the inefficient part of the structure is carried out referring to the value of rejection ratio, which is given as a definite value in advance for computation. Consequently, the rejection procedure is performed throughout the whole evolutionary process of computation based upon that definite initial value and no attention is paid on the situation of the structure on evolution.

In the extended ESO method, utilisation of the contour line is introduced as a new idea for evolutionary process to actively control the rejection ratio as well as the portion of evolution. This idea makes it possible to remove the inefficient parts of the structure largely at the early stage of the evolution and to gradually change the speed of the rejection process according to the actual situation of the evolution.

As had been mentioned above, deletions of the portions of the structure as well as addition are realised through usage of the contour lines for the 2-dimensional structures. In a similar manner, we can extend the way of thought to the evolutions of 3-dimensional structures, where the contour of the stress or the other prescribed characteristics values such as deflections, natural frequencies, linear buckling loads and so on should be replaced with the contour surface.

[27] Ohmori [27]

Figure 2.16 shows the evolutional process of the bridge type structures having the road on its upper part, for the case in which the relative stress of Von Mises is adopted for prescription of the contour surface, which can be written in the following form

$$\sigma^{Von\ Mises} = \frac{1}{2}\sqrt{(\sigma_x - \sigma_y)^2 + (\sigma_y - \sigma_z)^2 + (\sigma_z - \sigma_x)^2 + 6(\tau_{xy}^2 + \tau_{yz}^2 + \tau_{zx}^2)} \qquad 2.7$$

where σ_x, σ_y, σ_z, τ_{xy}, τ_{yz}, τ_{zx} represent the components of the normal stresses and the shear stresses in x-, y-, and z-direction. By using this characteristic value, we can get the mechanical information of the portion of the structure, by which the necessity of deletion or addition can be judged. It can be observed that in the evolutional process of the bridge the form changes not only in the elevation but also in its thickness distribution.

As it can be seen from the figure, the structure continuously changes its form in every point of itself and it is clear that only 3D approach can realise such characteristics.

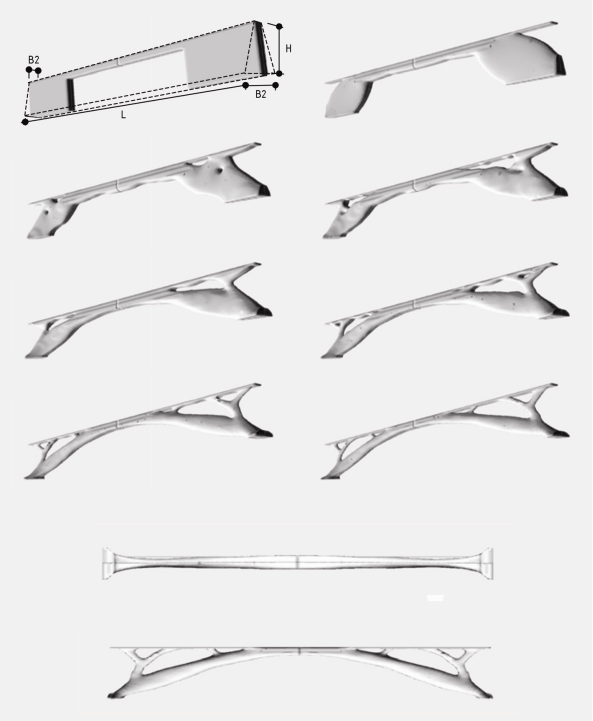


Fig. 2.16. The elevation and the plan of a bridge structure obtained with the ESO process [27]

performance-based optimisation

The principles described above are also dealt with in the performance-based optimisation (PBO) method[28]. Similarly, this method generates an optimal design by gradually removing inefficient material from a structure or adding efficient material to the structure until the performance of the structure is maximised. In PBO design, the weight of a structure is usually selected as the performance objective and structural response parameters such as stresses, displacements, overall stiffness and frequency are treated as performance-based constraints.
It is realistic to minimise the weight or cost of a structure subject to geometrical constraints and performance-based constraints, which include stress, displacement, mean compliance, frequency and buckling load constraints. This is because performance-based constraints are usually prescribed in the design codes of practice.

2.2.3 description of Michell structures

theory of Michell structures[29, 30, 31]

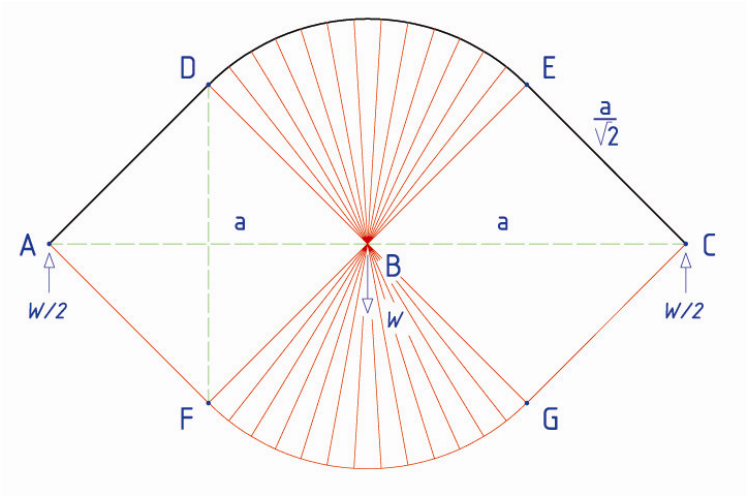
The theory of structures has two aspects: one is the analysis of stress distribution in a completely specified structure and the other is the design of a structure for a specific purpose. Before any attempt may be made to design a structure on strictly logical grounds some general objective must be set. Usually the designer's aim is value – that is, cheapness combined with serviceability. Clearly the achievement of this aim, the design of a structure which shall be both cheap to build and cheap to maintain, must depend principally on economic factors, and these factors themselves must be subjected to geographical and other influences.

As was already mentioned, the costs of any structure of a given type is often more or less proportional to its weight, so that it may at least be claimed that of two structures, both adequate to the same purpose and materially different neither in general design nor in performance, the lighter is to be preferred.

In general, statically determined structures will be lighter than the redundant structures from which they are derived by the omission of certain members; but it is also shown that certain redundant structures can be as light as the lightest non-redundant structure of their kind. In 1904 A.G.M. Michell [1870-1959] studied the problem of the form structures should have if they are to be as light as possible.
He found that structures, which are funicular polygons (a polygonal figure assumed by a cord fastened at its extremities, and sustaining weights at different points), are potentially of minimum weight and maximum stiffness. Michell structures of

[28] Liang [23]
[29] Cox [9]
[30] Hemp [18]
[31] Owen [28]

orthogonal tension and compression members in directions which can be those of equal principal strains are the next lightest. Figure 2.17 gives the most efficient Michell structure for carrying a central load *w*.



2.17. The most efficient Michell structure for carrying a central load *w*. All members are in tension or compression

The discussion of the possible lightest forms of structure has been based on the assumption that the structure has to carry one *unique set of loads* and that these loads were specified in magnitude and their points of application given. As a result, any structure will be composed essentially of systems of struts and ties, or sheets which behave as struts or ties. The simplest Michell strain field which can be imagined is that in which all the strains are equal in all directions. A member in any direction in this field will be strained the same amount as any other member. The structure will then be all in compression or all in tension.

In *The analysis of light structures*, Owen [28] studies structures of minimum volume of material similar to Michell.

Owen writes; suppose a typical compression member is designed to work to a mean compressive stress f_c and a typical tension member to a mean tensile stress f_t. If P_c is the compressive force in a typical strut, the area A_c of the member will be given by

$$A_c f_c = P_c \qquad 2.8$$

Similarly if P_t is the tensile force in a typical tie, the cross-sectional area A_t of the tie is given by

$$A_t f_t = P_t \tag{2.9}$$

However, Owen prefers to represent the force in a typical member *jf* of a structure by P_{jf} and to take this force as positive when it is tensile. P_{jf} will then be replaced when appropriate by either P_t or $-P_c$, and both P_t and P_c, will be positive values. The *x* component of the force in *jf* exerted on the joint *j* will be

$$P_{jf}(x_f - x_j)/l_{jf} \tag{2.10}$$

and the *x* component that it exerts on the joint *f*

$$-P_{jf}(x_f - x_j)/l_{jf} \tag{2.11}$$

By denoting the external loadings acting on the joints *j* and *f* in the direction of the axes for equilibrium, Owen finally comes to a constant which can be written as

$$k_m = \Sigma(xX + yY + zZ) \tag{2.12}$$

so k_m is the sum of the products of coordinates and corresponding force components.

When f_t for all ties and f_c for all struts are constant and when V_t and V_c are respectively the volume of the tension and compression members, then

$$V_t f_t - V_c f_c = k_m \tag{2.13}$$

Or in words, the volumes of the tension and compression members (V_t and V_c) multiplied with respectively the tension and compression stresses should be in equilibrium with the external loads acting on the structure. The total volume of the whole structure can be written as

$$V = V_t + V_c \tag{2.14}$$

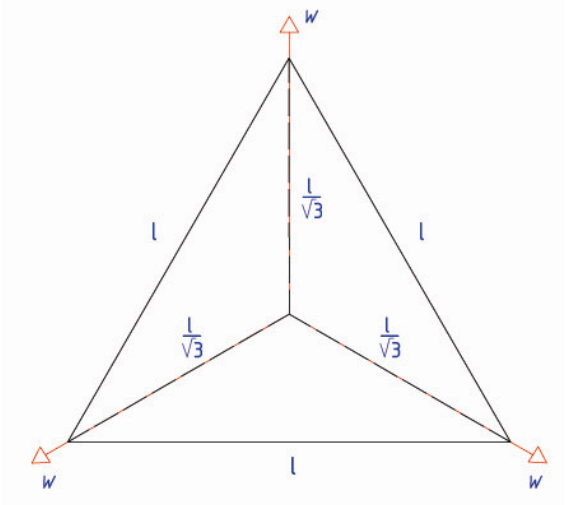
$$V = V_t(1 + f_t/f_c) - k_m/f_c \tag{2.15}$$

$$V = V_c(1 + f_c/f_t) + k_m/f_t \tag{2.16}$$

The volume of the joints is ignored.

Owen then states that since the volume of a structure cannot be negative, these results indicate that when k_m is positive the structure of least volume, if this exists, will be one which has no compression members, i.e. $V_c = 0$, and the least volume will then be k_m/f_t. On the other hand, if k_m is negative, then a structure which has the least volume will be one which has no tension members and that this least volume will then be $-k_m/f_t$. When k_m vanishes a dilemma arises because the theoretical least weight may now be zero, corresponding to V_t or V_c zero. This indicates that structures composed entirely of compression or tension members are no longer possible means of connecting the specified loads. Such structures must then contain a combination of tension and compression members. Owen then uses the concepts of virtual work to give a formula from which it can be observed that a Michell structure will have the least volume.

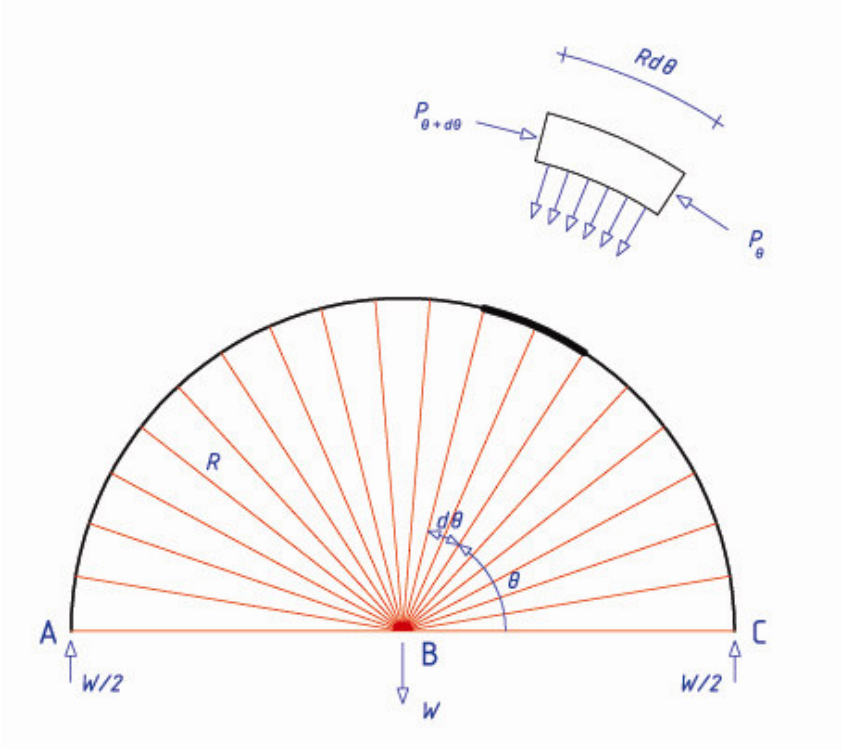
The simplest Michell strain field which can be imagined is that in which all the strains are equal in all directions. A member in any direction in this field will be strained the same amount as any other member. The structure will then be all in compression or all in tension. There is often an infinity of possible 'minimum' structures which will each have the volume $m_m/(f_c \text{ or } f_t)$, and this is the minimum possible volume of structural material to carry the given loads. Any geometry of such bars which maintains a specified set of loads in equilibrium is then as light as any other set of bars or any combination of such bars provided always equilibrium is maintained. Such arrangements of bars can range from mechanisms to highly redundant structures. So, structures as in Figure 2.18, where all the members work to a tensile stress f, are now recognisable as 'minimum' structures.



2.18. A minimum weight structure for concentric loads

a Michell structure of a centrally loaded simply supported beam

In a plane, if one principal strain at a point is extensional and the other compressive, it is known that these strains must be at right angles. A simple set of plane 'curves' which can depict a strain field is that of lines radiating from a point and concentric sectors of circles centred on this point. In this field, if the radial displacement u at a radial distance r is εr and if the tangential displacement v here is $-2\varepsilon r\theta$, where θ is the angle between the radius and some given direction, then the radial strain is $\frac{\partial u}{\partial r} = \varepsilon$, the tangential strain $\frac{u}{r} + \frac{\partial v}{r\partial\theta} = -\varepsilon$, and the shear strains on radial and tangential planes vanish. These radial and tangential extensional strains are then principal strains and this field is of the kind that Michell postulates. Half a 'spoked wheel' with the 'spoke sheet' in a state of tension and the rim in compression will now be shown to require only forces at the central point and at the ends of the rim to maintain equilibrium. This system is illustrated in Figure 2.19.



2.19. A Michell structure for carrying a central load w on two supports

Equilibrium of the forces acting on a length $Rd\theta$ of the rim, requires that along the tangent

$$P_\theta \cos\frac{d\theta}{2} = P_{\theta+d\theta} \cos\frac{d\theta}{2} \qquad 2.17$$

or

$$P_\theta = P_{\theta+d\theta} \qquad 2.18$$

i.e. the compressive force in the rim must be constant. Vertical equilibrium of a small piece of the rim requires

$$P_\theta = W / 2 \qquad 2.19$$

Radial equilibrium of a portion of the rim requires

$$2 \cdot P_\theta \cdot \sin\frac{d\theta}{2} = f_t \cdot R \cdot d\theta \cdot t_R \qquad 2.20$$

i.e.

$$t_R \cdot R = P_\theta / f_t = W / 2f_t \qquad 2.21$$

where t_R is the 'thickness' of the spoke sheet at radius R. Equilibrium of a small sector of the spoke field which is only in radial tension, requires that

$$f_t \cdot rd\theta \cdot t = f_t \cdot (r + dr)d\theta \cdot (t + dt) \qquad 2.22$$

From this, Owen states that for a particular value of θ that

$$t \cdot r = t_R \cdot R \qquad 2.23$$

$$= W / 2f_t \qquad 2.24$$

Analogy with pneumatic structures

At some points it is difficult to determine how Owen came to the results in the different steps. To give better insight, an analogy with pneumatic structures can be made.

The same results can be derived from the analogy with pneumatic structures. The rim under compression is replaced by a pneumatic structure (a membrane, which can only take tensile stresses) and the tensile stress in the spoke field can be replaced with the universal pressure of a gas. The spoked wheel can thus be inverted to a pneumatic structure as in Figure 2.20, where the radius of the half-

circle shaped pneumatic structure is R and the overpressure is p over a length b. The opening angle α is 90°, therefore, the tensile forces per length at the supports (A and B) have only a vertical component, $n_v = n = pR$ (n is the support reaction per length) or $N_v = N = pbR$. This formula (also known as the *kettle formula*[32]) is analogous with formula 2.23.

$$t \cdot r = W / 2f_t \quad \hat{=} \quad N = pbR \qquad 2.25$$

with

t	$\hat{=}$	b
r	$\hat{=}$	R
$W/2$	$\hat{=}$	N
f_t	$\hat{=}$	p

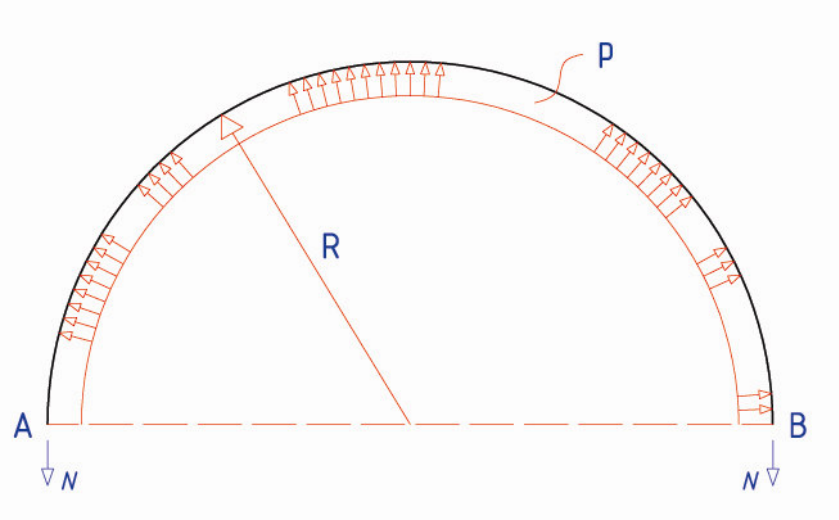


Fig. 2.20. A pneumatic structure analogous to a Michell structure

Returning to the Michell structure, the thickness of the spoke sheet is thus inversely proportional to the radius r and independent of the angular position θ. At the central point B in Figure 2.19, where $r = 0$, the thickness becomes infinite. From the practical viewpoint the load W can never be absolutely concentrated at B so that this infinity is not really disturbing. It requires more imagination to accept the idealisation to two dimensions of a spoke sheet of variable thickness which is really a three dimensional structure. Accepting this, a beam structure for carrying a central load has been derived. It consists of (i) a spoke field subjected to constant radial tensile stress, and (ii) a semi-circumferential rim orthogonal to the spoke field and in constant compression.

