# Single and Multi-Objective Optimization in Civil Engineering with Applications

by

Matěj Lepš

A treatise submitted in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy

Czech Technical University in Prague
Faculty of Civil Engineering

2004

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# ACKNOWLEDGMENTS

First of all, I would like to express my deepest gratitude to Prof. Ing. Zdeněk Bittnar, DrSc. for his support and patience not only during the whole course of the work on this thesis but also during my whole studies. I would like to give my thanks to Ing. Jan Zeman, Ph.D. for his inspiration throughout my research as well as for very precise proof-reading of my manuscripts.

I wish also to thank all members of our department who have given me valuable comments and advice. Particularly, I would like to thank Ing. Anička Kučerová and Ing. Ondřej Hrstka for close cooperation on many issues mentioned in this research work.

Most importantly, I would like to thank my parents, my girlfriend Jana, other family members and friends for their never ending encouragement and support that help me to attain the goal I have set for myself and the opportunity to fully concentrate on my study.

This work was supported by the research project MSM 210000003 and the European Union project Maecenas FIKS-CT-2001-00186.

Also the earlier support provided by the European Commission during my visit at Edinburgh Parallel Computing Center by the TRACS Research and Training programme, a grant number HPRI-CT-1999-00026, is gratefully acknowledged. At this point I would like to express my thanks to Prof. Barry Topping and his research team for hospitality and support during my stay at Heriot-Watt University in Edinburgh.

Last but not least, parts of this work were produced under the support of CTU grants CTU 0201611 and CTU 0300911 and FRVŠ grants G1 2074/2002 and G1 2078/2002.

Czech Technical University in Prague
Faculty of Civil Engineering

Abstract

# Single and Multi-Objective Optimization in Civil Engineering with Applications

by Matěj Lepš

The proposed thesis mainly deals with problems associated with the Civil Engineering area and therefore the methods as well as problems have been selected with emphasis on applications to Civil Engineering topics. And also as nowadays very popular optimization methods, Evolutionary Algorithms (**EA**s) will be presented as one of the possible ways how to solve today's challenging optimization problems that we are often facing.

First of all, two main optimization tasks are described - the **Global optimization** as a "mathematical" problem and the **Structural optimization** as an "engineering" one. Several categories among Structural optimization problems are described and some solutions from the Global optimization area are cited.

Next, a new classification for Evolutionary Algorithms is presented. It is based on the well-known notation developed for Evolution Strategies (**ES**s) [Bäck and Schwefel, 1995] and appropriately modified. The leading idea is that every **EA** can be described by a combination of three basic operations, namely **recombination**, usually presented by different cross-overs, **mutation** and **selection** mechanisms.

Traditionally, the **EA**s have been developed for single-objective problems (**SOP**s) and therefore they are not so suitable for problems coming from engineering practice where we usually deal with **multi-objective**, **constrained** and often **mixed integer-continuous** optimization problems (**CMOP**s). Solutions for all the three phenomena are presented: multi-objective nature can be solved by Pareto-optimality approaches, constraints by penalty functions and different types of variables by an appropriate encoding. Several other possibilities are discussed in the text as well.

Based on the above mentioned notation, four particular examples of **EA**s that have been developed in recent years at the workplace of the author are described and compared. In particular, three of them are based on the combination of Genetic Algorithms [Goldberg, 1989] with the Differential Evolution [Storn, 1996]; the last algorithm is the Differential Evolution alone. These optimization algorithms are then used to solve several tasks from engineering practice as well as two test functions and their advantages and disadvantages are shown.

The next part is devoted to the application of the presented optimization methods to the design of reinforced concrete frames. Generally, this task is multi-modal, multi-objective and highly constrained. To solve this problem as a whole, it is shown that this inevitably leads to an integer formulation of the problem and hence presented qualities of multi-objective Evolutionary Algorithms are utilized. As an illustrative result, typical examples are solved and the Pareto-fronts in terms of the total price of a structure against its deflection are depicted. A new system of visualization is also presented as an addition to the multi-objective optimization domain.

As an engineering example of a single-objective optimization problem, training of an artificial neural network and its use for a microplane material model parameters prediction is presented. A traditional method for neural network training used herein is the well-known Backpropagation method, which uses a gradient-based operator to minimize an output error. As a novel approach, the Evolutionary Algorithm can be used here for the same purpose. It is shown that obtained errors are much lower than the outputs obtained from the Backpropagation algorithm.

Next, an identification of the microplane material model [Bažant et al., 2000] is investigated. This model is a fully three-dimensional material law that includes tensional and compressive softening, damage of the material, different combinations of loading, unloading and cyclic loading along with the development of damage-induced anisotropy of the material. The rather severe disadvantage of a microplane model is an extreme demand of computational time and, therefore, an appropriate procedure is on demand. To define the problem more formally, the optimization goal is to find microplane parameters from a stress-strain diagram of a test specimen in a uniaxial compression. The objective function is then the least square error function, which contains differences between values of a known stress-strain curve and values from a microplane model simulation.

Several approaches are tested here to solve the introduced problem. An estimation by an artificial neural network trained on approximations of stress-strain curves shows that some properties can be predicted well but a significant error in other coefficient is obtained. Next, a parallel version of the evolutionary-algorithms-based global optimizer **SADE** is directly used to obtain required parameters by varying them within a nonlinear finite element analysis. The first main result is that this time consuming analysis can be solved by a parallel analysis in reasonable time. The second outcome is the fact that the objective function corresponding to the identification problem has several local minima, which are characterized by similar values but are far from each other. To solve the above mentioned obstacles and in the view of recent research in this domain, a new methodology is also presented: an application of a Latin Hypercube Sampling method as well as a sensitivity analysis are applied not only to investigate the influence of individual material model parameters, but also to minimize the need of training samples for an artificial neural network. Several promising results along with some concluding remarks are presented.

České vysoké učení technické v Praze
Fakulta stavební

Abstrakt

# Jedno a Vícekriteriální Optimalizace s Aplikacemi ve Stavebním Inženýrství

Matěj Lepš

Rozvoj výpočetní techniky v současné době umožňuje zcela nové přístupy k řešení mnoha teoretických i praktických problémů. Konkrétně v oblasti stavební mechaniky je modelování jednotlivých materiálů a díky tomu následně i predikce chování konstrukcí daleko přesnější než dříve. Snahou této práce je tedy ukázat nejnovější přístupy v oblasti globální optimalizace a jejich aplikace na vybrané úlohy stavebního inženýrství.

Nejprve je popsán rozdíl mezi *globální optimalizací*, kterou lze pokládat za typickou „matematickou” oblast, a *optimalizací konstrukcí*, která se obvykle zabývá úlohami „inženýrskými”. Pro lepší pochopení navrhovaných postupů jsou nejprve představeny různé formy optimalizace konstrukcí, jmenovitě *topologická optimalizace* (topology optimization), *tvarová optimalizace* (shape optimization), *rozměrová optimalizace* (size optimization), *topografická optimalizace* (topography optimization) a *optimalizace skladby*[1] (layout optimization). Pro každou z těchto forem optimalizace je vhodná jiná optimalizační metoda z oblasti globálních optimalizací. Je také nutné zdůraznit, že typická úloha bude nejspíše kombinací několika těchto forem.

Další část práce se zabývá klasifikací a popisem evolučních algoritmů. Jsou využity a dále obohaceny poznatky z oblasti evolučních strategií. Hlavní myšlenkou je skutečnost, že se valná většina evolučních algoritmů dá rozepsat jako konkrétní kombinace tří operátorů – rekombinace, nebo-li křížení, mutace a výběru.

Původně byly evoluční algoritmy navrženy pouze pro jednokriteriální funkce a z tohoto důvodu nejsou v této původní formě příliš vhodné pro řešení inženýrských úloh, kde se velice často potkáváme s vícekriteriální podmíněnou optimalizací (CMOP). Vícekriteriální optimalizace jsou zejména charakteristické tím, že jejich výsledkem není jedno optimum, ale množina dominantních řešení. Proto byly objeveny vícekriteriální evoluční algoritmy. Jejich výhodou je, že k hledání optima využívají množinu možných řešení a díky této vlastnosti jsou dobře přizpůsobitelné pro hledání většího množství dominantních řešení. Proto je značná část této práce věnována úpravě evolučních algoritmů pro vícekriteriální optimalizaci.

S poznatky již představené klasifikace evolučních algoritmů jsou představeny čtyři evoluční algoritmy, které byly vyvinuty na pracovišti autora v posledních letech. Tři z nich jsou založeny na kombinaci genetických algoritmů a diferenciální evoluce, poslední metodou je samotná diferenciální evoluce. Tato kolekce algoritmů je následně použita při řešení dvou inženýrských úloh a dvou optimalizačních problémů. Výhody a nevýhody těchto čtyř algoritmů jsou diskutovány v závěru této části práce.

Další část práce se zabývá vytvořením návrhového nástroje, který by jednoduše a spolehlivě navrhl a zoptimalizoval železobetonovou rámovou konstrukci. Obecně lze tuto úlohu označit jako multimodální, multikriteriální a podmíněnou optimalizaci. Aby bylo možné vyřešit návrh takovéto konstrukce jako jednu optimalizační úlohu, je nezbytně nutné ji formulovat v diskrétních proměnných a to následně nevyhnutelně vede k aplikaci multikriteriálních evoluč-

[1] Volně přeloženo.

ních algoritmů. Navrhovaný postup je dokumentován na několika typických úlohách. Dominantní řešení jsou pak vykreslena v prostoru celkových cen a příslušných deformací konstrukce. Pro tyto účely je též navrhnut nový systém grafického zobrazování dominantních řešení.

Příkladem inženýrské jednokriteriální úlohy může být trénování (učení) umělé neuronové sítě a její následné využití pro získání parametrů mikroploškového (microplane) modelu betonu. V práci je porovnáno učení neuronové sítě tradiční metodou zpětného šíření (Backpropagation) s jedním zástupcem evolučních algoritmů. Je ukázáno, že navrhovaný postup vykazuje na výstupu mnohem menší chyby než tradiční řešení.

Takto natrénovaná neuronová síť je následně použita k získání parametrů mikroploškového modelu betonu. Výhodou tohoto modelu je, že je schopen realisticky popsat různé druhy odezvy betonu. Velkou nevýhodou se naopak ukazuje jeho výpočetní náročnost. První řešení tedy využívá neuronové sítě naučené na aproximaci pracovních diagramů. Výsledky ukazují, že některé z parametrů se dají získat poměrně snadno, ale u zbylých byla chyba od požadovaných hodnot příliš velká. Proto byla použita přímo paralelní verze algoritmu SADE. První podstatný výsledek je ten, že se tato výpočetně náročná optimalizace dá díky paralelizaci vypočítat v reálném čase. Druhým výsledkem je skutečnost, že tato úloha je značně multimodální a tudíž vede na několik rozdílných optim. Jako nejnovější postup byla použita metoda Latin Hypercube Sampling a stochastická senzitivní analýza nejen k ocenění vlivu jednotlivých parametrů, ale též k lepšímu natrénování neuronové sítě. Výsledky získané touto metodou lze do budoucna označit za velice slibné.