[32] Hartsuijker [17]

The members of the structures all lie in the directions of the maximum strains envisaged. This is therefore a Michell structure for carrying a central load. The volume of the spoke field is

$$\int_0^R \int_0^\pi t.r d\theta.dr = \int_0^R \int_0^\pi \frac{W}{2f_t} \cdot d\theta \cdot dr = \frac{W}{2f_t} \cdot \pi \cdot R \qquad 2.26$$

and the volume of the rim is

$$\frac{W}{2f_c} \cdot \pi \cdot R \qquad 2.27$$

2.3 philosophy behind structural optimisation

After all this information on structural optimisation in general, can it be stated that the frequent use of it would give an enormous boost to the building engineering industry? Or is it just a 'hoax'-method? This section deals with the philosophy behind structural optimisation

First, it is known that structural optimisation techniques are effective tools that can be used to improve the performance of structures in terms of the material efficiency in transferring the applied loads. However, the performance of optimised designs is often limited to the optimisation methods used. It is of importance to realise that the formulation of a design problem in structural optimisation significantly affects the results. Incomplete and improper problem formulation may lead to poor or meaningless designs. Some structural optimisation methods use the behavioural quantity such as the compliance for the objective function and a somewhat arbitrarily chosen material volume for the constraint to search for optimal configurations. Optimisation methods based on such a problem formulation may not yield minimum weight designs[33].

Second, some of the optimisation techniques are used to create an optimum structure under a certain load. This means that the size and shape of the elements or the complete structure can only be qualified as 'optimal' for that specific load. For example, consider a Michell structure in Section 2.2.3. When subjected to multiple load cases the geometry of the structure has to be altered severely.

Next to this, many of the publications on the theoretical development and practice of structural optimisation are concerned with mathematical aspects of structural

[33] Hemp [18]

optimisation rather than practical applications. There is a clear gap between the development of structural optimisation theory and its practical applications in the building engineering industry. The main reason for the gap between the theory and practice of structural optimisation is the priority of mathematical over engineering aspects. Because of the mathematical complexity of structural optimisation methods, they remain to be an academic interest. Structural optimisation techniques could become more attractive to practicing building engineers if they are developed not only for saving materials but also for automating the engineering design process. It appears that the gap between structural optimisation theory and its practical applications to building engineering has not been reduced in the last two decades. However, in the beginning of structural optimisation analysis, the computational analysis ran up against two difficulties[34]: The first arises because of the enormous size of realistic design calculations, which in some cases still require more capacity than is available in the computers. The second is more fundamental and arises because non-linear problems can have many local minima or maxima and so there is always some doubt as to whether any solution that is obtained represents an absolute minimum or maximum or not. Nowadays, those two difficulties do not form an obstacle in the usage of structural optimisation analysis anymore.

Considering the pitfalls of structural optimisation analysis as ascribe above, what is the point of using it in the design process?

As an answer to this, it should be stated that structural optimisation methods should be used as design tools, not as an objective in the design process. This way structural optimisation enforces rather than removes the creative aspect of designing, and the final design can be a product of creativity rather than availability or lack of analysis facilities. Structures designed with the concept of optimality need to be subjected to an extra analysis, so that they can be utilised in practise. Optimal structures subjected to given conditions of loading can provide ideal norms against which 'standard' structures or other forms of optimum structures can be measured.

In other words, structural optimisation analysis can help in improving a design when it is used in the preliminary stage of the design process, but when it is used to present structural elements or complete structures based on merely theoretical optima, it could result in a misleading representation of the reality.

Considering all of this, is should be stated that at all times one needs to be aware of the fact that an optimal design is not a perfect design, but the best, most suitable or most desirable possible under a restriction expressed or implied.

[34] Liang [23]

2.3 concluding remarks

The field of structural optimisation is of such size, a complete overview of techniques and methods can never be given in one Master's thesis. Therefore, in this introduction into concepts of structural optimisation in building engineering, some striking examples are presented.
It is shown that optimisation is largely dependant on the boundary conditions set for a given problem, including the contradictorily aspects of the optimal values of each boundary condition. As fixed numerical values for different conditions are hard to predict, structural designers should be aware that, although structural optimisation is best utilised in the conceptual and preliminary design stage, optimisation does not present a finalised and optimal structural design yet. After an optimisation routine, other (maybe unforeseen) structural, functional or economical objectives need to be implemented in the design as well, possibly (or better probably) altering the 'optimised' design.

3 optimisation methods in the field of engineering

3.1 non-computational methods in structural optimisation

As was stated in the introduction, an example of a non-computational optimisation technique that is widely used is physical modelling, or in other words, the modelling of a (structural) design in real life. Examples of this type of modelling are soap film [Figure 3.1] and hanging chains modelling or *funiculars* [Figure 1.1]. And as was stated in the Introduction, by making a physical (scale) model and subjecting it to external loads, a designer can learn about the internal forces in the structural elements and remodel the design accordingly and thus working towards an optimum design.

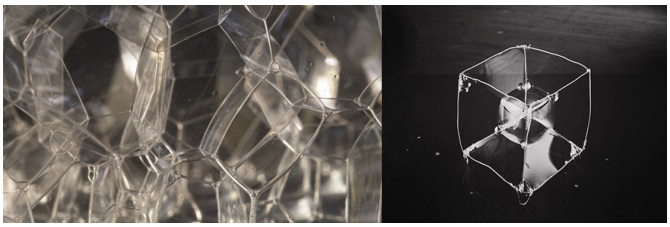


Fig. 3.1. Soap bubble structure [www.msm.cam.ac.uk, November 2006] and a soap film cube [www.yossimilo.com, November 2006]

So, when using an optimisation technique like physical modelling, an engineer is able to adapt the initial design in such a way that it will meet the non-structural boundary conditions, while corresponding to, e.g. compression lines or minimal energy surfaces. One of the main advantages of physical modelling is the insight in the behaviour of the modelled structure without the requirement that the designer has to have special skills in mathematics or computer science[35]. Besides, using a physical model can easily improve the effectiveness of any explanation of the structure resulting from it to someone, layman or not. But two big disadvantages of physical modelling are the amount of time to make a model and the difficulty of adjusting the model. Besides this, a physical model cannot be used to calculate the expected forces on a structure, whereas a computer model is capable of doing so.

[35] Coenders [7]

3.2 computational methods in structural optimisation

3.2.1 classical methods for structural optimisation

Any problem, in which parameter values must be determined, given certain constraints, can be treated as an optimisation problem[36]. The first step in optimisation is to identify a set of parameters, also called decision parameters, an objective function to be minimised and the problem constraints. The objective function gives a lower 'cost' for parameter values that represent a better solution. Restrictions on a solution are called constraints. The constraints of a solution show the values parameters cannot take. Constraints must be expressed in terms of the decision parameters. Some constraints are represented as inequalities and some as equalities. An interesting computational problem, often used to make novices become acquainted with computing is the Travelling Salesman Problem [Appendix B].

nonlinear programming problem

A nonlinear programming problem can be represented in the following manner[37].

Let 'x' represent an n-dimensional design variable vector. Then any design optimisation problem can be stated as follows: find x to

minimise an objective function f(x)

subject to

equality constraints:	$g_h(x) = 0,$	$h = 1$ to p
inequality constraints:	$g_h(x) \leq 0,$	$h = (p+1)$ to q
	$x_i^L \leq x_i \leq x_i^R$	$i = 1$ to k

where p is the number of equality constraints and (q-p) is the number of inequality constraints. x_i^L and x_i^R are the lower and upper bounds on the design variable x_i, and k is the total number of design variables. This optimisation problem is called a general mixed discrete-continuous variable nonlinear optimisation problem. In some situations, there may be two or more objective functions. This is called a *multi-objective optimisation problem*.

[36] Arora [1]
[37] Burns [6]

global and local maxima and minima[38]

The *feasible region* is the set of all solutions to the problem satisfying all the constraints. The *optimal solution* for a minimisation problem is the solution with the smallest 'cost value' in the feasible region. Similarly, for maximisation problems, it is the solution with the largest objective function value. The objective function f(x) has a local minimum (also called a relative minimum) at a point x^* in the feasible set *S* if the function value is the smallest at the point x^* compared to all other points x in a feasible neighbourhood *N* of x^*, that is

$$f(x^*) \leq f(x) \qquad 3.1$$

$\forall$ x (= for all x) in the feasible region.

If the inequality holds, then x^* is called the strict or unique global minimum.

Function *f(x)* has a *local minimum* at x^* if this equation holds for all *x* in a small neighbourhood of *N* of x^* in the feasible region. Neighbourhood *N* of the point x^* is mathematically defined as

$$N = \{x \mid x \in S \text{ with } \| x - x^* \| < \delta\} \qquad 3.2$$

for some small δ. Geometrically, it is a small feasible region containing the point x^*.

The global and local minima and maxima are shown in Figure 3.2.

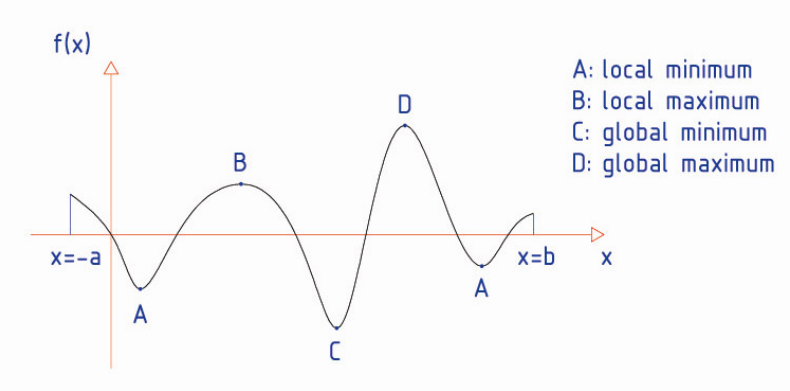


Fig. 3.2. Global and local maximum and minimum points of a multimodal function

[38] Burns [6]

Classification of problems[39]

If an optimisation problem has linear objective and constraint functions, it is called a *linear programming problem*. An *integer programming problem* is a linear programming problem in which some or all variables must be non-negative integers. Otherwise, it is called a *non-integer programming problem*. The search for an optimal arrangement, grouping, ordering or selection of discrete objects is called *combinatorial optimisation*. A problem having a quadratic objective function and linear constraints is called a *quadratic programming problem*.
It is important to note the following points for the foregoing nonlinear problem model relative to structural and mechanical system design problems.

1. the model is applicable to all problems with continuous design variables. *Multi-objective* and *discrete variable* problems can also be treated after certain extensions of the model;
2. the functions of the problem are assumed to be *twice differentiable*. Problems having non-differentiable functions can be treated with additional computational effort. Also, gradients of active constraints are assumed to be linearly independent at the optimum;
3. the objective and/or constraint functions may be *implicit* as well as *explicit* functions of the design variables. That is, their final form in terms of only the design variables may not be known. The functions, however, can be evaluated using analysis computer programs once a design is specified;
4. *derivatives* of the functions are needed in numerical methods of optimisation. Efficient methods to calculate them taking advantage of the structure of engineering design problems have been developed.

attributes of a good algorithm

In computing, and thus in computational optimisation, used is being made of algorithms. An algorithm is a procedure (a finite set of well-defined instructions) for accomplishing some task which, given an initial state, will terminate in a defined end-state. Algorithms often have steps that repeat (iterate) or require decisions and are often graphically represented with flowcharts.

[39] Burns [6]

The attributes of a good algorithm are the following

1. robustness: the algorithm must be reliable for general design applications and thus must be theoretically guaranteed to converge to a solution point starting from any initial design estimate;
2. generality: the algorithm must be general, implying that is should be able o treat equality as well as inequality constraints and should not impose any restrictions on the form of the cost and constraint functions, such as linear or quadratic functions;
3. accuracy: the ability of an algorithm to converge to the precise mathematical optimum point is important even though a precise optimum may not be needed in practical applications. An accurate optimisation algorithm is likely to have a sound mathematical basis and thus a higher reliability;
4. ease of use: the algorithm must be easy to use by experienced as well as inexperienced designers. An algorithm that requires selection of tuning parameters that are problem dependent is difficult to use in practice;
5. efficiency: the algorithm must be efficient for general engineering design applications. To be efficient, the number of repeated analyses of the system (such as finite element analyses) must be kept to a minimum. Thus an efficient algorithm has (i) a faster rate of convergence, requiring fewer numbers of iterations and consequently, fewer system analyses, and (ii) the least number of calculations within one design iteration.

Optimum design of large scale systems presents special challenges for any numerical algorithm in terms of its efficiency. For such systems, the algorithm should be selected cleverly. For example, if an algorithm does not use the potential constraint strategy in defining the search direction determination sub problem, then it is not suitable for large scale problems. Also an algorithm that requires tuning of certain parameters for its proper performance will not be suitable because it may require several trials before proper values for the parameters are found. This is not only inconvenient but also inefficient. Such algorithms should be avoided for optimisation of large scale systems. The algorithm may be slightly less efficient for small scale problems (small problems require very little CPU time any way), but it must be more efficient for large scale problems.

3.2.2 new artificial intelligence methods for structural optimisation

Various methods based on artificial intelligence for the solution of structural optimisation problems are at the hand of the structural designer. This section will deal with six of these methods.

- *genetic algorithms* (GAs) locate optima using processes similar to those in natural selection and genetics;
- *tabu search* is a heuristic procedure that employs dynamically generated constraints or tabus to guide the search for optimum solutions;
- *simulated annealing* finds optima in a way analogous to the reaching of minimum energy configurations in metal annealing;
- *neural networks* are computational models of the brain. Certain types of neural networks can be used for optimisation by exploiting their inherent ability to evolve in the direction of the negative gradient of an energy function and to reach a stable minimum of that function;
- *swarm intelligence* and ant colony optimisation deal with the modelling of social insects by means of theories of self-organisation that can help design artificial distributed problem-solving devices that self-organise to solve problems;
- *fuzzy set* theory provides a mean for representing uncertainties. It is a tool for modelling the kind of uncertainty associated with vagueness, with imprecision, and/or lack of information regarding a particular element of the problem at hand.

genetic algorithms

The basic idea of the method is to start with a randomly generated set of design alternatives using the allowable values of each variable. Each design alternative is represented by a unique finite length binary string of 0's and 1's for binary coding or with real values for real number encoding. This set of designs is called the population in a given generation. Each design is also assigned a fitness value based on the objective function (or in case of genetic algorithms, a *fitness function*). From the current population, a set of designs is selected randomly with a bias allocated to more fit members of the population. Random processes are used to generate a new set of designs for the next generation, until a satisfactory fitness value is reached (in other words, if the termination test is successful) [Figure 3.3, on the next page]. The size of the population of each generation is kept fixed. Since more fit members of the population are used to create new designs, the successive generations have a higher probability of having designs with better fitness values. An advantage of this approach is that continuity and differentiability of functions are not required, as for the simulated annealing method[40].

[40] Burns [6]

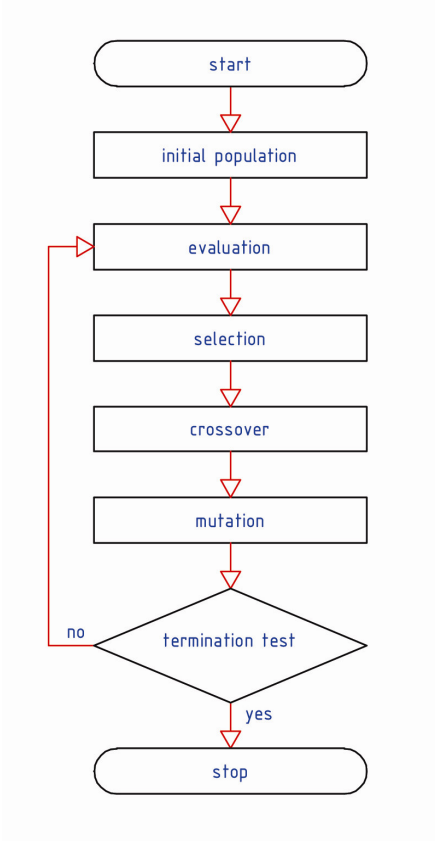


Fig. 3.3. Simple flow chart of genetic algorithm

The most common operators that are needed to implement a genetic algorithm are selection, crossover and mutation. Some of these operators were inspired by nature and, in the literature, many versions of these operators can be found. The choice or design of operators depends on the problem and representation scheme employed. For a detailed introduction into genetic algorithms, see Chapter 4.

The fitness evaluation unit acts as an interface between the GA and the optimisation problem. The GA assesses solutions for their quality according to the information produced by this unit and not by using direct information about their structure. In engineering design problems, functional requirements are specified to the designer who has to produce a structure which performs the desired functions within predetermined constraints. The quality of a proposed solution is usually calculated depending on how well the solution performs the desired functions and satisfies the given constraints. In the case of a GA, this calculation must be automatic and the problem is how to devise a procedure which computes the quality of solutions.

Fitness evaluation functions might be complex or simple depending on the optimisation problem at hand. Where a mathematical equation cannot be formulated for this task, a rule-based procedure can be constructed for use as a fitness function or in some cases both can be combined. Where some constraints are very important and cannot be violated, the structures or solutions which do so can be eliminated in advance by appropriately designing the representation scheme. Alternatively, the can be given low probabilities by using special penalty functions.

Conventional search techniques, such as hill-climbing, are often incapable of optimising non-linear multimodal functions. In such cases, a random search method might be required. However, undirected search techniques are extremely inefficient for large domains. A genetic algorithm (GA) is a directed random search technique, which can find the global optimal solution in complex multi-dimensional search spaces.

GAs do not use much knowledge about the problem to be optimised and do not deal directly with the parameters of the problem. Similar with the ant analogy (see below) the elements are very trivial, but the system is to be non-trivial and robust. They work with codes which represent the parameters[41]. Thus, the first issue in GA application is how to encode the problem under study, i.e. how to represent the problem parameters. GAs operate with a population of possible solutions, not only one possible solution, and the second issue is therefore how to create the initial population of possible solutions. The third issue in a GA application is how to select or devise a suitable set of genetic operators. Finally, as with other search algorithms, GAs have to know the quality of already found solutions to improve them further. Therefore, there is a need for an interface between the problem environment and the GA itself for the GA to have this knowledge. The design of this interface can be regarded as the fourth issue.

At the start of optimisation, a GA requires a group of initial solutions. There are two ways of forming this initial population. The first consists of using randomly produces solutions created by a random number generator. This method is preferred for problems about which no a priori knowledge exists or for assessing the performance of an algorithm. The second method employs a priori knowledge about the given optimisation problem. Using this knowledge, a set of requirements is obtained and solutions which satisfy those requirements are collected to form an initial population. In this case, the GA starts the optimisation with a set of approximately known solutions and therefore converges to an optimal solution in less time than with the previous method.

Important control parameters of a simple GA include the population size (number of individuals in the population), crossover rate and mutation rate. A large population size means the simultaneous handling of many solutions and increases the

[41] Burns [6]

computation time per iteration. However, since many samples from the search space are used, the probability of convergence to a global optimal solution is higher than when using a small population size.

tabu search[42]

The tabu search is a kind of iterative search and is characterised by the use of a flexible memory. It is able to eliminate local minima or maxima and to search areas beyond a local minimum or maximum. Therefore, it has the ability to find the global minimum/maximum of a multimodal search space. The process with which tabu search overcomes the local optimality problem is based on an evaluation function that chooses the highest evaluation solution at each iteration. This means moving to the best admissible solution in the neighbourhood of the current solution in terms of the objective value and tabu restrictions. The evaluation function selects the move that produces the most improvement or the least deterioration in the objective function. A tabu list is employed to store the characteristics of accepted moves so that these characteristics can be used to classify certain moves as taboo (i.e. to be avoided) in later iterations. In other words, the tabu list determines which solutions may be reached by a move from the current solution. Since moves not leading to improvements are accepted in tabu search, it is possible to return to already visited solutions. This might cause a cycling problem to arise. The tabu list is used to overcome this problem. A strategy called the forbidding strategy is employed to control and update the tabu list. By using the forbidding strategy, a path previously visited is avoided and new regions of the search space are explored.

A simple tabu search algorithm consists of three main strategies: forbidding strategy, freeing strategy and short-term strategy. The forbidding strategy controls what enters the tabu list. The freeing strategy controls what exits the list and when. The short-term strategy manages the interplay between the forbidding and freeing strategies to select trial solutions. Apart from these strategies, there can be also a learning strategy which consists in the use of intermediate and long-term memory functions. This strategy collects information during a tabu search run and this information is used to direct the search in subsequent runs.

Tabu search relies on the systematic use of memory to guide the search process. It is common to distinguish between short-term memory, which restricts the neighbourhood of the current solution to a subset, and long-term memory, which may extend the set through the inclusion of additional solutions. Tabu search uses a local search that, at every step, makes the best possible move from a solution to a neighbour solution, even if the new solution is worse than the current one. In this latter case, the move that least worsens the objective function is chosen.

[42] Burns [6]

simulated annealing[43]

Simulated annealing is inspired by an analogy between the physical annealing of solids (crystals) and combinatorial optimisation problems. In the physical annealing process a solid is first melted and then cooled very slowly, spending a long time at low temperatures, to obtain a perfect lattice structure corresponding to a minimum energy state. Simulated annealing transfers this process to local search algorithms for combinatorial optimisation problems. It does so by associating the set of solutions of the problem attached with the states of the physical system, the objective function with the physical energy of the solid, and the optimal solutions with the minimum energy states.

It is a local search strategy which tries to avoid local minima by accepting worse solutions with some probability [Figure 3.4].

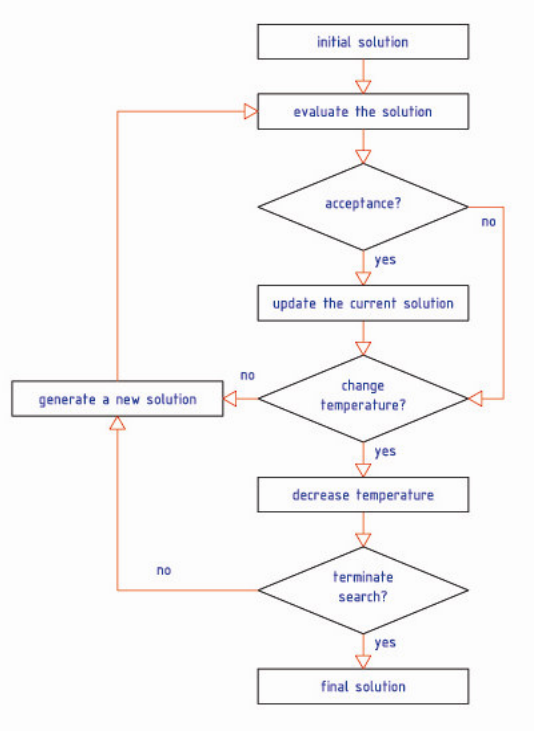


Fig. 3.4. Simple flow chart of simulated annealing

[43] Burns [6]

The algorithm consists of a sequence of iterations. Each iteration consists of randomly changing the current solution to create a new solution in the neighbourhood of the current solution. The neighbourhood is defined by the choice of the generation mechanism. Once a new conclusion is created the corresponding change in the objective function is computed to decide whether the newly produced solution can be accepted as the current solution. If the change in the objective function is negative the newly produced solution is directly taken as the current solution. Otherwise, the current solution is unchanged.

In order to implement the algorithm for a problem, there are four principal choices that must be made.

- representation of solutions
- definition of the objective function
- definition of the generation mechanism for the neighbours
- designing a cooling schedule

Solution representation and objective function definitions are as for GAs. Various generation mechanisms could be developed that again could be borrowed from GAs, for example, mutation and inversion.

In designing the cooling schedule for a simulated annealing algorithm, four parameters must be specified. These are an initial temperature, a temperature update rule, the number of iterations to be performed at each temperature step and a stopping criterion for the search. One example of a cooling schedule is the geometric cooling rule. This rule updates the temperature by the following formula

$$T_{i+1} = cT_i,\ i = 0,1,2,\ldots$$

where c is a temperature factor which is a constant smaller than, but close to 1.

neural networks[44]

Neural networks are modelled on the mechanism of the brain. Theoretically, they have a parallel distributed information processing structure. Two of the major features of neural networks are their ability to learn from examples, and their tolerance to noise and damage to their components.
A neural network consists of a number of simple processing elements, also called nodes, units, short-term memory elements and neurons. The elements are modelled on the biological neuron and perform local information processing operations.
A processing element has one output and several inputs which could be its own output, the output of other processing elements or input signals from external devices. Processing elements are connected to one another via links with weights,

[44] Burns [6]

which represent the strengths of the connections. The weight of a link determines the effect of the output of a neuron on another neuron. It can be considered as part of the long-term memory in a neural network.
After the inputs are received by a neuron, a pre-processing operation is applied. There are several alternative pre-processing procedures including taking the summation, cumulative summation, maximum or product of the weighted inputs. The output of the pre-processing operation is passed through a function called the activation function to produce the final output of the processing element. Depending on the problem, various types of activation functions are employed.

Rather than being seen as an optimisation tool, neural networks are more frequently the object *of* optimisation exercises. This is because the training of a neural network can be regarded as an optimisation problem. Such a problem could be solved by applying optimisation techniques such as GAs and tabu search.

swarm intelligence and ant colony optimisation[45, 46]

The swarm intelligent – the emergent collective intelligence of groups of simple agents – approach emphasises distributedness, direct or indirect interactions among relatively simple agents, flexibility, and robustness. Flexibility allows adaptation to changing environments, while robustness endows the colony with the ability to function even though some individuals may fail to perform their tasks. The number of its successful applications is exponentially growing in combinatorial optimisation, communication networks and robotics. However, it is fair to say that very few applications of swarm intelligence have been developed. One of the main reasons for this relative lack of success resides in the fact that swarm-intelligent systems are hard to program, because the paths to problem solving are not predefined but emergent in these systems and result from interactions among individuals and between individuals and their environment as much as from the behaviours of the individuals themselves. Therefore, using a swarm-intelligent system to solve a problem requires a thorough knowledge not only of what individual behaviours must be implemented, but also of what interactions are needed to produce such or such global behaviour.

The daily problems solved by a colony include finding food, building or extending a nest, efficiently dividing labour among individuals, efficiently feeding the brood, responding to external challenges, spreading alarm, etc. Many of these problems have counterparts in engineering and computer science. One of the most surprising behavioural patterns exhibited by ants is the ability of certain ant species to find what computer scientists call shortest paths and it is this behavioural pattern that inspired computer scientists to develop algorithms for the solution of optimisation problems.

---

[45] Bonabeau [4]
[46] Dorigo [10]

Ants coordinate their activities via *stigmergy*, a form of indirect communication mediated by modifications of the environment, [Grassé, 1959]. Indirect interactions are very subtle: two individuals interact indirectly when one of them modifies the environment and the other responds to the new environment at a later time. For example, a foraging ant deposits a chemical on the ground which increases the probability that other ants will follow the same path. The idea behind ant algorithms is then to use a form of *artificial stigmergy* to coordinate societies of artificial agents. An important insight of early research on ants' behaviour was that most of the communication among individuals, or between individuals and the environment, is based on the use of chemicals produced by the ants. These chemicals are called *pheromones*. By sensing pheromone trails foragers can follow the path to food discovered by other ants.

To test the *autocatalytic* or *positive feedback* process of the self-organising behaviour of the ants three double bridge experimental cases by Deneubourg are executed in 1990. In the first experiment the bridge between the ant nest and a food source consists of two branches of equal length [Figure 3.5]. At the start, ants were free to move between the nest and the food source and the percentage of ants that chose one or the other of the two branches were observed over time. The outcome was that, although in the initial phase random choices occurred, eventually all the ants used the same branch. This result can be explained as follows. When a trial starts there is no pheromone on the two branches. Hence, the ants do not have a preference and they select with the same probability any of the branches. Yet, because of random fluctuations, a few more ants will select one branch over the other. Because ant deposit pheromone while walking, a larger number of ants on a branch results in a larger amount of pheromone on that branch; this larger amount of pheromone in turn stimulates more ants to choose that branch again, and so on until finally the ants converge to one single path.

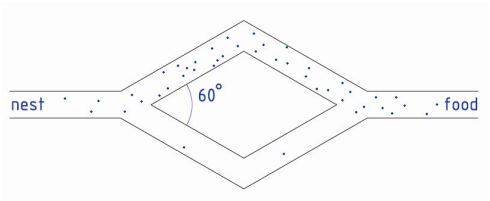


Fig. 3.5. First experiment: two branches of equal length between the nest and the food source

In the second experiment the bridge has two branches, with one branch twice as long as the other branch [Figure 3.6]. In this case, in most of the trials, after some time all the ants chose to use only the short branch. As in the first experiment, ants leave the nest to explore the environment and arrive at a decision point where they have to choose one of the two branches. Because the two branches initially appear identical to the ants, they choose randomly. Therefore, it can be expected that, on average, half of the ants choose the short branch and the other half the long branch, although stochastic oscillations may occasionally favour one branch over the other. However, this experimental setup presents a remarkable difference with respect to the previous one: because one branch is shorter than the other, the ants choosing the short branch are the first to reach the food and to start their return to the nest. But then, when they must make a decision between the short and the long branch, the higher level of pheromone on the short branch will bias their decision in its favour. Therefore, pheromone starts to accumulate faster on the short branch, which will eventually be used by all the ants because of the autocatalytic process. Interestingly, it can be observed that, even when the long branch is twice as long as the short one, not all the ants use the short branch, but a small percentage may take the longer one. This may be interpreted as a type of 'path exploration'.

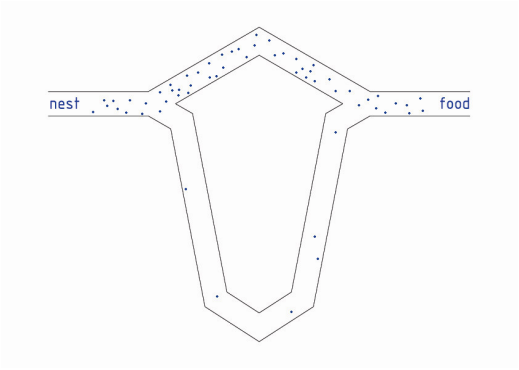


Fig. 3.6. Second experiment: two branches of different length between the nest and the food source

In the third experiment, after convergence over the long branch, the ant colony is offered a new and shorter connection between the nest and the food source [Figure 3.7, on the next page]. This did not affect the number of ants that was using the long branch; the short branch was only selected sporadically. This can be explained by the high pheromone concentration on the long branch and by the slow evaporation of it.

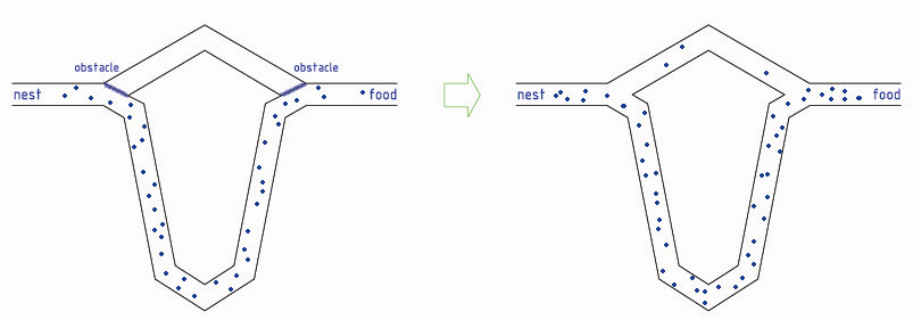


Fig. 3.7. Third experiment: with the removal of the obstacles, the model again consists of two branches with different length between the nest and the food

The double bridge experiments show that ant colonies have a built-in optimisation capability. By the use of probabilistic rules based on local information they can find the shortest path between two points in their environment.

The goal is to define algorithms that can be used to solve optimum problems on complicated graphs, where the optimal path (or the shortest path) between source and destination nodes needs to be determined. Unfortunately, the solving of a complicated graph can result in the following problem: the ants, while building a solution, may generate loops. As a consequence of the forward pheromone trail updating mechanism, loops tend to become more and more attractive and ants can get trapped in them. But even if an ant can escape such loops, the overall pheromone trail distribution becomes such that short paths are no longer favoured and the mechanism that in the simpler double bridge situation made the ant choose the shortest path with higher probability does not work anymore. Because this problem is due to forward pheromone trail updating, it might seem that the simplest solution to this problem would be the removal of the forward updating mechanism. In this way ants would rely only on backward updating.
Still, this is not a solution. If the forward update is removed, the system does not work anymore, not even in the simple case of the double bridge experiment. Therefore, it is needed to extend the capabilities of the artificial ants in a way that, while retaining the most important characteristics of real ants, allows them to solve optimum path problems on generic graphs. In particular, artificial ants are given a limited form of memory in which they can store the partial paths they have followed so far, as well as the 'cost' of the links they have traversed. In experiments with foraging ants, it was shown that the pheromone evaporation rate is so slow compared to the time necessary for the ant colony to converge to the short path that, for modelling purposes, it can be neglected. When considering artificial ants things are different. Experimental results show that on very simple graphs, like the ones modelling the double bridge or the extended double bridge

setups, pheromone evaporation is also not necessary. On the contrary, it improves the algorithm's performance in finding good solutions to the optimal path problem on more complex graphs.

The experiments with the ant algorithm method results in four conclusions

1. the differential path length effect, although important, is not enough to allow the effective solution of large optimisation problems;
2. pheromone updates based on solution quality are important for fast convergence;
3. the larger the number of ants, the better the convergence behaviour of the algorithm, although this comes at the cost of longer simulation times;
4. pheromone evaporation is important when trying to solve more complex problems

In the model of Deneubourg, the probability of choosing a branch at a certain time depends on the total number of ants that used the branch until that time. It is assumed that the amount of pheromone on a branch is proportional to the number of ants that used the branch to cross the bridge. With this assumption, pheromone evaporation is not taken into account: this is a plausible assumption, because the experiments typically last of the order of an hour, a time scale that may not be sufficient for the amount of pheromone to be reduced significantly.

So, in conclusion, the ant colony optimisation approach turns out to be more than just a fun metaphor. Recent developments, which combine the ant colony approach with local searches and/or other optimisation methods, are promising. What is the basic idea underlying all ant-based optimisation? It is to use a *positive feedback* mechanism, based on an analogy with the trail-laying, trail-following behaviour of some species of ants and some other social insects, to reinforce those portions of good solutions that contribute to the quality of these solutions, or to directly reinforce good solutions. A virtual pheromone, used as reinforcement, allows good solutions to be kept in memory, from where they can be used to make up better solutions. Of course, one needs to avoid some good, but not very good, solutions becoming reinforced to the point where they constrain the search too much, leasing to a premature convergence (*stagnation*) of the algorithm. To avoid that, a form of *negative feedback* is implemented through pheromone evaporation, which includes a time scale into the algorithm. This time scale must not be too large, otherwise suboptimal premature convergence behaviour can occur. But it must not be too short either, or otherwise no *cooperative behaviour* can emerge. Cooperative behaviour is the other important concept here: ant colony algorithms make use of the simultaneous exploration of different solutions by a collection of identical ants. Ants that perform well at a given iteration influence the exploration of ants in future iterations. Because ants explore different solutions, the resulting pheromone trail is the consequence of different perspectives on the space of solutions. Even when only the best performing ant is allowed to reinforce its solution, there is a

cooperative effect across time because ants in the next iteration use the pheromone trail to guide their exploration.

fuzzy set theory[47]

From an early age people are educated to think in a classical logic way, that is, to think in terms of true or false proportions. But fuzzy logic requires thinking in other terms than the (quasi-) classical logical, which is usually employed with the aid of various linguistic devices when the issues are not clear-cut. To clarify the meaning, it is emphasised that it is not the logic itself that is fuzzy, but rather it is the information to which it is applied, the logic is actually multi-valued and therefore more flexible than classical logic, hence providing a language environment which is not antagonistic to fundamentally vague or imprecise information.

A useful way of illustrating fuzzy logic sets is shown in Figure 3.8, in which the membership (μ), $0 \leq \mu \leq 1$, is shown on the vertical axis for a continuous distribution of x. Let the elements of the set be defined on the variable x and let μ represent the membership degree of the set. In case *A* of the classical logic set, for $0 < x < x_1$ membership of the set is zero. For $x > x_1$ the membership value is one until $x = x_4$. In contrast to this for the fuzzy logic set, for $0 < x < x_1$ membership is again zero, but for $x_1 < x < x_2$ the membership value gradually changes from zero to one at $x = x_2$. It remains at that value until $x = x_3$ when it commences to fall again (not necessarily at the same rate). Thus there is a gradual transition in this case from zero to full membership of the set and then back to zero again.

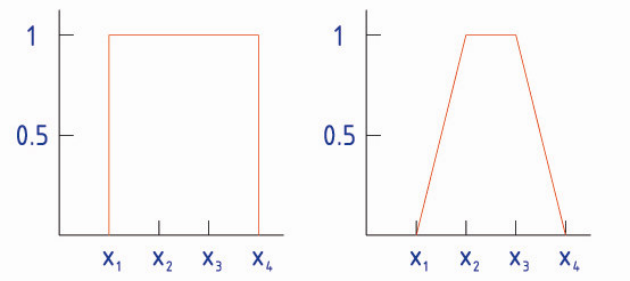


Fig. 3.8. Classical logic way of thinking (left) versus fuzzy logic thinking (right)

[47] Harris [16]

Example

Suppose that x represents temperature. Let x_1 = 35°C and x_2 = 55°C. Let the fuzzy set represent HOT.

Then	$0 < x < x_1$	*represents NOT HOT*
	$x_1 < x < x_2$	*represents HOT to grade μ*
	$x_2 < x < x_3$	*represents HOT*

A temperature of 50°C represents HOT to grade 0,75 and a temperature of 40°C represents HOT to grade 0,25.

In this example there is clearly a need to explain the treatment of a temperature below x_1 and above x_3. This by increasing the number of linguistic terms to include, say WARM and VERY HOT. A corresponding fuzzy set diagram is illustrated in Figure 3.9 on the next page. It may be noted that there are now overlapping sets and that temperature x has a membership grade of μ_1 WARM and μ_2 HOT, that is, it is predominantly HOT, but also WARM. It is not VERY HOT. Thus the need to make a categorical statement whether temperature x is HOT or WARM is avoided. This is achieved by partitioning the universe of discourse into several overlapping sets. The universe of discourse is the range of all the operating sets. Thus in fuzzy logic there is accommodation for uncertainty and it will be clearly seen that in this example it is not of a statistical nature.