Chapter 1

# INTRODUCTION

Everything that you could possibly imagine, you will find that nature has been there before you.

John Berrill

***Preface***

Nowadays a rapid growth of computer performance enables and encourages new developments in Civil Engineering as well as related areas. Particularly, within the field of structural mechanics, the modeling of materials and therefore the prediction of structural response is more accurate than in past decades. These are new challenges that we want to discover, but there are also several problems, that must be solved. For instance, the research within applied optimization is mainly lead by automotive and aerospace industries. Therefore, the emphasis is put mainly on the computational fluid dynamics domain and structural optimization area, especially on the shape optimization[1]. Because Civil Engineering problems are dominantly connected with static problems and topology and/or size optimization, there is a gap between current research and the application of new methods into a Civil Engineering area.

Therefore, the main goal of the proposed thesis is to review and enhance the current state of the art within the single and multi-objective optimization area and to show possibilities of these methods in several areas, such as the design of reinforced concrete (RC) structures and a material model parameters estimation.

## *1.1 Global vs. structural optimization*

When dealing with engineering problems, one may discovered two different areas of optimization - the first is usually called **Global optimization**[2]. By this term we will understand the optimization of a function or functions without any a-priori knowledge of the problem within these functions (sometimes called as “black-box” functions). This is an area where generations of mathematicians and also many economists have spent years of research. The second area, usually called **Structural optimization**, can be described as an applied science, where the methods from the Global optimization area are applied to a model of a structure or a material. At this point, the optimized function is no more black-box and adding some knowledge

[1] See Section 1.2 for notation of several types of structural optimization.

[2] From the mathematical point of view, the term **Global** is not correct, the term “numerical” will be more appropriate. Even though, the mathematical term “global” is usually used for problems, where the “global” optimum is sought, in the case of engineering tasks the methods used are exactly those of global optimization. Therefore, we will use the term **Global** in this general manner.

to help the optimization process to find desired solutions is not only advantage, but more often necessity.

Because this work deals with the application of several methods from the first area to the second, both domains will be investigated in very detail. We will start with the introduction of several forms of structural optimization with emphasis given to the design of frame structures. Chapter 2 aims to introduce and clarify several definitions, terms and methods from the Global optimization area, both for single and multi-objective optimization. As an addition, the handling of constraints will be discussed together with the current state of the art in parallelization of used algorithms. Chapter 3 is devoted to the comparison of four evolutionary algorithms applied to the suite of four black-box functions. Detailed and extensive numerical tests have been performed to examine the stability and efficiency of the proposed algorithms and several interesting results have been found. In Chapter 4, procedures for detailing and design of reinforced concrete structures will be presented together with proposed innovations in the field of the RC design. In particular, as one of the most frequent types of structures, the reinforced concrete frame structures will be investigated. Chapter 5 brings the problem of the applications of a genetic algorithm-based optimizer and an artificial neural network to material model parameters identification. The current application of this method to the microplane constitutive model reveals the limitations of the formulated identification problem and offers possible means for its regularization. The last Chapter 6 offers conclusion remarks as well as future perspectives in the domain of applied optimization within Civil Engineering applications.

## *1.2 Forms of structural optimization*

For better understanding of a new procedure proposed in Chapter 4 it is worthwhile to introduce the forms of structural optimization. In accordance with Prof. Grant Steven [Steven, 2003], four different forms of structural optimization can be distinguished. Each can be solved with a distinct optimization strategy and each form can include a reinforced concrete structure as a particular case. Note that solving real-world problems typically calls for combination of these forms.

### *1.2.1 Topology optimization*

By topology optimization we understand finding a structure without knowing its final form beforehand [Bendsøe and Sigmund, 2003]. Only the environment, optimality criteria and constraints are known. These tasks usually come from the mechanical engineering area, where designing parts of cars or aircraft are the most frequent topics. The major civil engineering representatives serve as a decision tool in selecting an appropriate static scheme of a desired structure. They are mostly applied to the pin-jointed structures, where the nodal coordinates of joints are optimization variables. Based on the position of supports and objective functions, several historically well-known schemes can be discovered (see Fig. 1.1). The typical example of this optimization form within the reinforced concrete area is placement of steel reinforcing bars into a concrete block. In other words, we search for the most suitable strut-and-tie model, see e.g. [Kim and Baker, 2002] or [Liang et al., 1999], in which the position of steel is not known in advance. In this case, the objective is usually minimization of amount of steel subjected to structural requirements.

In the first years of numerical optimization the traditional procedure for solving these tasks

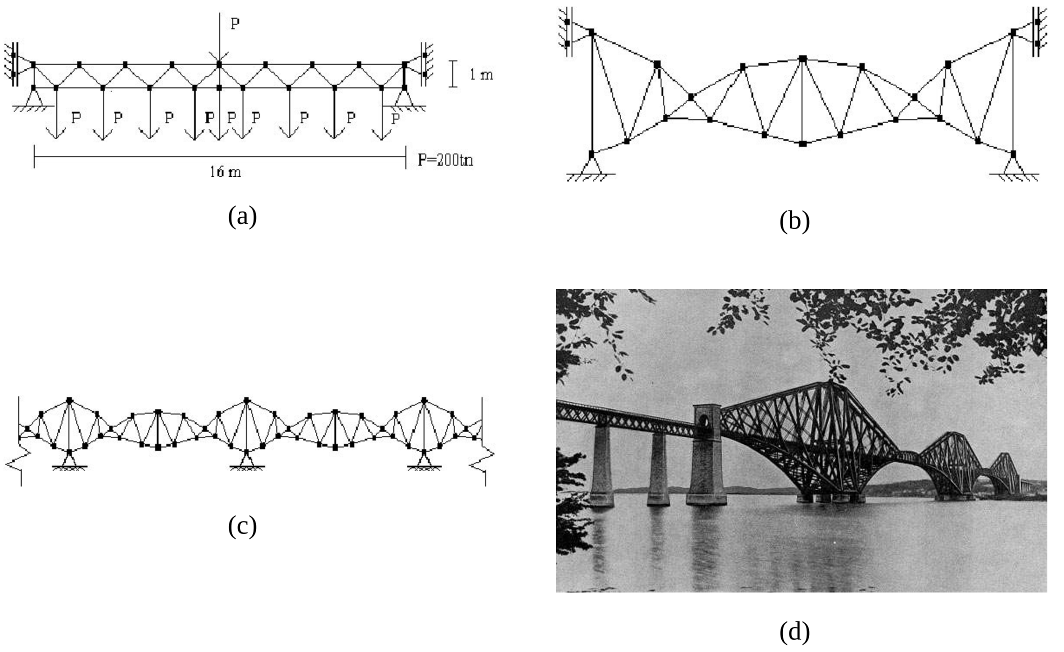


Figure 1.1: (a) Diagram for the calculation of the problem, (b) optimal solution of the problem, (c) optimized configuration formed by concatenating basic modules and (d) the First of Forth Bridge, built 1883–1890 as an example of the topology optimization presented in [Gil and Andreu, 2001].

was a fully stressed design, where stresses in all members aimed to be as close as possible to the material limits. The disadvantage is clearly visible for multiple loading cases or several support cases. Nevertheless, this approach leads to, in some sense, weighting result with application of the Lagrange multipliers for violated constraints.

Nowadays, the most frequently used methods herein to solve this class of problems are the *Optimality Criteria*[3] approach based on duality theory or convex programming [Olhoff, 1996], *homogenization* in connection with *Mathematical Programming* methods [Allaire, 2002] or [Cherkaev, 2000], *Evolutionary Structural Optimization* (**ESO**) [Xie and Steven, 1997] - another *hard-kill* method based on removing ineffective members from FE mesh, *Cellular Automata* - a very old dynamic simulation method studied since the 1960s [von Neumann, 1966] based on building block schemes with pre-defined behavior [Wolfram, 2002] and, finally, *Evolutionary Algorithms* (**EA**s) based on the principles of natural selection. For more details about the **EA**s, see Chapter 2.

### *1.2.2 Shape optimization*

In this form of optimization the topology of structure is known a-priori but there can be some part and/or detail of the structure, in which, for instance, high stresses can produce problems. Therefore the objective is usually to find the best shape that will result in the most suitable stress

[3] Referring to e.g. [Saka, 2003], the term *Mathematical Programming* will be assigned for gradient or approximation based methods while by *Optimality Criteria* we will understand the application of Karush-Kuhn-Tucker's optimality conditions.

distribution. Parameters of shapes are dimensions of the optimized parts or a set of variables describing the shape, e.g. coefficients of spline functions. Examples for the reinforced concrete area herein can be finding the proper shape of holes within plate members [Pedersen, 2000], the shape of a beam with holes [Yoshimura et al., 2002], the optimal shape of pre-cast retaining structures [Ceranic et al., 2001] or the ribs within box columns [Yoshimura and Inoue, 1995]. From the mathematical point of view, two representations of variables - continuous and discrete ones - can be found within the shape optimization area. The overview of the first case can be found, e.g. in [Sokolowski and Zolesio, 1992], and the latter case for the structural optimization is comprehensively summarized, for instance in [Bauer and Gutkowski, 1995].

Available algorithms for solving these tasks are representatives of *Mathematical Programming* [Haslinger and Neittaanmaki, 1996], again the **ESO**, a new method in this context is the *Simulated Biological Growth* based on the definition of a “fake” or artificial temperature [Mattheck and Burkhardt, 1990] and once more **EA**s.

### *1.2.3 Size optimization*

In this form of an optimization[4] a structure is defined by a set of sizes, dimensions or cross-sections. These are combined to achieve the desired optimality criteria. Within this area two main groups of structures can be distinguished.

**Discrete structures.** Here pin and rigid jointed structures can occur. In the case of steel structures in particular, nearly all possible optimization problems have been subjected to some form of investigation. To list a few successfully solved problems, optimization of structures with semi-rigid connections [Kameshki and Saka, 2001], optimization against buckling [Rong et al., 2001] or a finding minimum weight in connection with a minimum number of steel profiles used in a design [Greiner et al., 2001] and [Greiner et al., 2003] can be found in the corresponding literature. Many small-size examples from this area serve as benchmarks for different types of optimization algorithms, with the 10-bar truss [Belegundu, 1982] and the 25-bar space truss, e.g. [Adeli and Kamal, 1986] being the most often cited ones. Again, here all variables are selected from the pre-defined discrete admissible set. But this is not exactly the case of reinforced concrete frame structures which are more likely to be part of the next group of structures:

**Continuum structures.** This group contains beam-like structures defined by continuous variables, which are not known in advance in contrast to the previous case. The basic example is a beam with moments of inertia defined as a continuous variable [Lagaros et al., 2002]. All reinforced concrete optimization tasks, where the area of reinforcing steel is an unknown, will be the proper representatives of this group, too. Some examples of this class of problems are discussed in Chapter 4.