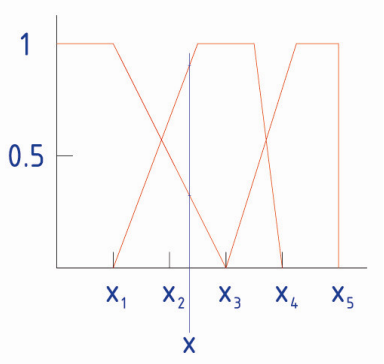


Fig. 3.9. A fuzzy set diagram for three sets

In many real cases, uncertainty is present in observations, which is of a non-random nature. This is where, for example, in engineering a factor of safety would be introduced. The uncertainty may be due to the complex nature of the problem, such as the stress distribution in a component of complex geometry. These are not uncertainties of a statistical nature and corresponding concept is of possibility rather than probability.

It is this category of problems where treatment on a basis of fuzzy logic is fruitful. As more knowledge about a system is accumulated and the uncertainty diminishes the need for a fuzzy logic treatment also diminishes and it can revert to a deterministic or statistical one.

Logical operators
Useful classical logic operations are: 'and' and 'or'. 'And' is called the intersection $\cap$. 'Or' is called the union $\cup$.

Venn diagrams are very useful as an aid to an intuitive approach to logic problems. The convention is that the surrounding rectangle of a diagram represents the universe of all the sets of the genre. Figures inscribed within the rectangle define arbitrary sets. A Venn diagram with two overlapping sets would appear as illustrated in Figure 3.10.

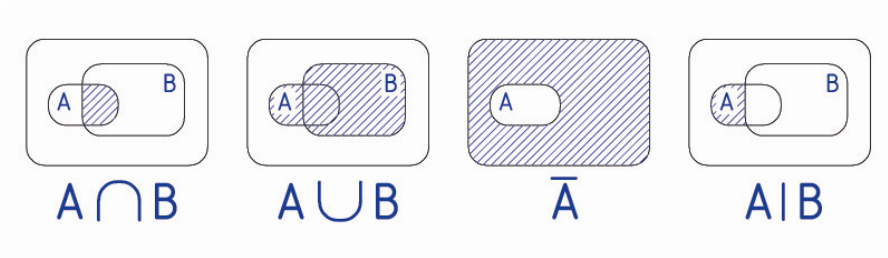


Fig. 3.10. Venn diagrams in classical logic

The boundaries of the sets are not sharp as in the classical logic case. There is a degree of vagueness. A Venn diagram with two overlapping fuzzy logic sets would appear as illustrated in Figure 3.11.

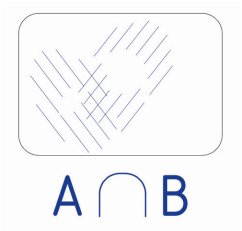


Fig. 3.11. A fuzzy Venn diagram

The integration of fuzzy logic with neural networks and genetic algorithms is now making automated cognitive systems a reality in many disciplines. In fact, the reasoning power of fuzzy systems, when integrated with the learning capabilities of artificial neural networks and genetic algorithms, is responsible for new commercial products and processes that are reasonably effective cognitive systems (i.e., systems that can learn *and* reason).

So, in conclusion, in classical or crisp sets the transition for an element in the universe between membership and non-membership in a give set is abrupt and well-defined (or crisp). For an element in a universe that contains fuzzy sets, this transition can be gradual. This transition among various degrees of membership can be thought of as conforming to the fact that the boundaries of the fuzzy sets are vague and ambiguous. Hence, membership of an element from the universe in this set is measured by a function that attempts to describe vagueness and ambiguity.
A fuzzy set, then, is a set containing elements that have varying degrees of membership in the set. This idea is in contrast with classical, or crisp, sets because members of a crisp set would not be members unless their membership was full, or complete, in that set (i.e. there membership is assigned a value of 1). Elements in a fuzzy set, because their membership need not be complete, can also be members of other fuzzy sets on the same universe.

3.3 concluding remarks

Besides widely accepted and used non-computational methods, the building practise experienced more familiarity with computational methods over the last two decades. Two important aspects of these methods are the run time 'costs' and the ability of locating global optima, something especially new artificial intelligence methods, such as genetic algorithms, simulated annealing, and swarm intelligence perform well on. These three methods including another set of three AI methods are presented in this chapter. From these six methods, genetic algorithms are chosen as the method to be utilised for the VBA script for this Master's thesis.

4 characteristics of genetic algorithms

The genetic algorithm (GA) mechanism possesses the unique ability to search and optimise a solution for a complex system, where other mathematical oriented techniques may have failed to compile the necessary design specifications[48]. But it has its disadvantages as well. Due to its evolutionary characteristics, a standard GA may not be flexible enough, and an engineering insight is always required whenever a GA is applied. This becomes more apparent where the problem to be tackled is complicated, multi-objective and conflicting. But for the problem to be dealt with in this Master's thesis, GA is expected to be able to stand up to it.

In Section 4.1 an introduction into genetic algorithms (GAs) is given, followed by the explanation of some relating terms in Section 4.2. Section 4.3 deals with the main operators of genetic algorithms and the operation possibilities of them. This chapter is concluded by some background information on counting.

4.1 introduction of genetic algorithms

4.1.1 history and background of evolutionary optimisation

Evolution is, in effect, a method of searching among an enormous number of possibilities for 'solutions' – the enormous set of possibilities is the set of possible genetic sequences, and the desired 'solutions' are highly fit organisms – , that are well able to survive and reproduce in their environment. Many computational problems require searching through a huge number of possibilities for solutions.

Since the 50s and 60s of the previous century, computer scientists have studied evolutionary systems with the idea that evolution could be used as an optimisation tool for engineering problems. The idea in all these systems was to evolve a population of candidate solutions (i.e. the search space) to a given problem, using operators inspired by natural genetic variation and natural selection[49].

[48] Man [25]
[49] Mitchell [26]

Genetic algorithms were 'invented' by John Holland in the 1960s and further developed in the following decade, although today, researchers often use the term 'genetic algorithm' to describe something very far from Holland's original conception. The original motivation for John Holland for developing GAs was to construct a theoretical framework for adaptation as seen in nature, and to apply it to the design of artificial adaptive systems. According to Holland, "an adaptive system must persistently identify, test, and incorporate structural properties hypothesised to give better performance in some environment".

Evolution is an optimisation process, where the aim is to improve the ability of individuals to survive, or in other words the main concept is *survival of the fittest*.

The evolutionary search process is influenced by the following main components of evolutionary algorithms[50]:

- an *encoding* of solutions to the problem as a chromosome;
- a function to evaluate the *fitness*, or survival strength of individuals, the *object function*;
- *initialisation* of the initial population;
- *selection* and *reproduction* operators.

Evolutionary computing has as its objective to model the natural evolution. In natural evolution, survival is achieved through reproduction. Offspring, reproduced from two parents, contain genetic material of both parents. Those individuals that inherit bad characteristics are weak and lose the battle to survive. In evolutionary computing a population of individuals is modelled, where an individual is referred to as a *chromosome*. A chromosome defines the characteristics of individuals in the population. Each characteristic is referred to as a *gene* (so a gene could be the angle, thickness or location of a column, together forming a set of gene values, or one chromosome). The value of a gene is referred to as an *allele*. For each generation, individuals compete to reproduce offspring. Those individuals with the best survival capabilities have the best chance to reproduce. Offspring is generated by combining parts of the parents, a process referred to as *crossover* [Section 4.3]. Each individual in the population can also undergo *mutation* [Section 4.3] which alters some of the allele of the chromosome. The survival strength of an individual is measured using a *fitness function* which reflects the objectives and constraints of the problem to be solved. After each generation, individuals may undergo *culling* (destroying of an individual), or individuals may survive to the next generation, referred to as *elitism*. In other words, with elitism the GA is forced to retain some number of the best individuals at each generation. Additionally, behavioural characteristics, as encapsulated in *phenotypes* can be use to influence the evolutionary process in two ways: phenotypes may influence genetic changes,

[50] Engelbrecht [12]

and/or behavioural characteristics evolve separately[51]. Definitions of different terms mentioned in the text above are given in Section 4.2

Different classes of evolutionary computing algorithms have been developed[52]:

- genetic algorithms
- genetic programming
- evolutionary programming
- evolution strategies
- differential evolution
- cultural evolution
- co-evolution

For the aspects of the different classes, other than genetic algorithms, reference is made to Engelbrecht [12].

4.1.2 translation from evolutionary to computational problems[53]

Large search problems can often benefit from an effective use of *parallelism*, in which many different possibilities are explored simultaneously in an efficient way. For a simultaneous evaluation, both computational parallelism (i.e., many processors evaluating sequences at the same time) and an intelligent strategy for choosing the next set of sequences to evaluate are needed. As the fitness criteria continually change as the organisms evolve, evolution is searching a constantly changing set of possibilities, and adaptivity – i.e. the ability to continue to perform well in a changing environment – can thus be seen as an important requirement for computer programs. Other problems require computer programs to be *innovative* – i.e. to construct something truly new and original.

Viewed from a high level the 'rules' of evolution are remarkably simple: species evolve by means of random, followed by natural selection in which the fittest tend to survive and reproduce, thus propagating their genetic material to future generations.

As a numerical optimiser, the solutions obtained by the GA are not mathematically oriented. Instead, the GA possesses flexibility and the freedom to choose desirable optima according to design specifications. In general, the bit string encoding (0s and 1s forming a string that contains the information about the chromosome it represents) is the most classic method used by GA researchers because of its

[51] Sarker [30], Xie [34]
[52] Engelbrecht [12]
[53] Mitchell [26]

simplicity and traceability. The conventional GA operations and theory (scheme theory) are also developed on the basis of this fundamental structure. Hence, this representation is adopted in many applications, even though the real number coding and the Gray code representation [Section 4.4] work (slightly) better than the 'normal' binary representation[54].

4.1.3 biological inspiration of evolutionary algorithms

similarities with DNA

Because of the uniqueness of the evolutionary process and the gene structure of a chromosome, the GA processing mechanism can take the form of parallelism and multiobjective. The fundamental unit of information in living systems is the gene. In general, a gene is defined as a portion of a chromosome that determines or affects a single character or *phenotype*: i.e. the visible property (for example, eye colour, hair colour – or the thickness of a column). It comprises a segment of deoxyribonucleic acid (DNA), commonly packaged into structures called chromosomes. This genetic information is capable of producing a functional biological product which is most often a protein [Figure 4.1].

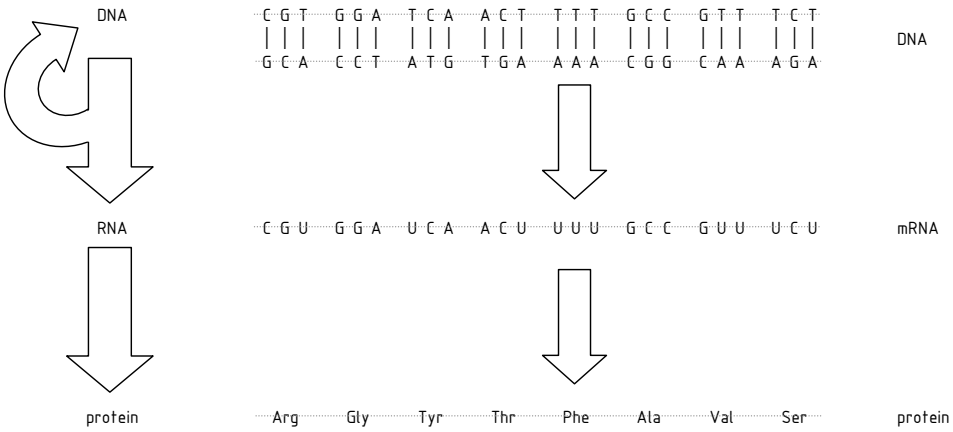


Fig. 4.1. A flowchart from DNA to protein

[54] Man [25]

conventional genetic algorithm[55]

Throughout a genetic evolution, the fitter chromosome has a tendency to yield good quality offspring, which means a better solution to the given problem. In a practical GA application, a population pool of chromosomes has to be installed and these can be randomly set initially. The size of this population varies from one problem to another. In each cycle of genetic operation, termed as an evolving process, a subsequent generation is created from the chromosomes in the current population. This can only succeed if a group of these chromosomes, generally called 'parents' or a collection term 'mating pool' is selected via a specific selection routine. The genes of the parents are mixed and recombined for the production of offspring in the next generation. It is expected that from this process of evolution (manipulation of genes), the fitter chromosome will create a larger number of offspring, and thus has a higher chance of surviving in the subsequent generation, emulating the survival-of-the-fittest mechanism in nature. Figure 4.2 shows the GA cycle.

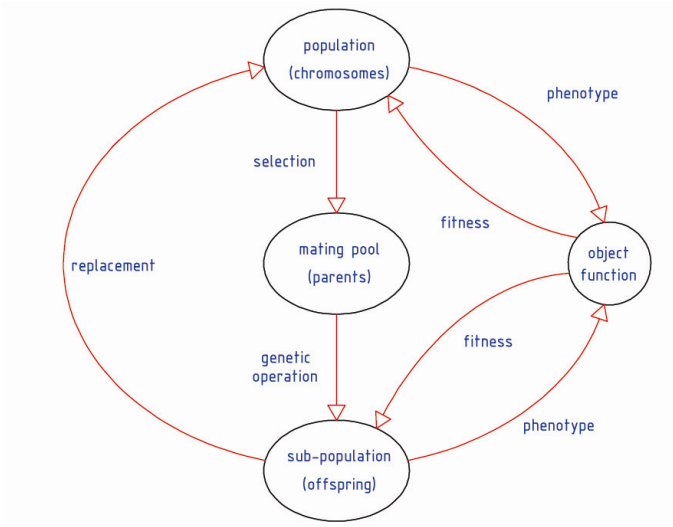


Fig. 4.2. The cycle of genetic algorithms

The cycle of evolution is repeated until a desired termination criterion is reached. This criterion can also be set by the number of evolution cycles (computational runs), or the amount of variation of individuals between different generations, or a pre-defined value of fitness.

[55] Man [25]

4.2 terminology of genetic algorithms

Some terms concerning genetic algorithms are already mentioned in the text above and some will be added in the following texts. Below these terms[56] will be provided with a simple definition.

chromosomes	the strings of DNA that serve as a 'blueprint' for the organism or individual
genes	the functional blocks of DNA each of which encodes a particular protein
alleles	the different possible 'settings' for a trait (e.g., blue, brown or green eyes)
locus	the position of a gene on the chromosome
genome	the complete collection of genetic material (all chromosomes taken together)
genotype	the particular set of genes contained in a chromosome
phenotype	the physical and mental characteristics of an organism
fitness	the probability that an organism will live to reproduce

An objective function or fitness function is a measuring mechanism that is used to evaluate the status of a chromosome. This is a very important link to relate the GA and the system concerned.

4.3 operation and operators of genetic algorithms

4.3.1 operation of genetic algorithms

The conventional theory of GAs assumes that, at a very general level of description, GAs work by discovering, emphasizing, and recombining good 'building blocks' of solutions in a highly parallel fashion. A simply GA has difficulty in tackling complicated, multi-tasking and conflicting problems, and the speed of

[56] Mitchell [26]

computation is generally regarded as slow. On the other hand, there are several reasons that make GA powerful optimisers. GAs are[57];

- very easy-to-understand for reaching a solution;
- general applicable;
- relatively good in finding global optima;
- not dependent of hard mathematical formulations;
- addressable to multi-objective problems;
- capable of handling problem with constraints.

representation[58]

The characteristics represented by a chromosome can be divided into classes of evolutionary information: *genotypes* and *phenotypes*. A genotype describes the genetic composition of an individual as inherited from its parents. Genotypes provide a mechanism to store experiential evidence as gathered by parents. A phenotype is the expressed behavioural traits of an individual in a specific environment. A complex relationship can exist between the genotype and the phenotype. Two such relationships are:

- *pleiotropy*, where random modification of genes cause unexpected variations in the phenotypic traits;
- *polygeny*, where several genes interact to produce a specific phenotype trait. To change this behavioural characteristic, all the associated genes need to change.

As was stated before, each chromosome represents a point in search space. A chromosome consists of a number of genes, where the gene is the functional unit of inheritance. Each gene represents one characteristic of the individual, with the value of each gene referred to as an *allele*. In terms of optimisation, a gene represents one parameter of the optimisation problem.

A very important step in the design of an evolutionary algorithm is to find an appropriate chromosome representation. The efficiency and complexity of a search algorithm greatly depend on the representation scheme, where classical optimisation techniques usually use vectors of real numbers, different EAs use different representation schemes. For example, GAs mostly use a binary string representation, whereas the binary values may represent Boolean values, integers or even (discretised) real numbers.

[57] Mitchell [26]
[58] Engelbrecht [12]

The original GAs developed by Holland had distinctive features[59]:

1. a bit string representation
2. proportional selection
3. crossover as the primary method to produce new individuals

The classical representation scheme for GAs is binary vectors of fixed lengths (e.g., 10001101111). In the case of an /-dimensional search space, each individual consists of / variables with each variable encoded as a bit string. If variables have binary values, the length of each chromosome is / bits.

While binary coding is frequently used, it has the disadvantage of introducing Hamming cliffs [Fig. 4.3 on the next page]. A Hamming cliff is formed when two numerically adjacent values have bit representations that are far apart. The Hamming distance is the number of corresponding bits that differ. For example, consider the decimal numbers 7 and 8. The corresponding binary representations have a Hamming distance of 4 [Table 4.1].

	Binary coding	*Gray coding*
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111

Table 4.1. Binary and (reflected) Gray coding using a 4-bit representation

This presents a problem when a small change in variables should result in a small change if fitness.

[59] Engelbrecht [12[

If, for example, 7 represents the optimal solution, and the current best solution has a fitness of 8, many bits (four to be precise) need to be changed to cause a small change in fitness value. An alternative bit representation is to use Gray coding, where the Hamming distance between the representation of successive numerical values is one [Figure 4.3].

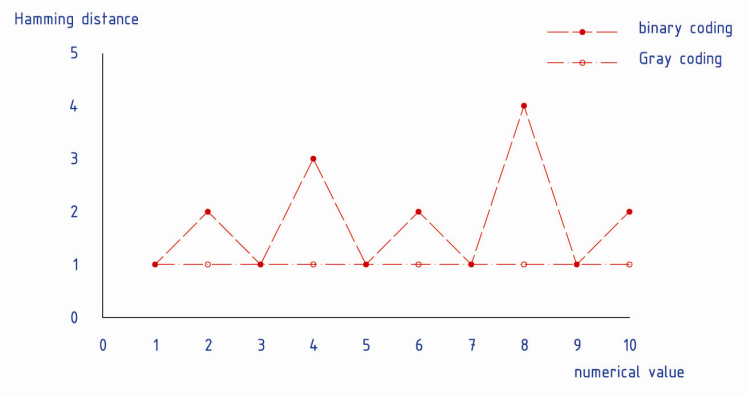


Fig. 4.3. A graph of the Hamming distance for binary coding and Gray coding

fitness function

The fitness function is possibly the most important component of a GA. The purpose of the fitness function is to map a chromosome representation into a scalar value[60].

Since each chromosome represents a potential solution, the evaluation of the fitness function quantifies the quality of that chromosome, i.e. how close the solution is to the optimal solution. Selection, crossover, and mutation operators make use of the fitness evaluation of chromosomes. For example, selection operators use the fitness evaluation to decide which the best parents to reproduce are. Also, the probability of an individual to be mutated can be a function of its fitness: highly fit individuals should preferably not be mutated.

It is therefore extremely important that the fitness function accurately models the optimisation problem. The fitness function should include all criteria to be optimised. In addition to optimisation criteria, the fitness function can also reflect the constraints of the problem through penalisation of those individuals that violate constraints. It is not required that the constraints are encapsulated within the fitness function; constraints can also be incorporated in the initialisation, reproduction and mutation operators.

[60] Engelbrecht [12]

initial population

Before the evolutionary process can start, an initial population has to be generated. The standard way of generating the initial population is to choose gene values randomly from the allowed set of values. The goal of random selection is to ensure that the initial population is a uniform representation of the entire search space. If prior knowledge about the search space and the problem is available, heuristics can be used to bias the initial population toward potentially good solutions.

The size of the initial population has consequences for performance in terms of accuracy and time to converge. A small population represents a small part of the search space. While the time complexity per generation is low, the GA may need more generations to converge than is needed for a large population. On the other hand, a large population covers a larger area of the search space, and may require fewer generations to converge. However, the time complexity per generation is increased. In case of a small population, the GA can be forced to explore a larger search space by increasing the rate of mutation.

fundamental procedures of genetic algorithms

As said before, genetic algorithms start with an initial population of individuals generated at random, and each individual in the population represents a potential solution to the problem under consideration. The individual evolve through successive iterations, called generations. During each generation, each individual in the population is evaluated using some measure of fitness. Then the population of the next generation is created through genetic operators. The procedure continues until the termination condition is satisfied [Figure 4.4, , see also Figure 3.3].

The general framework of genetic algorithms is described as follows[61]:

```
begin
        t:=0;
        initialise P(t);
        evaluate P(t);
        while (not termination condition) do
        begin
                t:=t+1;
                select P(t) from P(t-1);
                alter P(t);
                evaluate P(t);
        end
end
```

[61] Sarker [30]

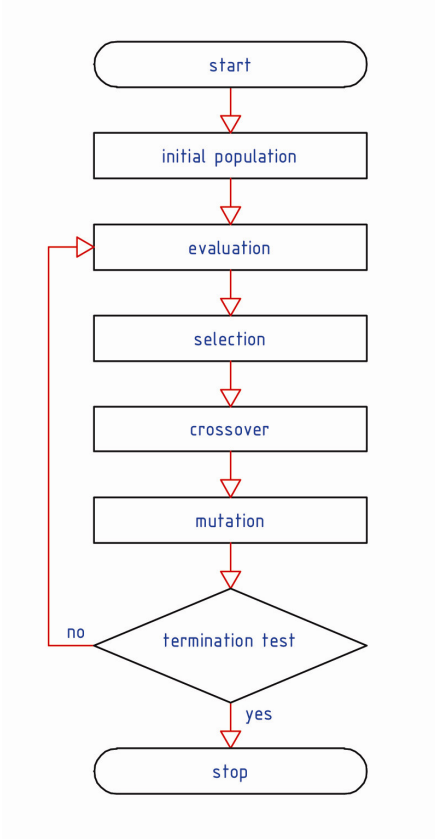


Fig. 4.4. The flowchart of genetic algorithms

Each iteration of this process is called a *generation*. A GA is typically iterated for anywhere from 50 to 500 or more generations. The entire set of generations is called a *run*. At the end of the run, there is often one of more highly fit chromosomes in the population. Since randomness plays a large role in each run, two runs with different random-number seeds will generally produce different detailed behaviours. That is why GA researchers often report statistics averaged over many different runs of the GA on the same problem.

4.3.2 operators in the genetic algorithm process

One common application of GAs is function optimisation, where the goal is to find a set of parameter values that maximise (or minimise) a complex multi-parameter function.

As mentioned before, three basic operators are needed to implement a simple genetic algorithm[62]:

1. *selection:* this operator selects chromosomes in the population for reproduction. The fitter the chromosome, the more times it is likely to be selected to reproduce;
2. *crossover:* corresponds to allowing selected members of the population to exchange characteristics of the design among themselves. Crossover entails selection of starting and ending positions on a pair of mating strings at random, and simply exchanging the string between these positions. The crossover operator roughly mimics biological recombination between two single-chromosome organisms;
3. *mutation:* is the third step that safeguards the process from a complete premature loss of valuable genetic material during reproduction and crossover. In terms of a binary string, this step corresponds to selection of a few members of the population, determining a location on the strings at random, and switching the 0 to 1 or vice versa. Mutation can occur at each gene position in a string with some probability, usually very small.

In this section, these three basic operators, including some others are explained.

selection operators

Each generation of an evolutionary algorithm produces a new generation of individuals, representing a set of new potential solutions to the optimisation problem. The aim of the selection operator is to emphasise better solutions in a population.

In the case of crossover, 'superior' individuals should have more opportunities to reproduce. In doing so, the offspring contains combinations of the genetic material of the best individuals. The next generation is therefore strongly influenced by the genes of the fitter individuals. In case of mutation, fitness values can be used to select only those individuals with the lowest fitness values to be mutated. The idea is that the fittest individuals should not be distorted through application of mutation – thereby ensuring that the good characteristics of the fit individuals persevere. Elitism is an operator that copies a set of the best individuals to the next generation, hence ensuring that the maximum fitness value does not decrease from one generation to the next. Selection operators are used to select these elitist individuals. By limiting the number of offspring that a single individual may produce, limitation of the diversity in the new population can be prevented[63].

[62] Burns [6]
[63] Engelbrecht [12]

There are two classes of selecting techniques, of which no method is superior to the other[64]:

- *explicit fitness remapping*, where the fitness values of each individual is mapped into a new range, e.g. normalisation to the range [0,1]. The mapped value is then used for selection;
- *implicit fitness remapping*, where the actual fitness values of individuals are used for selection.

The most frequently used selection operators are given below[65]:

- random selection
  individuals are selected randomly (all individuals have an equal chance of being selected), with no reference to fitness at all;
- proportional selection
  the chance of individuals being selected is proportional to the fitness values. It is possible that the population can be dominated by a few individuals with high fitness, having a narrow range of fitness values. Similar fitness values are then assigned to a large set of individuals in the population, leading to a loss in the emphasis toward better individuals, e.g. Roulette Wheel Selection;
- tournament selection
  a group of k individuals is randomly selected. These k individuals then take part in a tournament, i.e. the individual with the best fitness is selected. So for crossover, two individuals with the best fitness are selected from a random group k and l;
- rank-based selection
  rank-based selection uses the rank ordering of the fitness values (so, independent of the actual fitness value) to determine the probability of selection;
- elitism
  elitism involves the selection of a set of fit individuals from the current generation to survive to the next generation.

Elitism is an addition to many selection operators. It can be compared with cloning and although in real life, there is a lot of (social) resistance against cloning, for genetic algorithms elitism is a strong and important selection operator.

In elitism, if the fitness of an individual in the previous population is larger than that of every individual in the current population, this individual is preserved into the current generation. Else such individuals can be lost if they are not selected to reproduce or if they are destroyed by crossover or mutation. Introducing elitism,

[64] Burns [6]
[65] Burns [6]

the best individual generated up to generation t can be included in the population at generation $t + 1$. Many researchers have found that elitism significantly improves the GAs performance[66].

A scheme called Roulette Wheel Selection is one of the most common techniques being used for a proportionate selection mechanism[67] as described above. The mechanism works as follows.

The ith member in the original population has a probability of selection

$$P_i = \frac{F_i}{\sum_{1}^{j} F_i} \qquad 4.1$$

with j = 1,...,N_p, where F_i is the fitness of the ith design. The circumference of the roulette wheel in Figure 4.5, on the next page is F_{Np} for N_p chromosomes. Chromosome 2 is fitter than chromosome 4, and occupies the largest interval. To select a chromosome, a random number is generated in the interval [0, F_{Np}] and the individual whose segment spans the random number is selected.

The algorithm of the roulette selection is summarised as follows[68].

step 1.
: calculate the fitness F, i = 1,...,N_P, of N_p individuals and their whole sum F_{Np} in a population at generation t

step 2.
: generate a real random number `rand()` in [0,1] and set s = `rand()` x F_{Np}

step 3.
: obtain the minimal k such that $\sum_{i=1}^{N_P} F_i \geq s$, and select the kth individual at generation $t + 1$

step 4.
: repeat steps 2 and 3 until the number of selected individuals becomes N_p

[66] Mitchell [26]
[67] Man [25]
[68] Man [25]

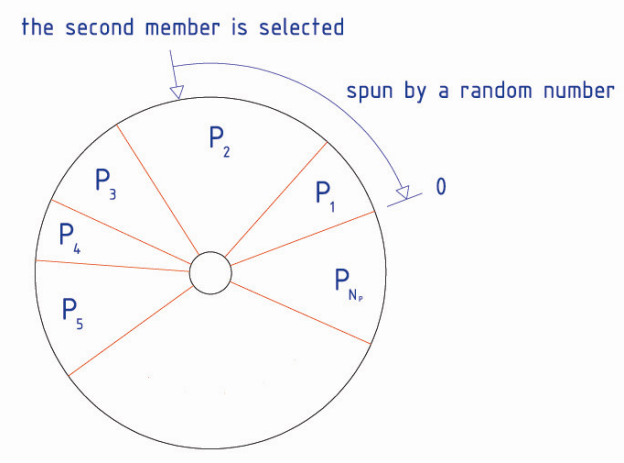


Fig. 4.5. The roulette wheel scheme for genetic algorithms

reproduction operators

The purpose of reproduction operators is to produce new offspring from selected individuals, either through crossover of mutation. Crossover is the process of creating a new individual through the combination of the genetic material of two parents. Mutation is the process of randomly changing the values of genes in a chromosome. The aim of mutation is to introduce new genetic material into an existing individual, thereby enlarging the search space.

crossover

As mentioned before, the aim of crossover is to produce offspring from two parents, selected using a selection operator. However, it is not necessary that each group of parents produces offspring. In fact, crossover takes place at a certain probability, referred to as the *crossover rate* $p_c \in [0,1]$[69].

[69] Engelbrecht [12]

A mask specifies which bits of the parents should be swapped to generate offspring. Several crossover operators have been developed to compute the mask [Figure 4.6]. Some of these crossover operators are;

- uniform crossover
  for uniform crossover the mask of length *l* is created at random for each pair of individuals selected for reproduction. A bit with value of 1 indicates that the corresponding allele have to be swapped between the two parents;
- one-point crossover
  a single bit position is randomly selected and the bit substrings after that point are swapped between the two chromosomes;
- two-point crossover
  two bit positions are randomly selected and the bit substrings between these points are swapped.

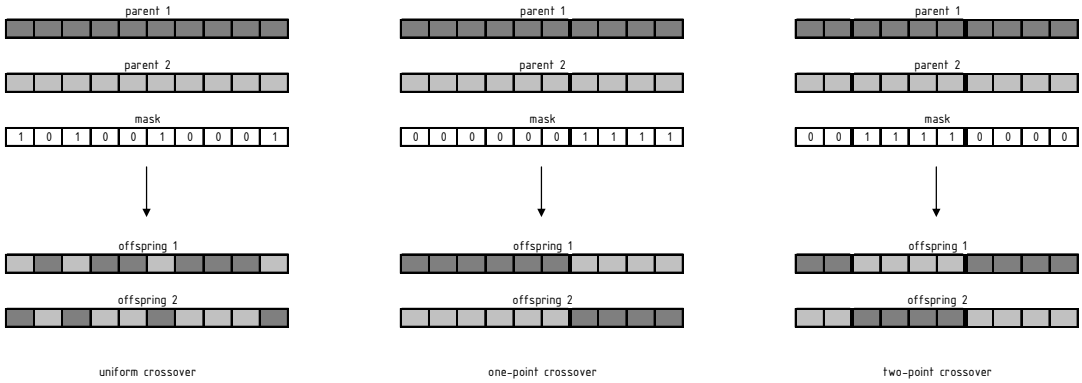


Fig. 4.6. Three crossover operator types

Although the one-point crossover method was inspired by biological processes, it has one major drawback in that certain combinations of schema cannot be combined in some situations. Multi-point crossover can be introduced to overcome this problem. As a result, the performance of generating offspring is greatly improved. Another approach is the uniform crossover. This generates offspring from the parents, based on a randomly generated crossover mask. The resultant offspring contain a mixture of genes from each parent. The number of effective crossing points is not fixed, but will be averaged at *l/2*, where *l* is the length of the chromosome[70]. The preference for using which crossover technique is still arguable. A general comment is that each of these crossover techniques is particularly useful for some classes of problems and quite poor for others.

[70] Man [25]

mutation

Also mentioned before, the aim of mutation is to introduce new genetic material into an existing individual; that is, to add diversity to the genetic characteristics of the population. Mutation is used in support of crossover to make sure that the full range of allele values us accessible in the search. Mutation also occurs at a certain probability p_m, referred to as the *mutation rate*. Usually, a small value for p_m ϵ [0,1] is used (so close to 0) to ensure that good solutions are not distorted too much. However, research has shown that an initial large mutation rate that decreases exponentially as a function of the number of generations improves convergence speed and accuracy. The initial large p_m ensures that a large search space is covered, while the p_m becomes rapidly smaller when individuals start to converge to the optimum. The following mutation schemes have been developed [Fig. 4.7][71]:

- random mutate
  where bit positions are chosen randomly and the corresponding bit values negated;
- inorder mutate
  where two bit positions are randomly selected and only bits on and between these positions are mutated

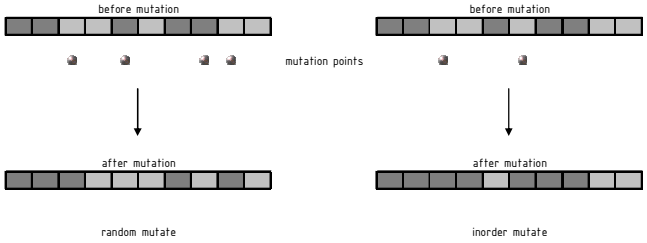


Fig. 4.7. Two mutation operator types

Mutation is what prevents the loss of diversity at a given bit position. For example, without mutation, every string in the population with a bit string encoding might come to have a one at the first bit position, and there would then be no way to obtain a string beginning with a zero. Mutation provides an 'insurance policy' against such fixation[72].

[71] Man [25]
[72] Mitchell [26]

crossover and mutation algorithm

Holland's analysis suggests that selection increasingly focuses the search on subsets of the search space with estimated above-average fitness, whereas crossover puts high-fitness 'building blocks' together on the same string in order to create strings of increasingly higher fitness. The crossover and mutation could be performed as follows[73]:

for i = 1, N_p
  Generate a random number z uniformly distributed in [0,1]
  If z > 0,1, perform crossover as follows:
    Randomly select two mating parents from the mating pool, and randomly choose two sites on the genetic strings and swap strings
  If $z \leq$ 0,1, skip crossover
  for j = 1, N_p
    Generate a random number z uniformly distributed in [0,1]
    If z > 0,99, perform mutation as follows:
      Randomly choose a member from the mating pool and switch a value to another possible value
    If $z \leq$ 0,99, skip to next j
  end
end

The choice of the crossover and mutation rate as the control parameters can be a complex nonlinear optimisation problem to solve itself. The increase of crossover probability would cause the recombination of building blocks to rise, and at the same time, it also increases the disruption of good chromosomes. On the other hand, should the mutation probability increase, this would transform the genetic search into a random search, but would help to reintroduce the lost genetic material. As each operator probability may vary through the generations, linear variations in crossover and mutation probability are suggested. Furthermore, their settings are critically dependent upon the nature of the objective function. This selection issue still remains open to suggestion although some guidelines have been introduced. For large populations ($N_p \geq 60$), the crossover rate can be set at 0,6 and the mutation rate can be set at 0,001. For small populations ($N_p < 60$) the crossover and mutation rate can be set at respectively 0,9 and 0,01[74].

[73] Burns [6]
[74] Man [25]

reordering

The order of genes on a chromosome is critical. The purpose of reordering is to attempt to find the gene order which has the better evolutionary potential. The order of genes between two randomly chosen positions is inverted within the chromosome, a technique which is known as *inversion* [Figure 4.8][75].

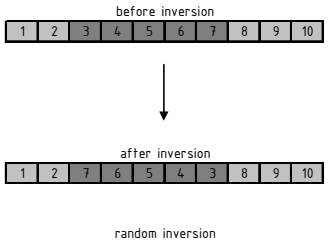


Fig. 4.8. The inversion operator

For the problem described in book one, the reordering operator is not adopted.

4.3.3 other aspects of genetic algorithms

island genetic algorithms[76]

In genetic algorithms, *islands* can be used for the representation of more populations. Selection, crossover and mutation occur in each subpopulation independently from the other subpopulation. In addition, individuals are allowed to migrate to another island, or subpopulation. In this way genetic material is shared among the subpopulations. Subpopulations can be initialised to cover different parts of the search space, thereby covering a larger search space and facilitating a kind of niching by individual islands. Also, in multi-criteria optimisation, each subpopulation can be allocated the task to optimise one criterion. A meta-level step is then required to combine the solution from each island.

Another aspect is, when can individuals migrate, from where, and to where? The simplest approach is to let migration occur at random. Individuals are selected randomly, as is the destination subpopulation. Alternatively, tournament selection can be used to select migrated individuals, as well as the destination. Intuitively, the best individuals of a poor island may want to migrate to a better island. However, individuals from a poor island may introduce bad genetic material onto a good island. Acceptance of an immigrant can then be based on a probability as a function of the immigrant's fitness value compared to that of the intended destination island, or acceptance if the immigrant has the ability to increase the

[75] Man [25]
[76] Engelbrecht [12]

diversity in genetic material, that is, if the individual is maximally different. It is also possible that a highly fit individual from a prosperous island visits islands, taking part in reproduction.

replacement scheme[77]

After generating the offspring, several representative strategies that can be proposed for old generation replacement exist. In the case of generational replacement, the chromosomes in the current population are completely replaced by the offspring. Therefore, the population with size *N* will generate *N* offspring.

Another modification for generational replacement is that not all of the chromosomes of the subpopulation are used for the next generation. Only a portion of the chromosomes (usually the fittest ones) are used to replace the chromosomes in the population.
Knowing that a larger number of offspring implies heavier computation in each generation cycle, the other scheme is to generate a small number of offspring. Usually, the worst chromosomes are replaced when new chromosomes are inserted into the population. A direct replacement of the parents by the corresponding offspring may also be adopted. Another way is to replace the eldest chromosomes, which stay in the population for a long time. However, this may cause the same problem as discarding the best chromosome.

robustness[78]

There are many instances where it is necessary to make the characteristics of the system variables adaptive to dynamic signal behaviour, and ensure that they are capable of sustaining the environmental disturbance. These often require an adaptive algorithm to optimise time-dependent optima which might be difficult to obtain by a conventional GA. When a simple GA is being used, the diversity of the population is quickly eliminated as it seeks out a global optimum. Should the environment change, it is often unable to redirect its search to a different part of the space due to the bias of the chromosomes. To improve the convergence of the standard GA for changing environments, two basic strategies have been developed.