Once again, available optimization methods are gradient based *Mathematical Programming*, *Optimality Criteria* algorithms, *hard-kill* methods like the previously mentioned **ESO** and again **EA**s.

As a consequence of the definitions introduced above, we can distinguish one additional form of structural optimization. If a design variable - the size of a member or the material property - can reach zero value, i.e. it is not necessary in the structure and can be removed, then

[4] Also known as *Sizing problems*.

this type of optimization is often called **Layout Optimization**, e.g. [Kirsch, 1995]. The cornerstone of this approach is the so-called *ground structure*, which defines all possible positions of nodes and the set of all possible members/connections among these nodes. Then the goal is the removal of inefficient members to obtain an optimal structure. If coordinates of nodes are also unknown, this form becomes part of topology optimization, see Section 1.2.1. Therefore the layout optimization can be seen as the connection point between the previously cited two kinds of optimization.

An interesting feature in solving this form of optimization is the possibility of failure of *hard-kill* methods. In some cases a weak member is removed although it is necessary for the efficiency of the static scheme, see [Zhou and Rozvany, 2001] for more detailed discussion.

### *1.2.4 Topography optimization*

This form is the least investigated part of structural optimization. Here you can find the search for a proper shape for shell, membrane or tent like structures. Only few papers on this topic can be found in the literature, e.g. [Goslingt and Lewist, 1996] or [Schwarz et al., 2001], with even fewer dealing with reinforced concrete structures. And finally, the *Mathematical Programming* methods are known as the only efficient solutions for this type of optimization problems.

Chapter 2

# OPTIMIZATION METHODS

Natural selection can be seen as cheating for scientists who want to find discoveries only by perfect deduction.

Peter Convey, Roger Highfield:
Frontiers of Complexity

At the beginning, it is perhaps worthwhile to recall that the aim of this work is to improve the current state of the optimization methods applied to engineering problems with emphasis on (although not restricted to) the area of Civil Engineering ones. As was stated already in the Introduction part, we will deal almost exclusively with **Global optimization** methods, used below. This work is not aimed at thorough introduction and review of these methods. For a very complete overview, see the works of Arnold Neumaier [Neumaier, 2004] and [Neumaier, 2003].

From the mathematical point of view, an engineering task can be understood as a **multi-objective**, **constrained** and often **mixed integer-continuous** optimization problem (**CMOP**). Below we offer a formal definition:

**Constrained Multi-objective Optimization Problem.** A general **CMOP** includes a set of $n$ parameters (decision variables), a set of $k$ objective functions, and a set of $m$ constraints. The optimization goal is to

$$\begin{aligned}
\text{minimize} \quad & \mathbf{y} && = \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x})) && (2.1)\\
\text{subjected to} \quad & g_j(\mathbf{x}) && = 0 \;, \quad j = 1, \ldots, ne \;, && (2.2)\\
& g_j(\mathbf{x}) && \leq 0 \;, \quad j = ne+1, \ldots, m = ne + ni \;, && (2.3)\\
\text{where} \quad & \mathbf{x} && = (x_1, x_2, \ldots, x_n) \in \mathbf{X} \;, \quad \mathbf{X} \subset \{\mathbb{N}, \mathbb{R}\}^n \;, &&\\
& \mathbf{y} && = (y_1, y_2, \ldots, y_n) \in \mathbf{Y} \;, \quad \mathbf{Y} \subseteq \mathbb{R}^n \;, &&
\end{aligned}$$

$ne$ and $ni$ are the numbers of equalities and inequalities, respectively, $\mathbf{x}$ is the decision vector, $\mathbf{y}$ is the objective vector, $\mathbf{X}$ is denoted as the decision space and $\mathbf{Y}$ is called the objective space. Note that the set of all feasible solutions, i.e. all solutions $\mathbf{x}$ for which conditions (2.2) and (2.3) are satisfied, is denoted $\mathbf{X}_f$ and its image in the objective space is referred to as $\mathbf{Y}_f$, i.e. $\mathbf{Y}_f = \mathbf{f}(\mathbf{X}_f)$. Also note that we assume minimization hereafter, the statements for maximization or combined minimization/maximization are similar.

The most ambitious task of an optimization method is usually finding the *global* minimum or maximum, which yet fulfills the given constraints. Whether such a solution is "optimal" (from the point of view of achieving the desired practical goal) or not, is another question, which depends on the formulation of the problem, but not on the system of finding the optimum by the search method. Therefore, we will understand the optimization method as a *black box*

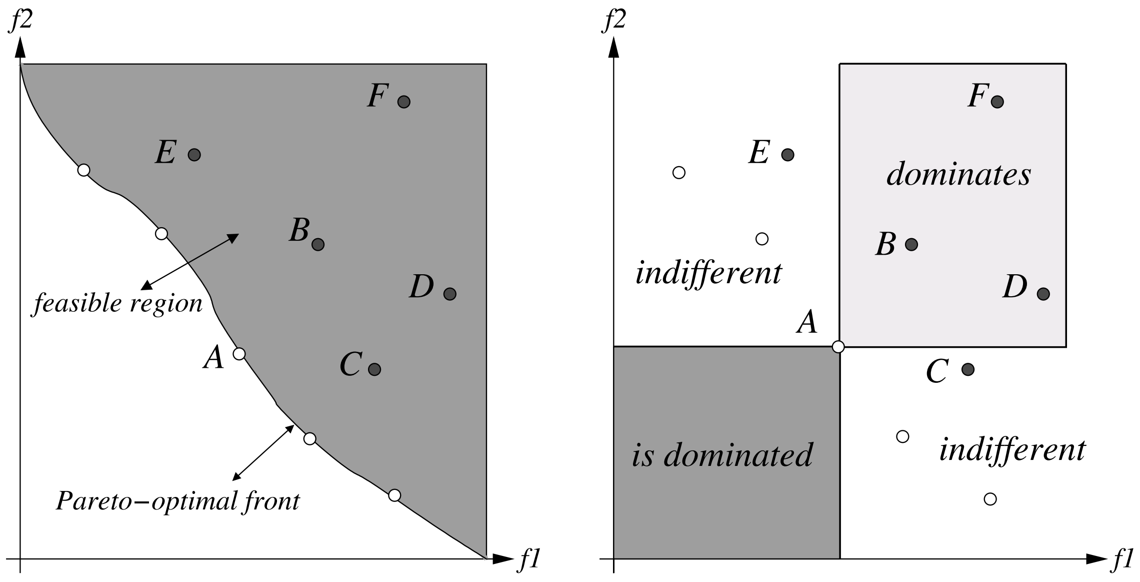


Figure 2.1: A graphical interpretation of the Pareto dominance.

application and the only “interesting” evaluation of a selected algorithm is its performance, i.e. usually reliability, in finding global optimum. Then, at least from the designer’s point of view, the problem might be seen as “simple” - “only” to find the superior optimization method for his problem. But relatively recently, the so-called **No free lunch theorem** was proven in [Wolpert and Macready, 1997] and therefore, as stated in [Fogel, 1999],

> . . . there is no best algorithm, whether or not that algorithm is “evolutionary”, and moreover whatever an algorithm gains in performance on one class of problems is necessarily offset by that algorithm’s performance on the remaining problems.

In other words, there is no best optimization algorithm. On the other hand, it does not mean that for a specific problem you cannot find a superior optimization algorithm. Furthermore, this is why there are so many available algorithms and tools for global optimization, see e.g. [Neumaier, 2003].

Yet, keeping in mind that our goal is to apply some optimization method to engineering problems, we cannot choose an arbitrary optimization algorithm. For example, a huge number of mathematical programming methods require continuity or convexity of the objective function. Many others require that derivatives of this function exist. But practice shows that engineering problems are not always continuous and, moreover, the derivatives cannot be obtained. This is the reason why Evolutionary Algorithms are so popular as the only input they need are the values of objective functions. Therefore, this work will primarily deal with these methods.

The next part will be devoted to the description of three main problems/aspects of optimization methods: tackling multi-objective nature, handling of constraints and selecting discrete or continuous domains to represent optimized variables. To bring the presented procedures closer to everyday use, one part of this section will focus on the parallelization of Evolutionary Algorithms.

## *2.1 Solving multi-objective problems*

In this work, attention is paid to the multi-objective nature of solved problems, because as was mentioned in the presentation by Eckart Zitzler during the EUROGEN 2001 conference [Zitzler et al., 2001]:

> Single objective optimization is a special case of multi-objective optimization (and not vice versa).

The main difference between single-objective and multi-objective optimization is that in the former case all solutions can be completely ordered according to the objective function $f$. In the latter case the solutions in $\mathbf{X}_f$ can be ordered only partially [Pareto, 1896]. As a consequence, in the case of single-objective problems (**SOP**s) only one global optimum exists[1], but in the case of multi-objective problems (**MOP**s) conflicting objectives can cause situation where no solution is superior to others. For the mathematical expression of the above mentioned statement we need to define the so-called **Pareto dominance**, see also Fig. 2.1.

**Pareto dominance.** For any two decision vectors $\mathbf{a}$ and $\mathbf{b}$,

$$\mathbf{a} \succ \mathbf{b} \quad (\mathbf{a}\text{ dominates }\mathbf{b}) \quad \textit{iff} \quad \forall i : f_i(\mathbf{a}) \leq f_i(\mathbf{b}) \wedge \exists i : f_i(\mathbf{a}) < f_i(\mathbf{b}), \tag{2.4}$$

$$\mathbf{a} \succeq \mathbf{b} \quad (\mathbf{a}\text{ weakly dominates }\mathbf{b}) \quad \textit{iff} \quad \forall i : f_i(\mathbf{a}) \leq f_i(\mathbf{b}), \tag{2.5}$$

$$\mathbf{a} \sim \mathbf{b} \quad (\mathbf{a}\text{ is indifferent to }\mathbf{b}) \quad \textit{iff} \quad \exists i : f_i(\mathbf{a}) < f_i(\mathbf{b}) \wedge \exists j : f_j(\mathbf{a}) > f_j(\mathbf{b}). \tag{2.6}$$

The definitions of the opposite binary relations $(\prec, \preceq, \sim)$ are analogical[2].

Then, a set of optimal trade-offs $\mathbf{X}_p \subseteq \mathbf{X}_f$ is called the global **Pareto optimal set**, iff

$$\forall \mathbf{x}_p \in \mathbf{X}_p : \nexists \mathbf{x} \in \mathbf{X}_f : \mathbf{x} \succ \mathbf{x}_p \; . \tag{2.7}$$

In other words, solutions from $\mathbf{X}_p$ cannot be improved in any of $k$ objectives unless the remaining objectives deteriorate[3]. Also note that the image of $\mathbf{X}_p$ in the objective space $\mathbf{Y}_p = \mathbf{f}(\mathbf{X}_p)$ is called the **Pareto optimal front** or **surface**, see also Fig. 2.2.

The traditional ways of solving the **MOP** are based on summations and/or different weighting methods, i.e. conversion of the **MOP** into the single criterion optimization, see e.g. the work [Miettinen, 1999] or the overview of all multi-objective algorithms [Marler and Arora, 2003] and its shorter version [Marler and Arora, 2004]. As a practical example, the author of this contribution in his previous research [Matouš et al., 2000] and [Lepš et al., 2002] used the total price of a structure as the unifying variable among different antagonistic constraints. As an unfavorable result, however, the multi-modal response of the objective function is obtained and the resulting optimization problem is very difficult to solve, see Section 2.2.4. Other problems cited in the literature for traditional solutions for the **MOP** are the sensitivity to the shape of the Pareto optimal front and the necessity to have extended knowledge about the given problem [Zitzler, 1999].

[1] or several solutions with the same objective value

[2] Recall, that we assume minimization here. For the maximization case consult e.g. [Zitzler, 1999].

[3] The difference between $\succ$ and $\succeq$ is only in the possibility of coexistence of two solutions $\mathbf{a}$ and $\mathbf{b}$ with the same profit, i.e. the case of $\mathbf{f}(\mathbf{a}) = \mathbf{f}(\mathbf{b})$ component-wise, which is a rare case in the engineering practice but can be important for a decision process.

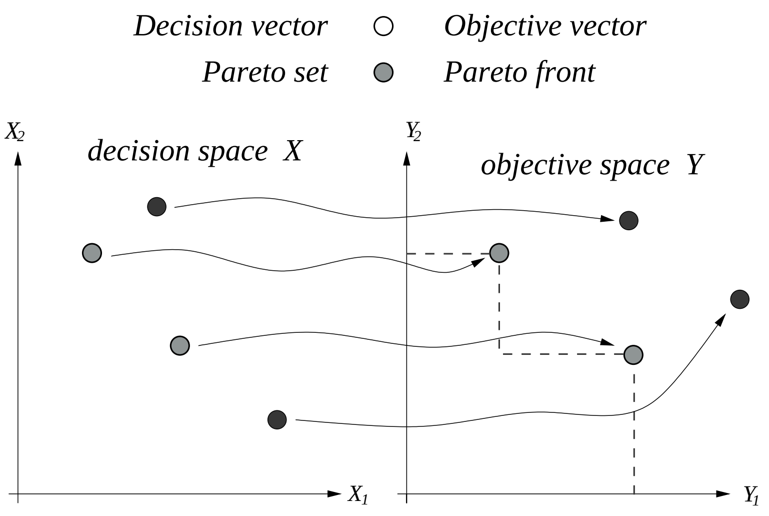


Figure 2.2: A basic notation for a multi-objective optimization.

In the last decades, Evolutionary Algorithms proved to be the proper solution strategy for the above mentioned problems. They can cover large domains and discover multiple trade-offs during one optimization run. Therefore, Section 2.2.1 will be devoted to a short description of Evolutionary Algorithms and their modification for the **MOP**.

## *2.2 Stochastic algorithms*

Although we have promised to introduce Evolutionary Algorithms, the starting point must be at a higher level, in the area of global stochastic optimization methods. The term "global" stands hereafter for the ability to find the global optimum in the case of an infinite number of iterations. This is usually handled by some random function, which explains the word "stochastic" in the title. The representatives of this group are blind *random search* (euphemistically called *Monte Carlo simulation*) [Pincus, 1970], *Tabu search*, see e.g. [Youssef et al., 2001], *Simulated Annealing* [Vidal, 1993] and its modifications [Ingber, 1995], different *Hill-climbing* algorithms [Yuret, 1994] or the already mentioned *Evolutionary Algorithms* which are described in more detail in the next subsection. For a more comprehensive summary of several stochastic methods see e.g. [Van Iwaarden, 1996].

### *2.2.1 Evolutionary Algorithms*

What mainly distinguishes Evolutionary Algorithms from others is the fact that they employ a set of possible solutions, often called *population*, instead of only one single search point. Therefore a new terminology must be introduced. The notation in this work is derived from the *Evolution Strategies* (**ES**s) [Bäck and Schwefel, 1995] and can be probably extended to all Evolutionary Algorithms.

**Evolutionary Algorithm's notation.** Let $\mu$ be the number of independently stored possible solutions for the given optimization problem (**OP**) that form a population $P^{(t)}$; $t$ stands for time or number of cycles. Then, if in every cycle of the **EA**, usually called *generation*, $\lambda$ new solutions are created, this algorithm will be denoted $(\mu \overset{+}{,} \lambda)$-**EA**. Moreover, $(\mu+\lambda)$ stands for selection of a new population for the next cycle $P^{(t+1)}$ from the union of $\mu + \lambda$ solutions and $(\mu\,,\lambda)$ denotes selection only from $\lambda$ members, respectively[4]. Next, three types of operators usually constitute the core of the algorithm:

$$\begin{array}{rlll} \text{recombination}^5 & \mathbf{rec}_i^j & : I^i \to I^j & i \le \mu\ , \quad j \le \lambda \\ \text{mutation}^6 & \mathbf{mut}_1^1 & : I \to I & \\ \text{selection} & \mathbf{sel}_i^j & : I^i \to I^j & i \le \mu \,||\, \mu{+}\lambda\ , \quad j \le \mu \end{array}$$

where the operator $\mathbf{opr}_i^j \in \{\mathbf{rec}, \mathbf{mut}, \mathbf{sel}\}$ denotes an output of $j$ solutions from $i$ input individuals[7], ${}^tI$ is a potential solution for a given problem[8] and ${}^tI^i$ is a set of $i$ possible solutions in time or generation $t$. The final scheme of an appropriate algorithm is the combination of the above mentioned operators **opr** and is repeated until some stopping criterion, usually the maximum number of function calls, is met.

To show the versatility of the proposed terminology and to highlight differences among evolutionary and single-point optimization methods, a number of (previously mentioned) very different and well-known optimization algorithms are classified according to the introduced scheme. Also note that all of the below mentioned algorithms are for the **SOP** only.

**Gradient methods** from the Mathematical Programming group are the best examples of the $(1\,,1)$-algorithm. They contain only one operator $\mathbf{x}_{t+1} = \mathbf{mut}_1^1(\mathbf{x}_t) = \mathbf{x}_t + \alpha_t \mathbf{d}_t$, where $\mathbf{d}_t$ is a direction of the descent and $\alpha_t$ is the step in the direction $\mathbf{d}_t$. Since the step $\mathbf{d}_t$ is assumed to be in a descent direction, the selection is redundant. Hence the general gradient-based algorithms can be written as

$$opt_{GRAD}({}^{t+1}I) = \mathbf{mut}_1^1({}^tI)\ . \tag{2.8}$$

**Simulated Annealing** (**SA**) is another traditional optimization method, which will be the best example of $(1+1)$-algorithm. Again, with $\mu = 1$, the only operator is mutation, in this case some random function. The actual implementation of the mutation operator is not important, it must be only ensured that each point in the space is visited at least once in the infinite number of runs. The core of this algorithm is a selection process

$$\mathbf{sel}_2^1(\mathbf{a}, \mathbf{b}) = \begin{cases} \mathbf{a}\ , & \text{iff} \quad u(0,1) \le p = \dfrac{1}{1 + e^{\Delta E/T}} \\ \mathbf{b} & \text{otherwise,} \end{cases} \tag{2.9}$$

[4] The mark $||$ will be used in the sense of $OR$ operator, i.e. the statement $\mu \,||\, \mu{+}\lambda$ will denote $\mu\ OR\ \mu{+}\lambda$.

[5] The term *recombination* will be used for cross-operations on more than one individual.

[6] A *mutation* will identify any change within one individual, no matter whether it is random or not.

[7] This notation is in contradiction to definitions in [Bäck and Schwefel, 1995], where indices $i$ and $j$ are swapped. The indexing used herein is more close to mathematical expressions like summations $\sum_{from}^{to}$ or progressions $\{\}_{from}^{to}$ .

[8] An individual $I$ is the encoded vector of design variables $\mathbf{x}$ (see Section 2.4) and in some **EA**s can also contain information on its history and/or parameters used in the time of its origin.

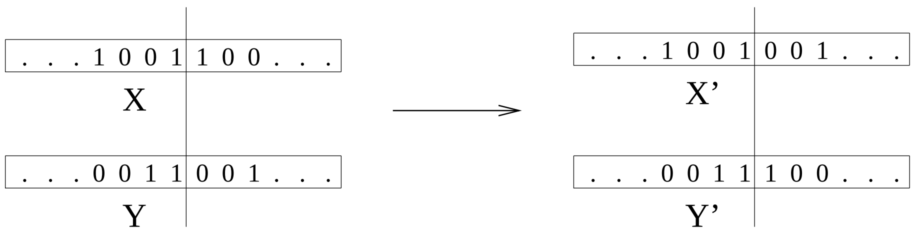


Figure 2.3: The cross-over operator.

where the energy difference is given by $\Delta E = f(\mathbf{b}) - f(\mathbf{a})$, $T$ is the artificial temperature determined by the so-called cooling schedule, usually in the form

$$T \approx \frac{T_0}{\ln t} \tag{2.10}$$

provided $T_0$ is sufficiently large, see e.g. [Ingber, 1993], and finally, $u \sim \mathbf{U}(\cdot,\cdot)$ is a realization of a uniformly distributed random variable from a given domain. The beauty of this algorithm is a non-zero probability $p$ that enables replacement of a better solution with a worse one. Therefore, the whole algorithm can be written in the following form

$$opt_{SA}({}^{t+1}I) = \mathbf{sel}_2^1(\mathbf{mut}_1^1({}^tI), {}^tI)\ . \tag{2.11}$$

**Simple Genetic Algorithms** (**SGA**s) are often cited as the oldest Evolutionary Algorithms, even though the Evolution Strategies (**ES**s) are actually older, see below. The reason why is that **SGA**s had been predicted (but not discovered yet) by J. Holland in [Holland, 1975] during his work on *Cellular Automata*. Later on, the topic was studied in more detail, including the proofs of convergence. Especially a book by D. E. Goldberg [Goldberg, 1989] is the most popular publication that deals with this topic. The **SGA**s follow an analogy of processes that occur in living nature within the evolution of live organisms during a period of many millions of years. Simple genetic algorithms treat individual solutions, here called *chromozomes*, as binary strings (see Section 2.4.2). This kind of representation seems to be very convenient[9] for optimization problems coming from a combinatoric area (e.g., the traveling salesman problem).

Based on binary coding, the cross-over and mutation operators have usually the following form. Firstly, the cross-over operator chooses two chromozomes, so-called *parents*, and then creates their two descendants (*children*) using the following operation: it selects a position inside the binary string and starting from this position exchanges the remaining parts of the two chromozomes (see Figure 2.3). Secondly, the mutation randomly alters one or more bits in the binary strings of new solutions.