The first strategy expands the memory of the GA in order to build up a repertoire of ready responses to environmental conditions.

The random immigrant mechanism and the triggered hypermutation mechanism are grouped as the second strategy. This approach increases diversity on the population to compensate for the changes encountered in the environment. The random immigrant mechanism is used to replace a fraction of a conventional GAs

[77] Man [25]
[78] Man [25]

population, as determined by the replacement rate, with randomly generated chromosomes. It works well in environments where there are occasional, large changes in the location of the optimum.

multi-objective and multi-modal[79]

The GA always has the distinct advantage of being able to solve multi-objective problems that other gradient type of optimisers have failed to meet. Indeed, engineering problems often exist in the class of multiple objectives.

Another attribute of the GA is its capability for solving multi-modal problems. However, there is no guarantee that the exact global optimum will be obtained by using GA (which hold true for any optimisation technique), although there is the tendency for this to occur.

constraints[80]

In the process of optimisation, the problem of constraints is often encountered. This obstacle is not always handled properly by the conventional, but mathematically governed optimisation techniques. By contrast, constraints present no problems to the GA and various methods can be used in this area.

- searching domain
  the search space of a chromosome can be confined to embed the condition of constraints in the system. This approach guarantees that all chromosomes are valid and that the constraints will not be violated. This method of solving the constraint problem requires no additional computing power, and all chromosomes created are regarded as potential solutions to the problem;
- repair mechanism
  the repair mechanism is a direct analogy to the DNA repair system. If any condition of the constraint is violated by a chromosome, the chromosome will be 'corrected' so that it becomes valid. This can be achieved by modifying some genes randomly within the valid solution space, or backtracking toward its parents' genetic material;
- penalty scheme
  invalid chromosomes can be given a penalty, such that they become less fit. The constrained problem is then transformed to an unconstrained condition by associating the penalty with all the constraint violations.

[79] Man [25]
[80] Man [25]

4.4 concluding remarks

Based on the biological evolutionary process, the genetic algorithm method is a technique to find an optimal solution in a large search space. Since the work of John Holland, numerous variations of the conventional GA have been proposed.
A standard simple GA is, actually, nothing more than a repeated process of a few number of simple operations. Genetic algorithms start with a random initial population of possible solutions. Every solution is an individual member of a large population of a generally fixed size. These individuals, also referred to as chromosomes, store a number of genes which are associated with a specific characteristic of the solution. In the traditional GA, the genes are usually bit-values (eg. 0 or 1), but they can actually contain strings, integers, real values, or Boolean operators as well. After generating the initial generation, a GA calls a sequence of operations, guided by the relative fitness value of the individuals, and drives the population to evolvement over a number of generations. The goal is to continually improve the fitness of the best (fittest) individual, until some or more termination criteria is/are satisfied.
Although, this process of evolution is relatively robust, the convergence of a GA is rather slow. In essence, a GA will waste time by testing the fitness of sub-optimal solutions as well. Also, the user must realise that genetic algorithms will only by chance find an optimal solution, whereas some traditional methods will find it exactly. But with the utilisation of the GA in the preliminary design stage, it is not completely necessary to find the optimal solution, as adaptations to the design are very likely. The question, however, remains; how fit, is fit enough?

5 structural optimisation projects

After the introduction of some computational optimisation techniques, the question may arise; in what way these techniques can be implemented in building engineering of building design. As an answer to this, several structures will be shown which are already built with the philosophy of structural optimisation. Four of them, with each different techniques are described in this chapter; The Groningen Twister project – Groningen [The Netherlands], The Web of North-Holland – Haarlemmermeer [The Netherlands], The Akutagawa Project – Takatsuki [Japan], and The Folded Roof Project – Frankfurt [Germany].

5.1 Groningen Twister – Groningen, The Netherlands[81, 82]

The Groningen Twister is a collaborative project between the design team of Kees Christiaanse Architects & Planners (KCAP) in Rotterdam, an engineering team of Arup in Amsterdam and the chair for Computer Aided Architectural Design (CAAD) at the ETH Zurich. The project was initiated in February 2003. The aim of the project was to develop a CAD-tool which would help the architects of KCAP to solve a complex design task. Underneath a pedestrian area that links the main station to the city centre of Groningen, there was a need for parking space for approximately 3000 bicycles [Figure 5.1].

Fig. 5.1. Model view of the Groningen Stadsbalkon [31]

[81] Scheurer [31]
[82] ETH [37]

To support the concrete slab of the pedestrian level, the desired design called for more than one hundred columns of different sizes to be placed in a random pattern, but to be then sized and controlled according to structural, functional, and aesthetic needs. To solve this problem, software was developed at the chair for CAAD that simulates a growth process for the columns. The distribution of the columns is defined by structural rules, provided by ARUP's engineers, as well as functional and design rules provided by KCAP's designers. The results are presented to the user as a three dimensional, dynamically evolving model. At any time during this process the user is able to control the model on the screen interactively. The user can control the process in two distinct ways, on the one hand by directly controlling the placement of single columns, on the other hand by adjusting various parameters that define the properties of the columns and the environment. The system provides real time feedback, as the column distribution tries to adapt to the changed configuration. This allows the user to test various alternative solutions in very short time [Figure 5.2].

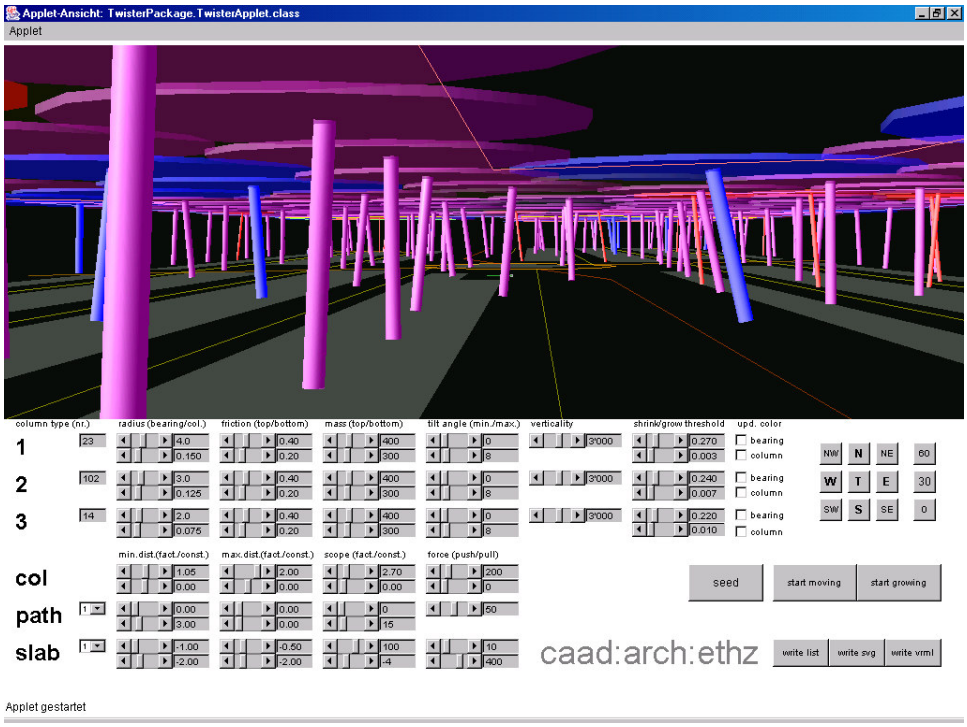


Fig. 5.2. Screenshot of the Groningen Twister model [31]

After a stable and satisfactory condition is achieved, the resulting column locations can be exported for construction documents in various digital file formats [Figure 5.3].

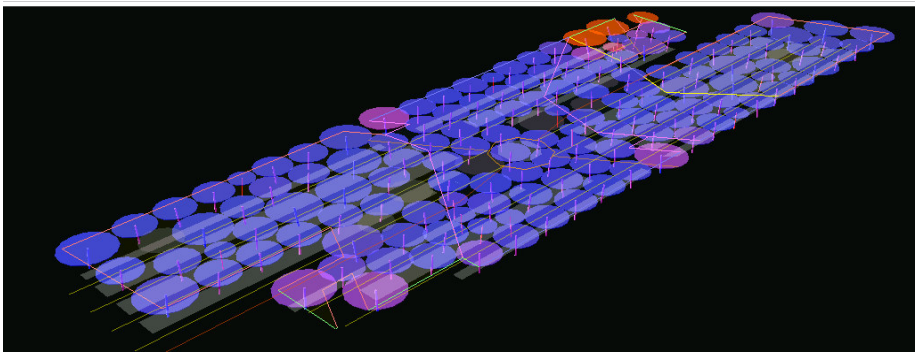


Fig. 5.3. Colour coding by kinetic energy [31]

The columns represent particles in a swarm system. Each column in the system is an autonomous individual, exploring the habitat and reacting to its neighbouring columns. According to the two layers of the habitat, the column model consists of two independent parts. The bottom end can move freely within the ground plane of the model, whereas the top end can move in the plane described by the slab. The actual column position, length and tilt are defined by the connecting line. It has to be assured, that the tilt angle stays below the assigned maximum [Figure 5.4].

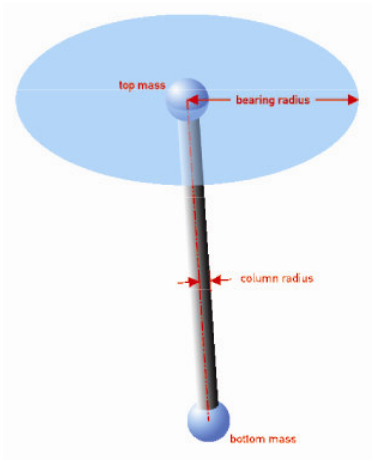


Fig. 5.4 The column model with a maximum bearing capacity and tilt [31]

This behaviour is easily described by a spring-mass-system [for more on particle-spring systems, see Appendix C]: punctual masses are connected by a virtual spring that pulls depending on the distances between the masses. In the model each organism is composed of two masses which describe the top and bottom end of the column and a spring in between. The force of this spring is proportional to the horizontal distance and, since the move of the masses is confined within the two planes of the habitat, they are drawn to positions above each other.

The columns are interacting with their adjacent columns as well as with the surrounding habitat following the same simple principles of attraction and repulsion by virtual springs. If they come to close, the top masses of each column are repelled by the slab outline, the holes, and the areas without cellar. The bottom masses are repelled by the paths.

To get the desired effect of distributing the columns, they seek to stay at a certain "social distance" to each other. This distance is defined by the maximum spanning distance of the slab and the bearing capacities of the respective columns. The bearing capacity of a column defines a circle around the top end marking the area where column is able to support the slab. Neighbouring columns therefore have to be aligned so that their radii touch or overlap slightly. This is also accomplished by virtual springs that push or pull between their respective top masses.

The specifications of the columns were given by Arup. There are three types (thicknesses) of columns with different diameters and bearing capacities. The bearing capacity results from the maximum radius of the column and defines the distance between the columns. The tilt angle of the columns was limited to 10 degrees so that this factor could be ignored in structural calculations. Also the height differences between the ground plane and the slab were not cared for and an average height of 3,0 meters was used throughout the habitat. The approximate number of columns needed was estimated by Arup based on the maximum radii and the building budget which would only allow for a certain number of columns.

By making the columns pressure sensitive for the lateral pressure from their neighbours and able to change their radius, an actual growth process was possible. Instead of assigning a column diameter and bearing radius from the beginning, the columns are able to adapt to their surroundings by changing their size autonomously.

6,0 Chapter 5 – structural optimisation projects

A column that is too far away from its neighbours detects a low surrounding pressure and starts to grow in diameter in discrete steps, matching the column types. If it reaches the largest possible state and still has no close neighbours, it splits into two small columns, which both start growing again. If a column gets to close with its neighbours or the edges of the habitat, the resulting pushing increases the pressure and it starts shrinking in just the same way. And if it reaches the smallest state while the pressure remains high, it finally dies. Thus, by "seeding" a single column the whole area of the slab is filling up with columns over time [Figure 5.5].

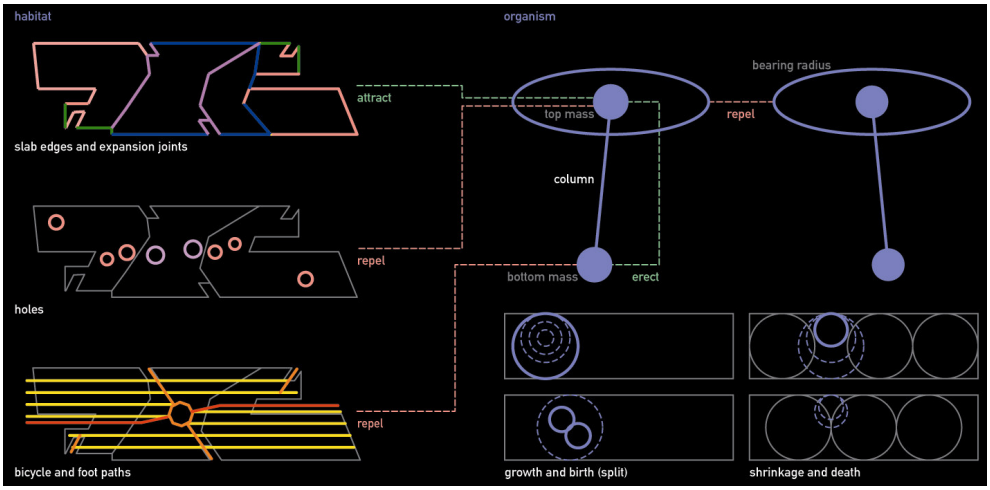


Fig. 5.5. The relation between the habitats and the 'agents' en the growth, birth, shrinkage and death of the columns [31]

5.2 Web of North Holland – Haarlemmermeer, The Netherlands[83]

Although the Web of North-Holland, is not an example of a structure based on computational structural optimisation, it is presented in this section, as computational aspects play a key role, and an optimal link between design and milling is aimed for. In other words, the Web was designed with the construction costs as one of the main optimisation objectives.

For the Dutch province of North Holland ONL designed a pavilion for the world horticultural exhibition 'Floriade' 2002. Architecturally there is no distinguishable difference between wall, floor or ceiling [Figure 5.6].

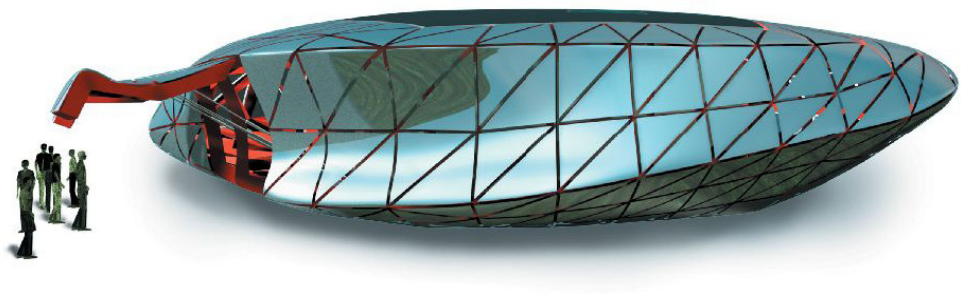


Fig. 5.6. A rendering of the Web of North Holland [5]

The design was based on a topological surface that governs the logical aesthetic continuity of the shape. The specific shape of the surface came about in a design process which combined milled physical models of the computer model with again computer modelling of adaptations to the milled models to attain a good space for its program as well as introducing the rigorous styling requirements of ONL. During this process a clear vision arose of the concave / convex dynamics and the shaping lines, the folding lines that fade in and fade out of the shape. ONL described the styling requirements in a number of shaping rules of the design. It was important to describe the design not in mass, but in a number of design rules and guidelines since its internal program was still to change. To control the shape and the look of the design a NURBS surface was created [Figure 5.7, on the next page].

[83] Boer [5]

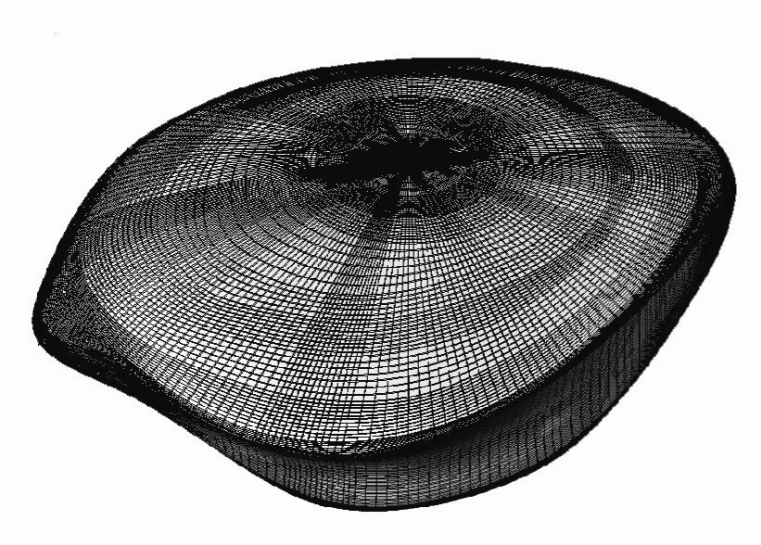


Fig. 5.7. The NURBS surface of the design [5]

NURBS is an acronym for Non-Uniform Rational Bezier Splines, a container for a number of polynomial algorithms. Its use is widespread in the design and character animation industry. In architecture the use of these techniques involves a genuine paradigm shift away from the use of two dimensional plans and sections. Simply put, one cannot build a double curved surface using plans and sections, because every plan and every section is different at different section planes. The logical reaction is to use the NURBS surface as the plan by having it govern the integrity of the construction. Expanding on the conventional paradigm of a construction grid ONL mapped a triangular grid with the internal integrity of an icosahedron (a 20-faced polyhedron on the NURBS surface.

The icosahedron system was chosen for a number of reasons, the main reason being that it is a closed system, like the design [Figure 5.8].

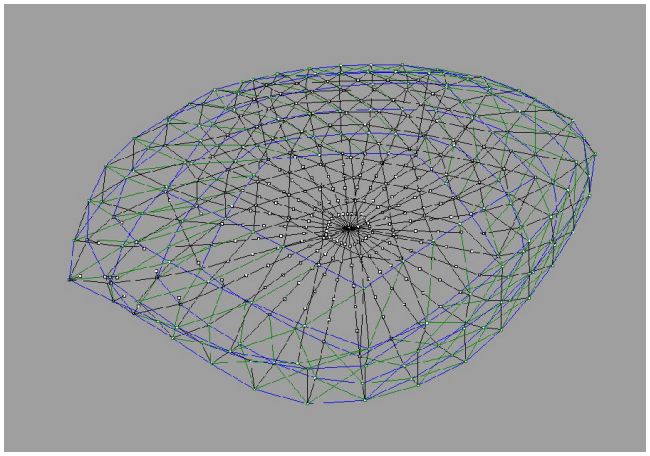


Fig. 5.8. Mapping of a constructive grid based on a icosahedron [5]

With the pavilion for the Web of North Holland ONL reaffirmed their strong beliefs acquired by previous projects [Elhorst-Vloedbelt, saltwater pavilion] that one can gain a maximum design freedom and keep the budget in check by gaining control over a system of similar, but different elements. A number of techniques can be determined that make this possible:

1. File to Factory: A construction process is greatly simplified by connecting the file created by the architect to the machine, eliminating intermediate steps that are inefficient - and even more so – susceptible to errors.