The next specific feature of the **SGA** is the selection for a reproduction cycle at the beginning of the algorithm. New solutions are created with cross-over or are copied into a new population. This can be classified as a $(\mu + \lambda)$-algorithm. Therefore, the scheme of the algorithm is

$$opt_{SGA}({}^{t+1}I^\mu) = \mathbf{mut}_1^1(\mathbf{rec}_2^2({}^{t+1}\bar{I}^\lambda), {}^{t+1}\bar{I}^\lambda)\ , \quad {}^{t+1}\bar{I}^\lambda = \mathbf{sel}_\mu^\lambda({}^tI^\mu)\ , \tag{2.12}$$

[9] The main cited advantage of a binary coding is the maximum information carried by a one bit.

where the "elitist" set ${}^{t+1}\bar{I}^{\lambda}$ of the $\lambda \leq \mu$ cardinality is usually called the *mating pool*.

Although the codings, operators and proofs of convergence were initially based on the binary basis, nowadays real-encoded and other alphabets based genetic algorithms **GA**s have proved their reliability and are widely used for solutions of real-world problems, see e.g. [Michalewicz, 1999, and references therein].

**Evolution Strategies** (**ES**s) were developed as $(1+1)$ strategy in 1964 at the Technical University of Berlin by Rechenberg and Schwefel as an experimental optimization technique. As the entirely first *Evolutionary Algorithm* can be considered the $(\mu+1) - ES$ strategy presented in [Rechenberg, 1973]. Since then **ES**s have developed into very robust tools with many $(\mu+\lambda)$ and $(\mu\,,\lambda)$ alternatives. The comprehensive notation used and extended herein, see [Bäck and Schwefel, 1995], as well as the proofs of convergence [Schwefel and Bäck, 1995] and self-adaptive control [Eiben et al., 1999] predetermines these algorithms for a wide range of applications. Moreover, the fact that from the starting point they were developed as real-coded, moves **ES**s into engineering methods rather than mathematical[10]. To conclude this paragraph, the notation of these methods cannot be missed:

$$opt_{(\mu+\lambda)-ES}({}^{t+1}I^{\mu}) = \mathbf{sel}^{\mu}_{\mu+\lambda}(\mathbf{mut}(\mathbf{rec}({}^{t}I^{\mu})), {}^{t}I^{\mu}) \ , \tag{2.13}$$

$$opt_{(\mu\,,\lambda)-ES}({}^{t+1}I^{\mu}) = \mathbf{sel}^{\mu}_{\lambda}(\mathbf{mut}(\mathbf{rec}({}^{t}I^{\mu}))) \ , \tag{2.14}$$

where the recombination operator is more or less similar to real-coded genetic algorithms' cross-over operators and the mutation usually alters a solution by a normally (Gaussian) distributed random number.

A huge number of other evolutionary methods not explicitly cited above are usually a combination of the above mentioned ideas and in the view of the **No free lunch theorem** any of them can be successful in solving the given single-objective optimization problem.

### *2.2.2 Parameters tuning*

Besides many advantages that Evolutionary Algorithms have, they are connected with several difficulties. In addition to the demand for thousands of evaluations of an objective function, it must be noted, that one of the most important obstacles is the finding of proper parameters - either the cooling scheme in **SA**, see Eq. (2.10) or Eqs. (3.18) and (3.27), or several cross-over and mutation parameters, see Appendix C for parameters of four **EA**s used within this work. During last several years, this problem has been solved usually by minimizing needed parameters [Harik and Lobo, 1999], usually in some form of self-adaptation [Saravanan et al., 1995]. For an overview of self-adaptation see references [Hinterding et al., 1997] and [Eiben et al., 1999]. However, it can be stated, that there is a dependency between a number of parameters and the popularity of the method - lower a number of parameters, higher a number of satisfied users. See also Section 3.3.1 and 3.3.2 for such examples of low-number-of-parameters methods.

[10] The place of origin, too, plays an important role - the **SGA**s are favored in English-speaking countries, on the other side, the **ES**s are very popular in Europe.

### *2.2.3 Multi-objective Evolutionary Algorithms*

As mentioned earlier, the group of **EA**s seems to be suitable for solving **multi-objective problems** (**MOP**). The reason is that the use of a population of possible solutions can easily cover a searched Pareto optimal set. Referring to [Coello, 2003], two generations of **Multi-objective Evolutionary Algorithms** (**MOEA**s) can be distinguished. In the case of the first one, to evaluate each individual its distance (or Pareto dominance) to already found or a-priori known Pareto optimal set is used. This relatively simple idea was firstly implemented in 1984 by David Schaffer in his Vector Evaluated Genetic Algorithm (**VEGA**) [Schaffer, 1984]. The core of this first group is built upon algorithms like the Multi-Objective Genetic Algorithm (**MOGA**) [Fonseca and Fleming, 1993], the Niched-Pareto Genetic Algorithm (**NPGA**), presented e.g. in [Horn and Nafpliotis, 1993], and also the Nondominated Sorting Genetic Algorithm (**NSGA**) [Srinivas and Deb, 1994]. The second generation of **MOEA**s is characterized by the idea of *elitism* which is usually implemented in the form of externally stored solutions from an already found Pareto optimal set[11]. This group is represented by algorithms like the Strength Pareto Evolutionary Algorithm (**SPEA**) [Zitzler and Thiele, 1999] and especially its second version **SPEA2** [Zitzler et al., 2001]. It is also worth to mention the second version of the **NSGA** algorithm - **NSGA II** [Deb et al., 2000], the Micro Genetic Algorithm (**MGA**), see e.g. [Coello and Pulido, 2001] or [Annicchiarico, 2003], and finally, the Pareto Archived Evolution Strategy (**PAES**) [Knowles and Corne, 2000].

From a general point of view, two conflicting objectives in solving multi-objective problems are often cited: the exploration and the exploitation. The first one deals with the level of diversity in a population and the second with the convergence to the Pareto optimal set. The former one must be solved inevitably using evolution of a population and therefore will be solved as a part of the current Evolutionary Algorithm. In spite of this, exploration should be the basic ability of all **EA**s and hence fulfilment of this criterion should always be ensured. Therefore the important (but not unique) characteristic of any **MOEA** will be its ability to get close to an optimal set. As mentioned previously, the convergence to the desired Pareto optimal set is in the most modern algorithms tackled by a set of *elitist* solutions. And following ideas presented in [Zitzler, 1999], the approach used in many of the previously mentioned multi-objective algorithms can be generalized for any **SOP** Evolutionary Algorithm. Particularly, for the **SPEA** algorithm, the management of an elitist set obtained from $ANY - EA$ can be written as

$$\begin{aligned} opt_{SPEA}({}^{t+1}I^{\mu}, {}^{t+1}\bar{I}^{\bar{\mu}}) &= (\mathbf{sel}^{\bar{\mu}}_{\mu+\bar{\mu}}({}^{t+1}I^{\mu}, {}^{t}\bar{I}^{\bar{\mu}}), {}^{t+1}I^{\mu}) \;, \\ {}^{t+1}I^{\mu} &= opt_{ANY-EA}({}^{t}I^{\mu} \mid\mid {}^{t}I^{\mu} + {}^{t}\bar{I}^{\bar{\mu}}) \;, \end{aligned} \tag{2.15}$$

where ${}^{t}\bar{I}^{\bar{\mu}}$ is an elitist set (*archive*) of already found Pareto optimal solutions in time $t$ that contains $\bar{\mu}$ items. Note that in the **SPEA** algorithm the $\bar{\mu}$ is not constant during time, but is increasing from some starting value, e.g. $0$, to a given limit value $\bar{\mu}_{max}$.

This area is still unexplored and many applications and research results on this subject are published every month. For example, Carlos A. Coello Coello's "List of References on Evolutionary Multi-objective Optimization" [Coello, 2004] actually contains more than 1,400 references of papers or technical reports dealing with evolutionary multi-objective optimization. Note also, that the **No free lunch theorem** is valid here too, see [Corne and Knowles, 2003] for more details.

[11] Usually called *archive*.

To conclude this section, it must be emphasized that the difference between single and multi-objective optimization is not only at the programming level, but also in the system of gathering information from an output. While in the case of the single-objective optimization, the designer is forced to use usually one global optimum found by any algorithm, in the second case there is a set of different solutions and the designer must decide and choose the appropriate structure. This domain of research is called Decision Making and is usually solved by algorithms from Operational Research. For a small review on Decision Making concerning evolutionary multi-objective optimization (**EMOO**), see e.g. [Coello, 2000b].

### *2.2.4 Note on multi-modal optimization*

The **multi-modal optimization** term is usually used for problems with a several number of local minima. Such a response (or landscape) is typical for engineering problems, where especially constraints can cause a local valley on the path to the global optimum. The problem of being trapped in a local minimum, in the **EA**s area called the *premature convergence*, goes throughout all optimization algorithms, starting from gradient optimizers and ending in Evolutionary Algorithms. Even when using the Simulated Annealing method such a situation can frequently emerge. At this point, several optimization problems can be distinguished. The most common and in this situation the "minimal" task is searching for the exactly one global optimum. On the opposite side, and therefore here called "maximal" problem, is finding all local optima for a given multi-modal function. However, the requirement from engineering practice will be typically a combination of these extreme cases. The traditional "in-direct" solution for the minimal problem is restarting a search from a different point in gradient methods or a new search with a different starting population in Evolutionary Algorithms. In the Simulated Annealing method, except re-starting, also re-annealing (change of temperature) can be used for the same purposes (see Section 3.3.3 for one particular implementation). As a "direct" solution we understand the management of previously discovered local minima and some procedure for avoiding the next visit in these points. This is done by the so-called *niching* algorithms, which store the found local optima and penalize solutions in their close neighborhood. These methods are also used for the maximal case, where one needs to discover all sub-optimal solutions. A comprehensive review and many suggestions on the maximal case can be found in [Mahfoud, 1995b, and references therein]. For the minimal case different solutions exist, concerning this topic the most recent work of A. Kučerová [Hrstka and Kučerová, 2004] looks promising. Nevertheless, the existence of a huge number of multi-modal solvers leads to an idea that the **No free lunch theorem** can be extended to these methodologies, too. This can be supported e.g. by research done in [Mahfoud, 1995a], where different niching methods were superior in solving different levels of difficulty. In spite of this, the main conclusion from Mahfoud's work is that parallel methods are better than sequential ones, not only from the point of reliability but also in terms of speed. These findings are in agreement with the results obtained from the area of parallel Evolutionary Algorithms (see the next section).

Last but not least, let us note that the multi-modal behavior in engineering problems is mainly caused by single-objectivization [Knowles et al., 2001], i.e. by the combination of different, usually conflicting, objectives into only one. This is so inappropriate intervention into the process of finding an optimal solution that the multi-objective methodology presented above in Section 2.1 and later seems to be rather a necessity than a choice.

### *2.2.5 Parallel Evolutionary Algorithms*

The most common reason for using parallel implementation of **EA**s (except for utilization of existing parallel computers) is the shortening of a computational demand of an optimization process. The requirement on computational resources leads to the development of **parallel** or, as they are more often called, **distributed Evolutionary Algorithms** (**PEA**s)[12]. Moreover, it was also discovered, that these algorithms are superior in solving multi-modal objective functions (see previous section), but still, such algorithm can be implemented on a sequential platform. Therefore, the term parallel Evolutionary Algorithm will be hereafter devoted only to algorithms implemented in parallel environment like shared memory systems or clusters of personal computers. Especially the latter case will be the most emphasized one in this work, mainly because it aims at small engineering companies usually equipped with this type of networks and computers.

Throughout the last two decades the terminology used within **PEA**s became stable, therefore one can distinguish three kinds (or topologies) of **PEA**s [Cantú-Paz, 1997]. The first one is created by the **fine-grained** Evolutionary Algorithms, where small subpopulations are spread into grid-like space. The connection among subpopulations is characterized by high frequency communication with the nearest neighbors. This scheme is well suited for massively parallel systems, where the nearest processors are directly connected and therefore the communication cost is low. The second one, the **coarse-grained** (or **island**) **EA**s, consists of much smaller number of subpopulations, usually formed in star or circular shapes. The arbitrary connectivity is followed by a low frequency exchange of optimal solutions, here called *migrants*, among individual "islands". Although the topology predetermines this methodology to the cluster of computers, other platforms can be used as well. The last one - the **global parallel model** - is based on the master-slave paradigm, where the *master*, or *root*, processor runs an optimization algorithm and other processors, called *slaves*, solve only the objective function. This is based on the idea of the independence of individual solutions within one generation, here called "implicit parallelism" [Mahfoud and Goldberg, 1995][13]. The first advantage of this last model is independence on the hardware platform, i.e. it can be run on shared memory systems as well as on the distributed memory ones. The second main feature is that the global parallel model produces the same behavior as his serial ancestor. This is mainly characterized by the same number of tuning (or control) parameters, which is not fulfilled for the first two topologies. And again, our goal usually deals with optimization of a design process that inevitably contains finite element analysis, which is usually the most expensive part of an optimization process. Therefore, the distribution of a more complex analysis part is suitable. Moreover, the chosen parallel computing scheme ensures constant distribution of the work among the processors provided that the time spent on evaluation of two solutions does not differ. Although this condition is not regularly met, especially for nonlinear analyses, it appears that the sufficiently high number of solutions assigned to each processor eliminates this disadvantage. The missing multi-modal solving ability (because it has no parallel subpopulation) is the main disadvantage in comparison with two others, but can be solved with previously mentioned methods, see Section 2.2.4.

To sum the above lines up, it seems that the global parallel model is the proper choice for the optimization of engineering tasks. The last cited advantage is the possibility to easily check

[12] The bibliography [Alander, 1995] contains more than 500 references on this topic

[13] An Evolutionary Algorithm in this case is not divided into processors and hence it is not *distributed*, but the whole algorithm is parallel and therefore belongs to **PEA**s.

the optimum amount of required processors for a given problem. It can be simply estimated by the following relation [Cantú-Paz, 2001]

$$P^* = \sqrt{\frac{nT_f}{T_c}} \; , \tag{2.16}$$

where $P^*$ is the optimal number of processors, $n$ is the number of solutions in a population, $T_f$ is time for one evaluation of an objective function and $T_c$ is the latency time - hardware dependent variable which is spent on starting communication between two processors.

## *2.3 Handling of constraints*

Thus far we have supposed that the optimal solution is chosen from the feasible set of solutions, i.e. from $\mathbf{X}_f$. In the case of constrained optimization there arises the need to tackle the problem of promising solutions that, unfortunately, violate some constraints. In the literature several strategies can be found, but we will limit our attention only to methods that are easily applicable to the Evolutionary Algorithms nature and have proved their reliability for engineering optimization tasks. Note that traditional methods come from the **SOP** area and therefore the adopted notation will be for one objective function only. One example of multi-objective approaches will be introduced in the last part of this section.

### *2.3.1 Death penalty approach*

The term "death penalty" stands for the rejection of an infeasible solution from a search process. The advantage of this strategy is its easiness, the disadvantage can occur in problems, where the feasible domain is not convex or is divided into a number of disjoint parts. Also in the case of highly constrained problems, where the problem of finding the first feasible solution can arise, this method usually fails. To overcome these obstacles, the "death penalty" is often combined with repair or problem-dependent search operators, see e.g. [Michalewicz and Nazhiyath, 1995] or [Schoenauer and Michalewicz, 1997].

### *2.3.2 Penalty function methods*

Note that in the present work we use only *exterior* penalty functions which penalize infeasible solutions, which is in contrast with the *interior* penalty approach that penalizes feasible solutions near the boundary of a feasible domain. The former one admits infeasible solutions during the whole optimization process and therefore cannot ensure the feasibility of the found optimum. On the other hand, the big advantage is that the optimization can start everywhere. Therefore this procedure is much more flexible than the other one. The latter works only with feasible vectors, therefore the found optimum as well as intermediate solutions always fulfill the given conditions. The disadvantages are clear - this procedure cannot work with equality constraints (because it is almost impossible not to violate them) and must start in the feasible area.

The *exterior* penalty approach is one of the most often used approaches for handling constraints, especially within the Evolutionary Algorithms community. The basic idea is to move the solution from the infeasible to feasible space by adding some value to the objective function, i.e.

$$f_f(\mathbf{x}) = f(\mathbf{x}) + Q(\mathbf{x}) \; , \tag{2.17}$$

where $Q$ is equal to zero if the solution is feasible or equals some positive value (in minimization problems) otherwise. The value of $Q$ can be defined on the three different bases:

1. An individual is penalized only for its unfeasibility, with its distance from the feasible set playing no role.

2. The value can be defined as a measure of distance from the feasible domain or

3. as a price or energy spent to repair such a solution.

In practice, the definition of a penalty function can take several forms. In a general form, the most common implementation can be written as

$$Q(\mathbf{x}) = \lambda(\tau) \sum_{j=1}^{ne} g_j(\mathbf{x})^\alpha + \lambda(\tau) \sum_{j=ne+1}^{ne+ni} \min[0, g_j(\mathbf{x})^\beta] \, , \tag{2.18}$$

where $\alpha$ and $\beta$ are usually constants equal to $1$ or $2$. In the case that the function $\lambda(\tau)$ does not change in time, it is called *static*, in the opposite case *dynamic*. In the latter case, the function is usually assumed to be increasing with respect to “time” or “temperature” $\tau$ to ensure feasibility in the last stages of the optimization process. For a more comprehensive overview of different forms of penalty functions, see e.g. [Coello, 2000c]. The not-so-strict requirements on penalty functions enable formulation of a problem-dependent penalty function or different engineering-like forms of penalty terms (see e.g. [Lepš and Bittnar, 2001] or Eq. 3.4 in Section 3.2.3) and hence increase the popularity of this approach.

### *2.3.3 Constraints as objectives*

A completely new and revolutionary approach has been presented by Carlos A. Coello Coello in [Coello, 2000a]. Even though his work is not the first one on this subject, it is closer to an optimal solution than other approaches. The idea is very straightforward. During an optimization process we admit infeasible solutions but in the end we want to obtain only the feasible ones, i.e the Pareto set. This can be done by minimizing the distance between infeasible and feasible regions. And this is nothing more than next optimization process and, therefore, it can be included in the original multi-objective one. Generally speaking, we deal with a new optimization problem:

**Multi-objective solution for constrained problems.** Again, we deal with a set of $n$ parameters (decision variables), a set of $k$ objective functions, and a set of $m$ constraints. The optimization goal is to

$$
\begin{aligned}
\text{minimize} \quad \mathbf{y} &= \mathbf{e}(\mathbf{x}) = (e_1(\mathbf{x}), e_2(\mathbf{x}), \dots, e_K(\mathbf{x})) \; , \quad K = k + m \; , && (2.19)\\
\text{where} \quad e_i(\mathbf{x}) &= f_i(\mathbf{x}) \; , \quad i = 1, \dots, k \; , && (2.20)\\
e_i(\mathbf{x}) &= g_j^2(\mathbf{x}) \; , \quad i = k+1, \dots, k+ne \; ; \quad j = 1, \dots, ne \; , && (2.21)\\
e_i(\mathbf{x}) &= \begin{cases} g_j(\mathbf{x}) \; , & \textit{iff} \quad g_j(\mathbf{x}) > 0 \; , \\ 0 & \text{otherwise}, \quad i = k+ne+1, \dots, K \; , \end{cases} && (2.22)\\
& \quad j = ne+1, \dots, m = ne + ni \; .
\end{aligned}
$$

Now this problem has become an unconstrained **MOP** and any appropriate multi-objective solver can be applied. Note that in the equation (2.21), the absolute value instead of the square power of the equalities $g_j(\mathbf{x})$ can be used.

The next advantage can be seen in the ability of finding promising solutions in problems with no feasible regions, i.e. solutions with the minimal distance to an admissible domain. As we assume the application of Evolutionary Algorithms, the disadvantage will be the $\mathcal{O}(KN^2)$ computational cost, where $N$ is the number of simultaneously compared solutions in the Pareto dominance procedure. To overcome this obstacle, the methodology based on the order of solutions in terms of violated constraints is also proposed in [Coello, 2000a]. This procedure and other possibilities have not been systematically studied yet and await a more detailed investigation.

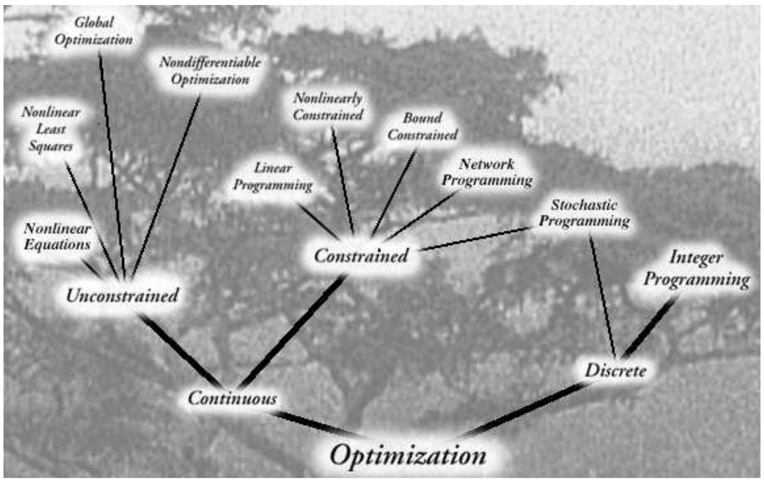


Figure 2.4: The division of optimization algorithms in accordance with [NEOS Guide, 1996].