2. Mass customization: An irregular shape can only exist by the grace of irregular elements, therefore control over mass customization greatly increases design freedom.

3. Parameterisation: One Building, One Detail. Ideally, in a mass customized solution more parameters can be found than those that account for shape alone. These can be utilized to optimise the design. ONL mentioned earlier that an iterating construction calculation program can converge towards a construction that does not only have variable thicknesses, but also variable heights and an optimal point distribution. Similarly, in a design process parameters can change in accordance to design requirements and iterative scripts can be written to accommodate very specific demands. This is however, very complex (or perhaps impossible) in practice.

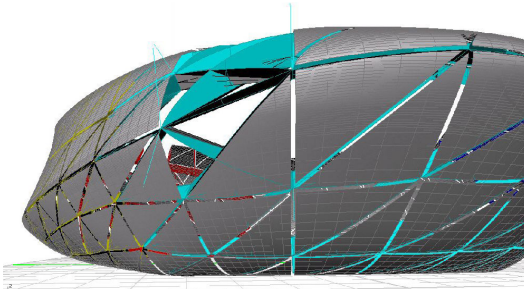


Fig. 5.9. 3D model of the panels with the construction showing [5]

4. Design control hierarchy: In this specific pavilion the shape is described in a single NURBS surface, essentially all that follows will refer to this surface. A NURBS surface is created using NURBS lines, keeping this creation link intact yields control on a higher level, by changing the line, the surface changes and the entire system changes. Primarily for designers this notion is paramount.

5. Body Styling: These techniques give the architect / designer full freedom to shape the volume of the building, to propose styled creases and smooth transitions of creases disappearing into the surface of the overall body. In the meantime ONL now has two projects in the production phase that have been designed with the above in mind: the Cockpit building and the Acoustic Barrier. The Cockpit building is part of a fluid design of the Acoustic Barrier, to accommodate the transition from the one to the other the design control hierarchy proved to be essential, both projects share the same outlines, but differ in construction principle. Construction is based on a streamlined File-to-Factory process described earlier.

Soon, the transferred Web will be delivered in front of the faculty of Architecture, Delft University of Technology.

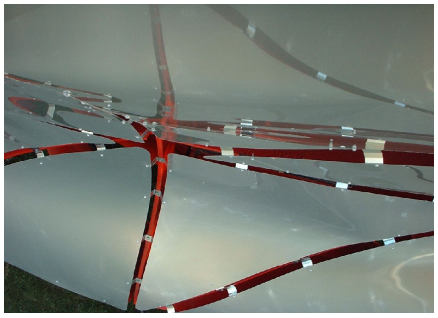


Fig. 5.10. Specific view to illustrate the effectiveness of the application of the Hylite [5]

5.3 Akutagawa River Side Project – Takatsuki City, Japan[84]

The Akutagawa River Side project is the project which brought a practical project to appeal the potential and possibility of the computational morphogenesis method for the future. The present project has been designed not only as to be effective as a leading project to the campaign, but also to have an attractive appearance, even if it does not have a big mass.

The west and south side wall development diagram of the building after completion (April of 2004) is shown in Figure 5.11 together with a rendering of the complete building. As can be seen from the diagram and the rendering, the west side and south side wall structures have a non-geometrical form, which has been generated through the proposed process of computational morphogenesis through usage of the extended ESO method.

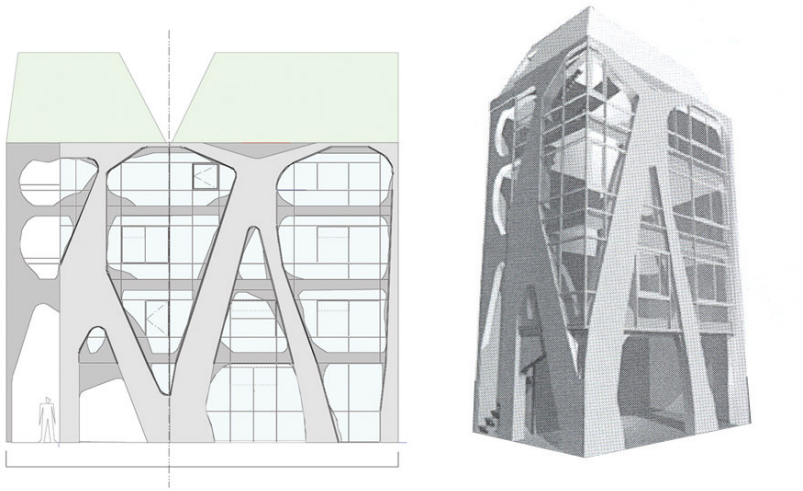


Fig. 5.11. The west and south side wall development diagram of the building and a perspective view of the building in a rendering [27]

[84] Ohmori [27]

Figure 5.12 shows the evolutionary process of the extended ESO method from which it can be observed how the south side wall form has been changed through the process of deletion of the portion with low density of Von Mises' relative stress, as well as the process of addition of the necessary portion. In the evolutionary process, it has to be noticed that the slabs of each floor level are treated not to be deleted by the evolutionary process.

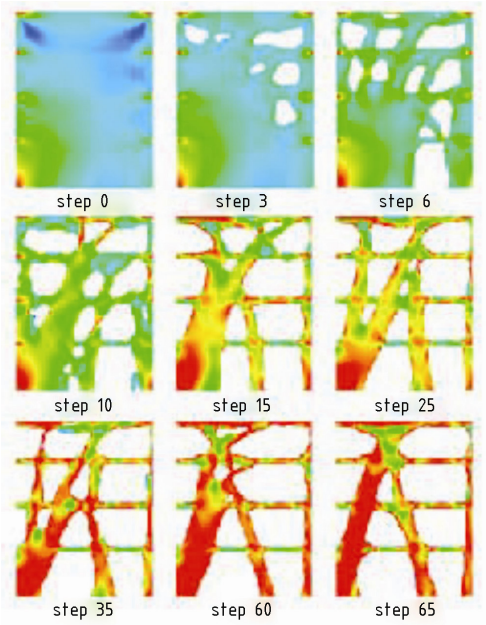


Fig. 5.12. The evolution process of the south wall [27]

In order to ensure and proof that the structure with the form obtained through the computational morphogenesis procedure has enough capability, 3D elasto-plastic numerical analysis is carried out. Figure 5.13, on the next page respectively shows the deflection state of the whole structure subjected to the horizontal loads in x-, y-direction and also in the direction at an angle 45 degrees to the x-axis. In Figure 5.14, on the next page a photograph is given with the building under construction.

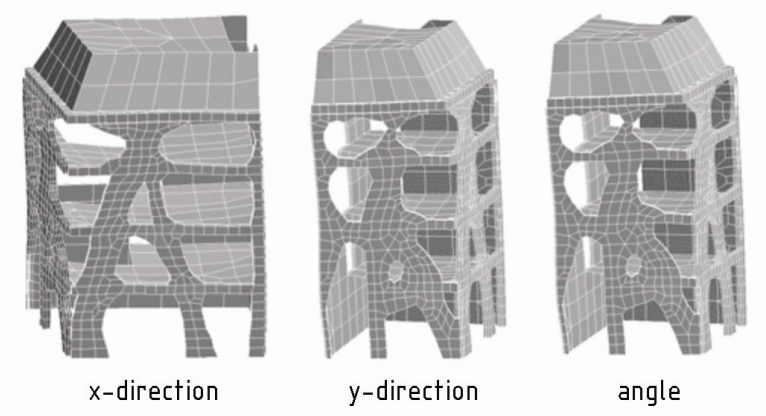


Fig. 5.13. The limit state deflection of the whole structure subjected to horizontal loads in x-, y-direction and under an angle [27]

Fig. 5.14. Inside views and outside views of the completed building [27]

5.4 Folded Roof Project – Frankfurt, Germany[85, 86]

The Folded Roof Project, a large roof structure assembled from steel members and nodes standing on a central foot, was developed by Fabian Scheurer of the chair for CAAD in collaboration with Bollinger+Grohmann Engineering in Frankfurt. By combining structural analysis software with a genetic algorithm, it was possible to evolve possible solutions for a highly sophisticated problem. In terms of topology the structure is a single plane, triangulated in a regular pattern of 289 nodes and 800 members, three-dimensionality and stability are achieved by folding. The nodes are arbitrarily translated along the z-axis within certain boundaries. At the foot, the plane folds down to the ground. This posed a tricky problem to the engineers. To achieve the necessary cantilevering capacity, the structure needed to have deep folds running from the centre to the rim of the roof [Figure 5.15]. But since there are no members spanning across the hole in the middle, a tension-bearing ring around the hole is the only way of keeping it from being drawn apart. Obviously, those are two contradicting concepts.

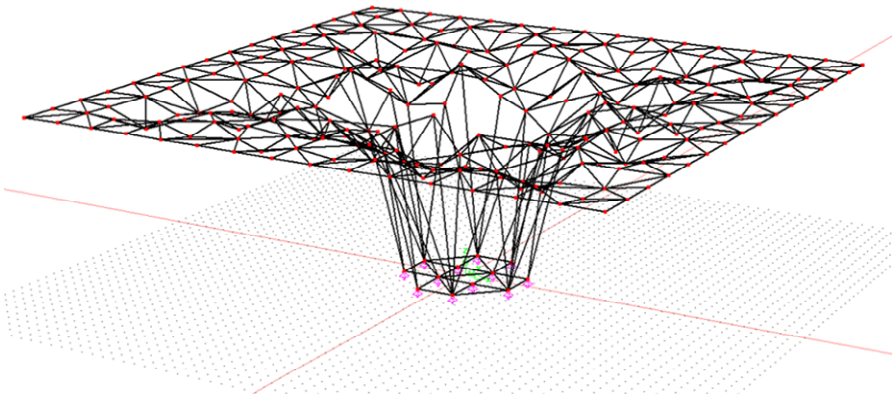


Fig. 5.15. The optimised folded roof structure [32]

Since it was impossible to find a set of geometric rules for creating stability, the only way to determine the structural performance was to simulate the behaviour of the whole structure with analysis software and to optimise it based on these results – which took about 4 seconds per run. By encoding the z-coordinates of all nodes into a genome and using a genetic algorithm, which allowed for crossover and mutation, the performance of the structure could be significantly improved. As a fitness criteria, the deflection of the nodes under self-weight was calculated by the analysis software, the worst node defining the inverse fitness fore each individual instance.

[85] Vrachliotis [33]
[86] Scheurer [32]

After 200 generations with 40 individuals each, the deflection repeatedly reached a minimum of 129 mm – at a cantilever of 25 meters [Figure 5.16].

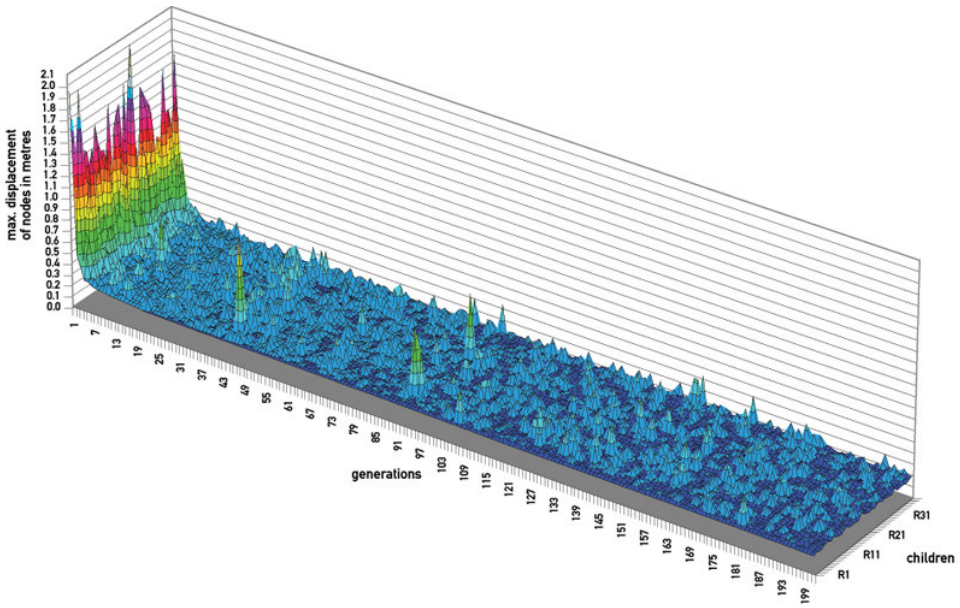


Fig. 5.16. Node displacement values of 200 generations with 40 individuals [32]

In this case, the results generated have not (yet) used to build the structure, but were used as a 'proof of concept' and to develop an understanding for the mechanisms which could be used to create an optimal design. Interestingly, not one of the engineers on the project, with an impressive amount of experience between them, would have come up with the same engineering concepts that evolved from the A-life algorithm.

5.5 concluding remarks

As shown in the four building projects, presented in Chapter 5, optimisation methods can be and are implemented in the design process. And as was shown, the optimisation objectives can be different for every project, be it a stress-related objective (The Groningen Twister and The Akutagawa River Side project), a construction and cost-related objective (The Web of North-Holland), or a serviceability – in the sense of the displacements – related objective (The Folded Roof project). It can be expected that the number of building projects, based on or driven by AI optimisation methods will increase over the upcoming years.

6,12 Chapter 5 – structural optimisation projects

6 concluding remarks and recommendations of the Master's thesis

This chapter concludes book two of the Master's thesis 'optimisation of structural transfer zones in multi-use buildings'. Here conclusions are given in two distinct sub sections. The first deals with conclusions on structural optimisation, the second with conclusion on genetic algorithms. Finally, recommendations are posted for further research on the subject dealt with in book two in the field of building engineering.

conclusions on structural optimisation

optimisation techniques

- As non-standard and free form (or better difficult-form) architecture is gaining more interest, optimisation of structures or structural elements will play an important role in the design process. The building engineer can adopt both manual and computational optimisation methods and techniques, in order to optimise the structure to be designed. The goal of utilising these methods and techniques is to explore unforeseen solutions to structural problems and to compare their overall design value with standard solutions.
- As the spread of different methods is very large, a number of these manual and computational optimisation techniques, some of which are based on the new artificial intelligence methods, are presented in the Master's thesis. And as the main result from this research, it can be concluded that many optimisation methods can be implemented in the conceptual and preliminary design stage, but that the choice for the technique to be used is largely dependant on the problem to be solved.

threats

- The biggest threat in using the computational techniques lies in the aspect that frequently, no or hardly any insight is given in the procedures and functions of the computational run. That is why, it must be avoided that a structural designer will adopt the results acquired from a computational optimisation tool, without using his engineering abilities to check the outcome.

- Another threat can be the formulation of the design objectives of the model. On one hand, the number of objectives may not be too large, as the optimisation process will then not converge fast enough or will not converge at all. On the other hand, many aspects play an important role in the design process and omitting some of these aspects results in an inadequate model.

expectations

- Various projects are already built based on or driven by optimisation techniques, and it can be expected that the number of building projects with optimisation aspects in the design process will increase exponentially in the upcoming years. This aspect mainly comes from the good results of these projects and the prediction that first, the structural designers will gain more interest and experience in the field of optimisation, and second that the collaboration with computer scientists will lead to more defiant designs.

conclusions on genetic algorithms

GA in general

- Genetic algorithms are known to be able to come to optimal results for a specific given problem, with metaphorically evolving living creatures in a well-defined environment. Key issue in adopting genetic algorithms is the fitness function with which these 'living creatures' or individuals are evaluated. In contrast to more some traditional numerical techniques, genetic algorithms operate on entire populations of possible solutions to find an optimal solution. This makes it much more likely to locate a global peak than the traditional techniques. So in conclusion, especially problems with numerous possible solutions that have to comply with a set of boundary conditions can be solved using genetic algorithms.

disadvantages

- Attempts to optimise 'everything' by filling the fitness function with to many parameters, have a higher probability of generation total chaos rather than emergent patterns[87].
- Another drawback is the extensive test and validation procedures that need to be undertaken to acquire the best GA parameters, such as the crossover and mutation rate [see the addendum of this Master's thesis].
- With the characteristic of a relative slow convergence speed and the fact that the optimal solution of a GA run is merely an estimate based on probability, the utilisation of GA's as an optimisation tool is not always justified with the advantages given above.

[87] Scheurer [32]

usability

- It can be stated that with the right type of genetic operators and the right values of the genetic parameters, new tools based on GA can be created that become full members of the design process of structural engineers. Of course, as long as the problem to be solved contains objective functions characterised by many local optima and a large search space. In this situation, the genetic algorithms offer a powerful alternative for the traditional methods.

recommendations

- When dealing with new (computational) techniques, ageing of knowledge on this subject can be somewhat problematic. Besides, as the fields of building engineering and computational engineering are relatively far apart from each other in terms of collaboration, it can be well recommended that both a structural engineer and a computer scientist, working closely together, must 'attack' the design process including the structural optimisation aspects together. In order to achieve this, both academics need to be educated to a small extent in the field of their colleague. In other words, it is recommended that structural and building engineering students can have the opportunity (and it is to them to grasp it) of education on computational (optimisation) techniques.