### 2.4 Discrete vs. continuous optimization

If an interested reader visits e.g. web pages of the NEOS server [NEOS Guide, 1996] with a huge number of global optimization methods, one thing he or she can find is some classification of these algorithms. In this particular example, the tree of optimization methods, as depicted in Fig. 2.4, can be found. If we recall different forms of structural optimization (Section 1.2), two representations of design variables are mainly used - discrete and continuous, respectively. From Fig. 2.4 it seems that only one sort of methods, particularly the stochastic programming, can be used for problems with both representations. The real situation is not so strict because the transformations among different encodings are available. It is worthwhile to mention that we presume a computer implementation of the proposed procedures, therefore internal representation of variables in a computer must be taken into account too.

In historical context, several codings or alphabets have been studied and applied within the optimization methods area. Until now the most often used codings have been the following ones: *real* or *continuous*, *integer*, *binary* - popular especially in the starting years of Evolutionary Algorithms [Goldberg, 1989], *trees* - nowadays e.g. in **Genetic Programming** methods [Koza, 1992], *lists* - for the so-called **traveling salesman problem** (TSP) and similar tasks, see e.g. [Michalewicz, 1999], or different alphabetic codings, e.g. **DNA computing** [Adleman, 1994]. We will limit our attention only to the first three cited and their computer implementation. Also the speculation on the influence of these codings on the algorithms performance will be left out[14]. Moreover, as shown in the next chapter, the results of real and integer coding-based Evolutionary Algorithms are comparable.

### *2.4.1 Integer coding*

Recall the fact that we deal with integers or continuous real numbers. In the case of a discrete set of real numbers, the indices will play the role of alternative variables. Consider $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$ as a vector of $n$ variables, integer or real numbers $x_i$, defined on a closed interval $min_i \leq x_i \leq max_i$ on an appropriate domain $\mathbf{x} \in \{\mathbb{N}, \mathbb{R}\}^n$. Further assume that each variable $x_i$ is to be represented with some required precision $p_i$, defined as the smallest unit the number $x_i$ can attain. Then, each variable $x_i$ can be transformed into a nonnegative integer number $z_i \in \mathbb{N}_0^+$ as

$$z_i = \left\lfloor \frac{x_i - min_i}{p_i} \right\rfloor , \tag{2.23}$$

where operator $\lfloor \cdot \rfloor$ denotes the integer truncation. The inverse transform is given by

$$x_i = z_i \, p_i + min_i \; . \tag{2.24}$$

For instance, the real number $314.159$ with precision $p = 0.001$ and minimum $min = 0.0$ is transformed into the integer number $314159$ and vice versa.

This is also the meet-point to the binary coding, because the encoded vector $\mathbf{z}$ can be operated as a binary string using native properties of computer memory data and/or can be used as a vector of integer numbers.

### *2.4.2 Binary coding*

In accordance with the previous section, we may deal with a problem of decoding an integer vector $\mathbf{z}$ into a binary string provided that this operation is not covered by the selected programming language. First, the length $q$ of the final binary string is needed and can be computed e.g.

[14] Actually, there is a mathematical proof, that these representations are equivalent, see e.g. [Fogel, 1999].

by

$$q = \left\lfloor \frac{\ln(max_i - min_i) - \ln(p_i)}{\ln\ 2} \right\rfloor + 1\ , \tag{2.25}$$

where the notation is the same as in the previous section. Then the binary string $\mathbf{B}^i \in \{0;1\}^q$ pertinent to the integer number $z_i$ can be obtained by

$$B^i_j = z_i/2^{j-1} \mod 2\ , \quad j = 1, \ldots, q\ . \tag{2.26}$$

As a result, $\mathbf{B}^i$ will contain the list of bits of the number $z_i$ in the ascending order, i.e. from the least to the most important one. The recursive relationship is given by

$$z_i = \sum_{j=1}^{q} B^i_j\ 2^{j-1}\ . \tag{2.27}$$

### *2.4.3 Real coding*

The beauty of this encoding can be seen in its easiness, because in common programming languages the assignment

$$\mathbf{z} = \mathbf{x}\ , \quad \mathbf{z} \in \mathbb{R}^n\ , \quad \mathbf{x} \in \{\mathbb{N}, \mathbb{R}\}^n \tag{2.28}$$

will usually work. The opposite direction can be handled with some truncation or rounding procedure. For example,

$$x_i = \lfloor z_i\ /\ p_i + 0.5 \rfloor\ p_i \tag{2.29}$$

is the most common.

Chapter 3

# A COMPETITIVE COMPARISON OF FOUR EVOLUTIONARY ALGORITHMS

Who cares how it works, just as long as it gives the right answer?

Jeff Scholnik

## *Preface*

This chapter was originally produced as a joint work of four authors: O. Hrstka, A. Kučerová, J. Zeman and the author of this thesis. This contribution was firstly published as a conference paper [Hrstka et al., 2001] and later on it was, in a revised form, accepted to the international journal [Hrstka et al., 2003]. Because this research is an essential part of the author's work, it is presented here, enhanced by the notation introduced in the first chapter.

A comparison of several stochastic optimization algorithms developed by the above-introduced authors in their previous works for the solution of some problems arising in Civil Engineering is presented. These optimization methods are: Integer Augmented Simulated Annealing (**IASA**), Real-coded Augmented Simulated Annealing (**RASA**) [Matouš et al., 2000], Differential Evolution (**DE**) in its original fashion developed by R. Storn and K. Price [Storn, 1996] and Simplified Real-coded Differential Evolution (**SADE**) [Hrstka and Kučerová, 2000]. Each of these methods was developed for some specific optimization problem; namely the Chebychev trial polynomial problem, the so-called *type 0* function and two engineering problems – the reinforced concrete beam layout and the periodic unit cell problem respectively. Detailed and extensive numerical tests were performed to examine the stability and efficiency of the proposed algorithms. The results of our experiments suggest that the performance and robustness of the **RASA**, **IASA** and **SADE** methods are comparable, while the **DE** algorithm performs slightly worse. This fact together with a small number of internal parameters promotes the **SADE** method as the most robust one for practical use.

## *3.1 Introduction*

Nowadays, optimization has become one of the most discussed topics of engineering and applied research. Advantages coming from using optimization tools in engineering design are obvious. They allow to choose an optimal layout of a certain structure or a structural component from the huge space of possible solutions based on a more realistic physical model, while the traditional designing methods usually rely only on some simple empirical formulas or guidelines resulting in a feasible but not necessarily an (sub-)optimal solution. Using optimization as a method of design can raise engineering job to a higher level, both in terms of efficiency and reliability of obtained results.

Typically, optimization methods arising in engineering design problems are computationally demanding because they require evaluation of quite a complicated objective function many times for different potential solutions. Moreover, the objective function is often multi-modal, non-smooth or even discontinuous, which means that traditional, gradient-based optimization algorithms fail and *global optimization techniques*, which generally need even a larger number of function calls, must be employed. Fortunately, the rapid development of computational technologies and hardware components allows us to treat these problems within a reasonable time.

As indicated previously, the optimization methods can be divided generally into two groups: the gradient methods, which operate on a single potential solution and look for some improvements in its neighborhood, and global optimization techniques – represented here by *Evolutionary Algorithms* – that maintain large sets (populations) of potential solutions and apply some recombination and selection operators on them. During the last decades, Evolutionary Algorithms have received considerable attraction and have experienced a rapid development. Good compendium of these methods can be found for example in [Michalewicz et al., 1997] and references therein. The main paradigms are: *Genetic Algorithms* (binary or real coded), *Augmented Simulated Annealing* (binary or real coded), *Evolution Strategies* and *Differential Evolution*. Each of these methods has many possible improvements (see, e.g., [Andre et al., 2000] or [Fan et al., 2000]).

Many researchers all over the world are united in an effort to develop an Evolutionary Algorithm that would be able to solve reliably any problem submitted to it. Currently, there is no such method available. Each method is able to outperform others for certain type of optimization problem, but it extremely slows down or even fails for another one. Moreover, many authors do not introduce a reliable testing methodology for ranking their methods. For example they introduce results of a single run of a given method, which is rather questionable for the case of stochastic algorithms. Finally, the methods are often benchmarked on some test functions, that even if presented as complicated, are continuous and have few local extremes.

This paper presents several optimization methods that were developed and tested for different types of optimization tasks. Integer Augmented Simulated Annealing (**IASA**), derived from a binary version of the algorithm [Lepš and Šejnoha, 2000], was developed to optimize a reinforced concrete beam layout from the economic point of view. For solving the problem of a periodic unit cell layout [Zeman and Šejnoha, 2001], Real-coded Augmented Simulated Annealing was applied. Differential Evolution arose to solve the famous Chebychev polynomial problem [Storn, 1996],[Storn, WWW]. The last of the introduced methods is the so-called **SADE** technology. It is a simplified real-coded differential Genetic Algorithm that was developed as a specific recombination of a Genetic Algorithm and Differential Evolution intended to solve problems on high-dimensional domains represented by the *type 0* test function [Hrstka and Kučerová, 2000]. All these methods may aspire to be a universal optimization algorithm. So, detailed numerical tests of all these four optimization methods for the aforementioned optimization problems to examine their behavior and performance have been conducted.

The chapter is organized as follows. Section 3.2 provides brief description of each optimization task, while individual optimization algorithms are discussed in Section 3.3. Numerical results appear in Section 3.4. Summary of the performance of individual methods is presented in Section 3.5. For the sake of completeness, the parameter settings of algorithms are shown in Appendix C.

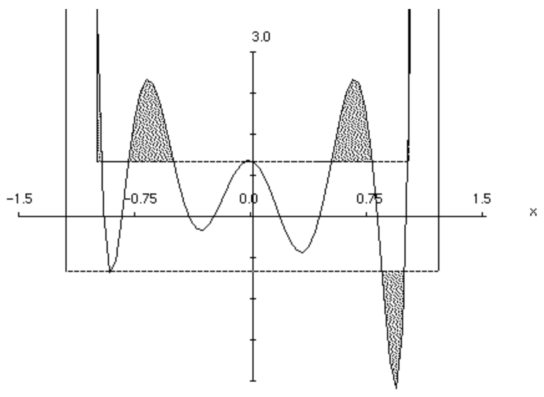


Figure 3.1: A graph of a Chebychev polynomial ($n = 8$).

### *3.2 Optimization tasks*

The optimization problems that are used as a set of test functions can be divided into two groups: the “test suite”, containing “artificial functions” and the “engineering problems”, which collect more (hopefully) practical optimization tasks. Specifically, these problems are :

**Test suite** containing the Chebychev trial polynomial problem and *Type 0* benchmark trial function and

**Engineering problems** consisting of the Reinforced concrete beam layout and the Periodic unit cell problem.

The following section provides description of selected functions in more detail.