7,0 Chapter 6 – concluding remarks

references

books and reports

[1] Arora, J.S., *Guide to structural optimization*. New York: American Society of Civil Engineers, 1997

[2] Bendsøe, M.P., *Optimization of structural topology, shape, and material*. Berlin Heidelberg: Springer-Verlag, 1995

[3] Bendsøe, M.P., and O. Sigmund, *Topology optimization – Theory, methods and applications*. Berlin Heidelberg: Springer-Verlag, 2003

[4] Bonabeau, E., et al., *Swarm intelligence – from natural to artificial systems*. New York: Oxford University Press, 1999

[5] Boer, S., and K. Oosterhuis, *Architectural parametric design and mass customization*. Istanbul: eWork and eBusiness in Architecture, Engineering and Construction. Proceedings of the 5th European Conference on Product and Process Modelling in the AEC Industry, ed. A. Dikbas, 2004

[6] Burns, S.A., *Recent advances in optimal structural design*. Reston: American Society of Civil Engineers, 2002

[7] Coenders, J.L., *Master's thesis – Form finding and structural optimization, research and development of a new concept*. Delft: Delft University of Technology, Structural Design Lab, 2004

[8] Cook, R.D., et al., *Concepts and applications of finite element analysis*. Third edition. New York: John Wiley & Sons, Inc. 1989

[9] Cox, H.L., *The design of structures of least weight*. Oxford: Pergamon Press Ltd., 1965

[10] Dorigo, M., and Th. Stützle, *Ant colony optimization*. Massachusetts: Massachusetts Institute of Technology, 2004

[11] Elling, R., et al., *Rapportagetechniek – Schrijven voor lezers met weinig tijd*. Second edition. Groningen: Wolters-Noordhoff bv, 2000

[12] Engelbrecht, A.P., *Computational intelligence – an introduction*. West Sussex: John Wiley & Sons Ltd., 2002

[13] Escher, M.C., *M.C. Escher – Grafiek en tekeningen*. Köln: Taschen GmbH, 2001

[14] Haftka, R.T., and M.P. Kamat, *Elements of structural optimization*. Dordrecht: Martinus Nijhoff Publishers, 1985

[15] Haftka, R.T., et al., *Elements of structural optimization*. Second revised edition. Dordrecht: Kluwer Academic Publishers, 1990

[16] Harris, J., *An introduction to fuzzy logic applications*. Dordrecht: Kluwer Academic Publishers, 2000

[17] Hartsuijker, C., *Toegepaste mechanica, Deel 1 Evenwicht*. Schoonhoven: Academic Service, 1999

[18] Hemp, W.S., *Optimum structures*. Oxford: Clarendon Press, 1973

[19] Kilian, A., and J. Ochsendorf, *Particle-spring systems for structural form finding*. Journal of the Interational Association for Shell and Spatial Structures: IASS. 2005

[20] Kirsch, U., *Structural optimization – Fundamentals and applications*. Berlin Heidelberg: Springer-Verlag, 1993

[21] Kodiyalam, S., and M. Saxena, *Geometry and optimization techniques for structural design*. Southampton Boston: Computational Mechanics Publishers, 1994

[22] Koenen, M.J., and J.B. Drewes, *Wolters' woordenboek Nederlands*. Twenty-eighth edition. Groningen: Wolters-Noordhoff bv, 1990

[23] Liang, Q.Q., *Performance-based optimization of structures – Theory and applications*. Oxon: Spon Press, 2005

[24] Longman Dictionaries, *Dictionary of contemporary English*. Third edition. Essex: Longman Dictionaries, 1995

[25] Man, K.F., et al., *Genetic algorithms – Concepts and designs*. London: Springer-Verlag, 1999

[26] Mitchell, M., *An introduction to genetic algorithms*. Cambridge: The MIT Press, 1996

[27] Ohmori, H., et al., *Computational morphogenesis and its application to structural design*. Romania: Proceedings of the International symposium on shell and spatial structures, ed. M. Mihailescu, 2005

[28] Owen, J.B.B., *The analysis and design of light structures*. London: Edward Arnold (Publlishers) Ltd., 1965

7,2 references

[29] Pfeifer, R., and C. Scheier., *Understanding intelligence*. Cambridge: The MIT Press, 1999

[30] Sarker, R., et al., *Evolutionary optimisation*. Dordrecht: Kluwer Academic Publishers, 2002

[31] Scheurer. F., *The Groningen Twister – an experiment in applied generative design*. Milan: In Generative Art 2003. Proceedings of the 4th International Conference, ed. C. Soddu, 2003

[32] Scheurer. F., *Getting complexity organized using self-organisation in architectural construction*. Paper for Automation in construction – Elsevier, 2005

[33] Vrachliotis, G., and F. Scheurer, *Theory and applications of biological inspired computer-aided architectural design*. Delft: Game Set and Match II Conference, ed. X. Xia, 2006

[34] Xie, Y.M., and G.P. Steven, *Evolutionary structural optimization*. London: Springer-Verlag, 1997

internet

[35] American Scientist Online, *Computing science – how to count*. January 2006. http://www.americanscientist.org/

[36] Bourke, P., *Particle System Example*. December 2005. http://astronomy.swin.edu.au/~pbourke/modelling/particle/

[37] Eidgenössische Technische Hochschule Zürich, *The Groningen Twister – An experiment in applied generative design*. November 2005. http://wiki.arch.ethz.ch/twiki/bin/view/Extern/GroningenTwister

[38] Lexico Publishing Group, LCC, *Dictionary.com*. August 2005. http://www.dictionary.com

[39] TOPOPT research group, *Topology optimization*. September 2005. http://www.topopt.dtu.dk/

[40] Universität Stuttgart, Institute for Structural Mechanics, *Adaptive Finite Element Methods for fast transient, highly nonlinear processes*. August 2005. http://www.uni-stuttgart.de/ibs/research/adaptive/index.html

7,4 references

Appendix A. introduction in Finite Element Analysis[88, 89, 90]

Structural optimisation has achieved far less popularity in practice compared to finite element analysis, despite the extraordinary progress of the optimisation theory and associated algorithms over the last three decades. This is caused by the difficulties and complexities of existing structural optimisation methods. The finite element method (FEM) is a numerical technique for analysing structures and continua and has been widely used by design engineers and researchers in all engineering fields. In the finite element method, a structure or structural component is discretised into finite geometrical parts (finite elements). These elements defined by their coordinates describe the shape of the component, which may be very complex. The method generates many simultaneous algebraic equations that are solved on a digital computer. One of the great advantages of the FEM is its versatility. The FEM can be applied to various physical problems with arbitrary shape, loads and support conditions and the mix elements of different types, shapes and material properties can be used. Another advantage of the FEM is that the finite element model physically represents the actual structure. With advances in digital computers, the FEM had become a computational tool in the performance-based design of engineering structures.

The PBO method employs the FEM as a modelling and computational tool. From the results of the finite element analysis (FEA), the PBO programs identify the underutilised elements that are inefficient in carrying the loads. The underutilised elements are then removed from the structure to improve its performance. The process of the FEA, performance evaluation and element removal is repeated until the optimal structure is generated.

So, the finite element method is a numerical procedure for analysing structures and continua. Usually the problem addressed is too complicated to be solved satisfactorily by classical analytical methods. The finite element procedure produces many simultaneous algebraic equations, which are generated and solved on a digital computer. The results are rarely exact. However, errors are decreased by processing more equations, and results accurate enough for engineering purposes are obtainable at reasonable cost.
The finite element method originated as a method of stress analysis. Problems that previously were utterly unsolvable are now solved routinely.

[88] Cook [8]
[89] Liang [23]
[90] Xie [34]

7,6 appendices

Appendix B. travelling salesman problem[91,92]

A combinatorial optimisation problem is either a maximisation or a minimisation problem which has associated a set of problem instances. The term *problem* refers to the general question to be answered, usually having several parameters or variables with unspecified values. The term *instance* refers to a problem with specified values for all the parameters. Although some important combinatorial optimisation problems have been shown to be solvable in polynomial time, for the great majority of combinatorial problems no polynomial bound on the worst-case solution time could be found so for. For these problems the run time of the best algorithms known increases exponentially with the instance size and, consequently, so does the time required to find an optimal solution. A notorious example of such a problem is the Travelling Salesman Problem.

The travelling salesman problem is the problem faced by a salesman who, starting from his home town, wants to find a shortest possible trip through a given set of customer cities, visiting each city once before finally returning home. The TSP can be represented by a complete weighted graph $G = (N,A)$ with N being the set of $n = |N|$ nodes (cities), A being the set of arcs fully connecting the nodes. Each arc $(i,j) \in A$ is assigned a weight d_{ij} which represents the distance between cities i and j.

$$d_{ij} = \left[\left(x_i - x_j \right)^2 + \left(y_i - y_j \right)^2 \right]^{\frac{1}{2}}$$

The TSP is the problem of finding a minimum length Hamiltonian circuit of the graph, where a Hamiltonian circuit is a closed walk (a tour) visiting each node of G exactly once. Thus, an optimal solution to the TSP is a permutation π of the node indices $\{1,2,\ldots,n\}$ such that the length $f(\pi)$ is minimal, where $f(\pi)$ is given by

$$f(\pi) = \sum_{i=1}^{n-1} d_{\pi(i)\pi(i+1)} + d_{\pi(n)\pi(1)}$$

[91] Bonabeau [4]
[92] Dorigo [10]

Tours are constructed by applying the following simple constructive procedure to each ant.

1. choose, according some criterion, a start city at which the ant is positioned;
2. use pheromone and heuristic values to probabilistically construct a tour by iteratively adding cities that the ant has not visited yet, until all cities have been visited;
3. go back to the initial city.

In ant optimisation the main mechanism at work in ACO algorithms that triggers the discovery of good tours is the positive feedback given through the pheromone update by the ants. The shorter the ant's tour, the higher the amount of pheromone the ant deposits on the arcs of its tour. This in turn leads to the fact that these arcs have a higher probability of being selected in the subsequent iterations of the algorithm.
With good parameter settings, the long-term effect of the pheromone trails is to progressively reduce the size of the explored search space so that the search concentrates on a small number of promising arcs. Yet, this behaviour may become undesirable, if the concentration is so strong that it results in an early stagnation of the search. Search stagnation is defined as the situation in which all the ants follow the same path and construct the same solution. In such an undesirable situation the system has ceased to explore new possibilities and no better tour is likely to be found anymore.

Although generality is a desirable property, it makes theoretical analysis much more complicated, if not impossible.

It is important to note that, when considering a stochastic optimisation algorithm, there are at least two possible types of convergence: *convergence in value* and *convergence in solution*. Informally, and making the hypothesis that in case of problems with more than one optimal solution we are interested in convergence toward any of them, when studying convergence in value we are interested in generating an optimal solution at least once. On the contrary, when studying convergence in solution we are interested in evaluating the probability that the algorithm reaches a state which keeps generating the same optimal solution. Note, however, that although in general convergence in solution is a stronger and more desirable result to prove than convergence in value, in optimisation we are interested in finding the optimal solution once (after is has been found the problem is solved and the algorithm can be stopped), so that convergence in value is all that we need.

7,8 appendices

Appendix C. particle-spring systems[93].

Particle-spring systems serve as an excellent approximation for hanging models. These systems are based on lumped masses, called particles, which are connected by linear elastic strings [Figure A1]. Each spring is assigned a constant axial stiffness, an initial length, and a damping coefficient. Springs generate a force when displaced from their rest length. External forces can be applied to the particles, as in the case of gravitational acceleration. To solve for the equilibrium geometry of particle-spring systems, there are many techniques with varying degrees of efficiency and stability

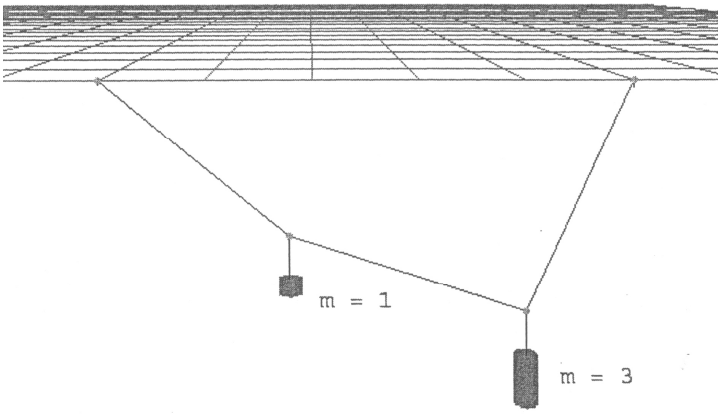


Fig. A1. Equilibrium of a simple particle spring system [19]

Each particle in the system has a position, a velocity, and a variable mass, as well as a summarised vector for all the forces acting on it. A force in the particle-spring systems can be applied to a particle based on the force vector's direction and magnitude. Alternatively the magnitude of the force can be calculated using a function as in the case of springs. Springs are mass-less connectors between two particles that exercise a force on the particles based on the spring's offset from its length. Particles can be restrained in any dimension, so it is straightforward to add supports by restraining the displacement vectors of an individual particle. The particle-spring system is usually not in equilibrium when the simulation is started and there will be movement throughout the system as the particles and springs seek their equilibrium positions. To prevent oscillations of the particles about their equilibrium positions, it is necessary to apply damping to the system, which can be applied as a coefficient to each spring. Another damping method is the subjection of viscosity to each particle in the surrounding environment.

[93] Kilian [19]

How the particles move given the forces acting on them is described as follows

$$a = \frac{d^2x}{dt^2} = \frac{f}{m}$$

where f is the force on a particle, m its mass, x its position, and a its acceleration. Note that f, x, and a are vectors in a 3D space (x,y,z).
The second order system can be written as two first order differential equations making them easier to solve.

$$v = \frac{dx}{dt} \quad \text{and} \quad a = \frac{dv}{dt}$$

There are two data structures required, one for the particles and one describing the springs. The structure for the particles includes their mass (normally constant) and the instantaneous position, velocity and force.

The structure for a spring includes the two particles it connects, and various spring attributes required for the force calculation, see later.

The key to determining the dynamics of the system is calculating the forces acting on a particle given the current state of the whole system. Three forces will be considered here, they are[94]

1. Gravitation
   This normally acts downwards and is added to the force vector of each particle individually.

   $$f = m \cdot g$$

   where g is normally (0,0,-9.8). Often particle systems don't include gravitation unless they are constrained in some way otherwise the particles keep "falling".

[94] Bourke [36]

2. Viscous Drag
   This acts to resist motion and like gravitation it acts on each particle independently of the others, depending only on the current velocity of the particle.

$$f = -k_d \cdot \frac{dx}{dt} = -k_d \cdot v$$

   $-k_d$ is called the drag coefficient, if zero then the particles are in a frictionless environment (vacuum), high values simulate drag in liquids.

3. Hooks Spring Law
   This determines the forces on two particles connected by a spring. We only need to work out the force on particle "a" due to particle "b", the force on "b" due to "a" is the negative of the first force. Each spring has a rest length, if the spring length is greater than this length then the force acts to pull the two particles together, if the spring length is less than the rest length then the force acts to repel the two particles. The spring damping force depends on the difference in the velocity of the two particles. Both of these forces act along the line defined by the two particles.
   The force on particle "a" due to particle "b" is given by:

$$f = -\left[k_s(\| x_a - x_b \| - r) + k_d(v_a - v_b) \frac{x_a - x_b}{\| x_a - x_b \|} \right] \frac{x_a - x_b}{\| x_a - x_b \|}$$

   Where k_s is known as the spring constant, k_d is the damping constant and r is the rest length.
   The particles might exert a gravitational attraction on each other. The gravitational attraction of one particle "a" due to another particle "b" is given by

$$f = \frac{G \cdot m_a \cdot m_b}{\| x_a - x_b \|^2} \cdot \frac{x_a - x_b}{\| x_a - x_b \|}$$

   and is in the direction along the line joining the two particles. G is referred to as the universal gravitational constant and equals $6.672 \times 10^{-11} N\ m^2\ kg^{-2}$

The particles could have a charge, this very similar in form to the gravitational law except that now the particles may repel if the charges are the same sign and attact if they are the opposite sign. The exact relationship is given by Coulombs law

$$f = \frac{k\,|q_a||q_b|}{\| x_a - x_b \|^2} \cdot \frac{x_a - x_b}{\| x_a - x_b \|}$$

where k is known as Coulombs constant equal to 8.9875 x 10^9 N m^2 C^{-2}. As for the gravitational force the electrostatic force acts along the line defined by the positions of the two particles.
Other forces can be added to this list dependent on requirements, it is only necessary to determine the appropriate physical forces involved.

One of the primary attractions of the particle-spring approach is that it allows the user to watch the system approach equilibrium and to intervene during the solution process [Figure A2]. The user can alter the applied loading, add or subtract structural elements, and change the support conditions in order to discover new structural forms while the simulation is running[95].

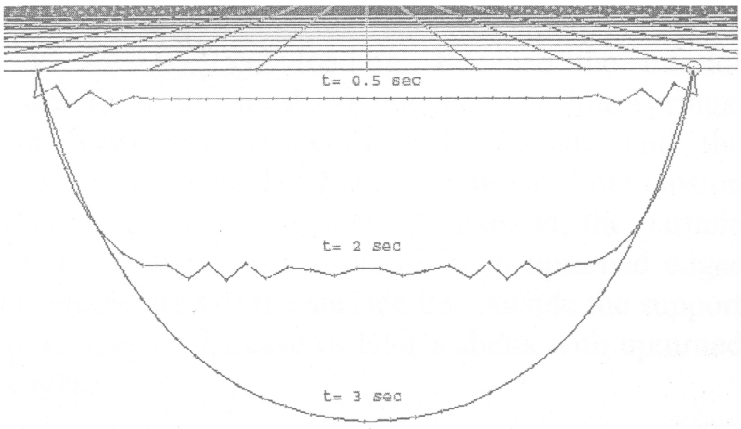


Fig. A2. Solution process for a cable with forty discrete masses at equal spacing [19]

[95] Kilian [4]

By nature of the solution procedure, the particle-spring system always finds a possible load path for which the forces are in equilibrium if it exists [Figure A3].

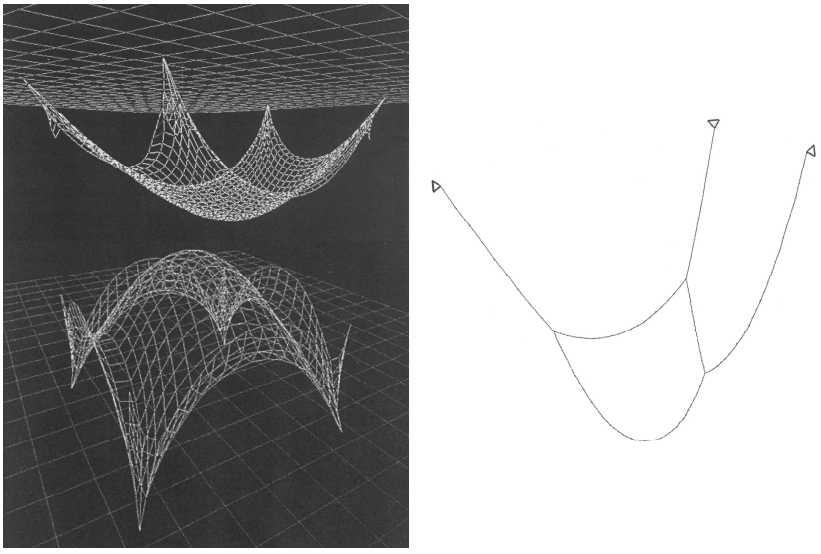


Fig. A3. A simple square mesh supported near the corners [left] and a statically determinate funicular form in 2D modelled with particle-spring simulations [19]

However, the particle-spring systems have to deal with some disadvantages as well. Due to the properties of axial springs they not only contract upon being stretched but also expand upon being compressed. This creates the possibility of tension and compression members being present in a hanging structure at the same time. Additionally, the choice of an appropriate mesh pattern can allow for statically indeterminate systems, such that each particle is attached to three springs or fewer for three dimensional structures. The solution procedures are somewhat expensive computationally, which limits the size of real-time simulation at this point. Current solutions are practical for solving up to 1,000 particles in real-time on a desktop computer. There is also a slight loss of precision due to the approximation of the simulation. The process is not static but the addition of string elements is dynamic so that the shape and form is constantly in flux. The final equilibrium shape emerges from the modelled topology. As in the case of a physical hanging model, any addition disturbs the balance of the initial shape. Finally, another disadvantage is that for very complex problems in combination some solutions can become unstable.

Although this method has a long way to go before it can becomes a standard approach in structural form-finding, the particle-spring system provides a powerful new method in structural designing.