#### *3.2.1 Chebychev problem*

The Chebychev trial polynomial problem is one of the most famous optimization problems. Our goal is to find such coefficients of a polynomial constrained by the condition that the graph of the polynomial can be fitted into a specified area (see Fig. 3.1).

Thus, the optimized values are the parameters $a_i$ of a polynomial expression:

$$f(x) = \sum_{i=0}^{n} a_i x^i, \tag{3.1}$$

and the value of objective function is determined as a sum of the areas, where the function graph exceeds a given boundary (hatched areas in Fig. 3.1).

#### *3.2.2* **Type 0** *function*

This trial optimization problem was proposed in [Hrstka and Kučerová, 2000] to examine the ability of the optimization method to find a single extreme of a function with a high number

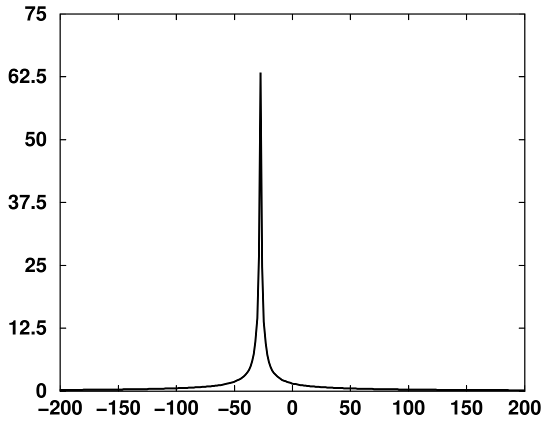


Figure 3.2: An example of a *type 0* function.

of parameters and growth of computational complexity with the problem dimension. For this reason, we used a function with a single extreme on the top of the high and narrow peak:

$$f(\mathbf{x}) = y_0 \left( \frac{\pi}{2} - \arctan \frac{\|\mathbf{x} - \mathbf{x_0}\|}{r_0} \right), \tag{3.2}$$

where $\mathbf{x}$ is a vector of unknown variables, $\mathbf{x_0}$ is the point of the global extreme (the top of the peak) and $y_0$ and $r_0$ are parameters that influence the height or the width of the peak, respectively. An example of such a function on one dimensional domain is shown in Fig. 3.2.

Although this example function has only a single extreme, to find it even with a moderate precision is a non-trivial task for several reasons. First, in the very neighborhood of the extreme the function is so steep that even a futile change of the coordinates causes a large change of the function value; in such a case it is very difficult for the algorithm to determine which way leads to the top. Second, the peak is located on a very narrow part of a domain and this disproportion increases very quickly with the problem dimension.

### *3.2.3 Reinforced concrete beam layout*

An effort to create an optimal design of a steel-reinforced concrete structure is as old as the material itself. In present times emphasis is put on this problem due to widespread use of RC structures in Civil Engineering. Frame structures are a major part in this field with beams playing an important role as one of the basic building blocks of this construction system. An objective is to choose the best design from all possible configurations that can create the requested structure – in our case a continuous beam (see Fig. 3.3).

The total cost of the structure is used as a comparison factor. An advantage of the financial rating is its natural meaning to non-experts and easiness of constraints implementation. In our particular case, the objective function reads

$$f(\mathbf{X}) = V_c P_c + W_s P_s + \sum p f_i \,, \tag{3.3}$$

where $V_c$ is the volume of concrete and $W_s$ is the weight of steel; $P_c$ and $P_s$ are the price of concrete per unit volume and steel per kilogram, respectively. From the mathematical point of view the penalty function $pf_i$ is the distance between a solution and the feasible space, or

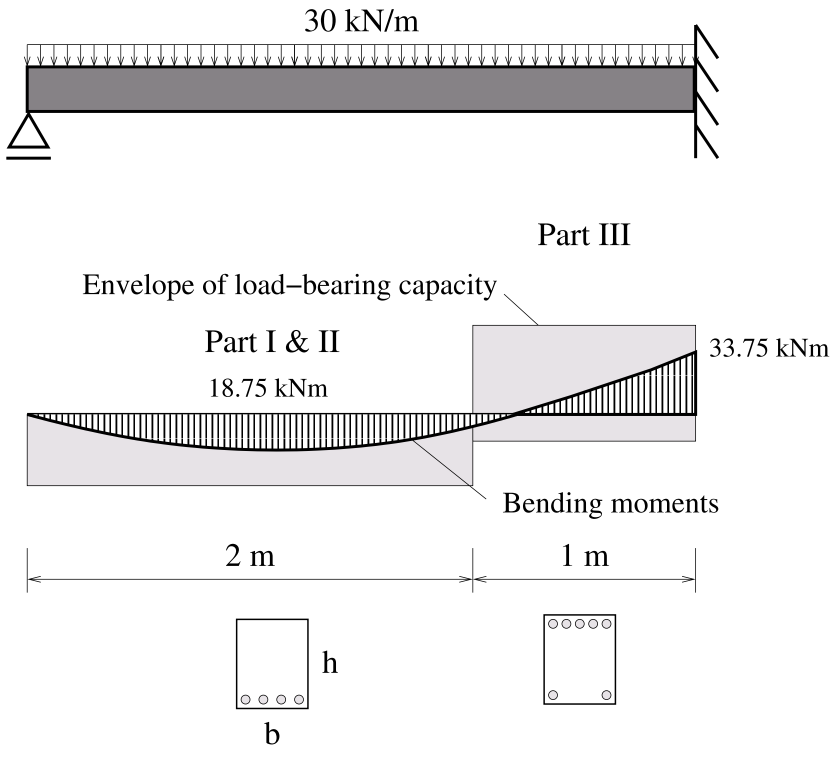


Figure 3.3: A continuous beam subjected to a uniform loading.

equivalently, a price spent on the fulfillment of the condition $i$. Suppose that a variable $\Phi_i$ should not exceed a certain allowable limit $\Phi_{i,max}$. Then, the penalty $p_{fi}$ assumes the form

$$p_{fi} = \begin{cases} 0 & \text{if } \Phi_i \leq \Phi_{i,max}, \\ w_i \cdot (\Phi_i/\Phi_{i,max})^2 & \text{otherwise}, \end{cases} \tag{3.4}$$

where $w_i$ is the weight of the $i$-th constraint.

The constraints in this procedure deal with allowable strength and serviceability limits given by a chosen standard – in our case EUROCODE 2 (EC2) [Eurocode 2, 1991]. An interested reader can find implementation details for example in [Lepš and Šejnoha, 2003].

Consider a rectangular cross-section of a beam. There is the width $b$ and the height $h$ to optimize. Other variables in $\mathbf{X}$ come from the model of a general RC beam which was presented in [Matouš et al., 2000] : the beam is divided into three elements between supports with the same diameter of longitudinal reinforcement along the top surface and another one along the bottom. The differences are only in numbers of steel reinforcement bars in particular elements. The shear reinforcement is designed alike. There are three shear-dimension parts - each of them with different spacing of stirrups but the same diameter in the whole element. This partitioning reflects the characteristic distribution of internal forces and moments in frame structures, where the extremal values usually occur at three points–at mid-span and two end joints. The novelty of our approach is the assumption that the length of parts may attain only the discrete values, in our case corresponding to 0.025 m precision. The same principle is used for the cross-section dimensions $b$ and $h$.

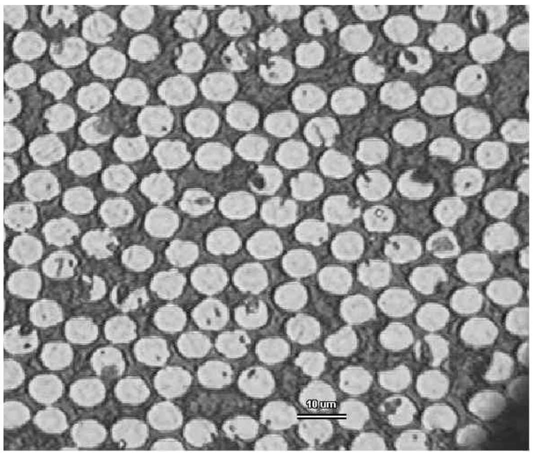


Figure 3.4: An example of a microstructure of a unidirectional fiber composite.

### *3.2.4 Periodic unit cell construction*

The motivation for this problem comes from the analysis of unidirectional fiber composite materials. Such materials consist of a large number of fibers (which serve as a reinforcement) embedded in the matrix phase. The goal is to determine the overall behavior of such a material system provided that material properties of the matrix and fibers are known. It turns out that for this prediction, the geometrical arrangement of fibers must be taken into account.

Unfortunately, the distribution of fibers in real composite materials is quite complex (see Fig. 3.4). To avoid such an obstacle, we attempt to replace a complicated microstructure with a large number of fibers by a certain *periodic unit cell,* which resembles the original material sample. More specifically, we describe the actual distribution of fibers by a suitable microstructural function and then determine the parameters of the periodic unit cell such that the difference between the function related to the periodic unit cell and function related to the original microstructure is minimized (for detailed discussion see [Zeman and Šejnoha, 2001]).

The microstructural function used in the present approach is the *second order intensity function* $K(r)$, which gives the number of further points expected to lie within a radial distance $r$ of an arbitrary point divided by the number of particles (fibers) per unit area ([Ripley, 1977])

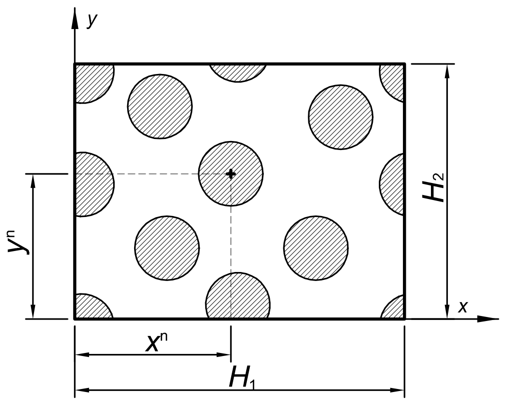
$$K(r) = \frac{A}{N^2} \sum_{k=1}^{N} I_k(r), \tag{3.5}$$

where $I_k(r)$ is the number of points within a circle with center at the particle $k$ and radius $r$, $N$ is the total number of particles (fibers) in the sample and $A$ is the sample area.

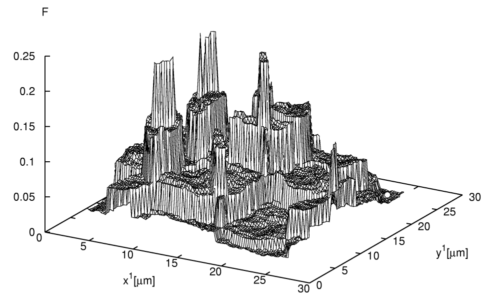
An objective function related to this descriptor can be defined as

$$F(\mathbf{x}^N, H_1, H_2) = \sum_{i=1}^{N_m} \left( \frac{K_0(r_i) - K(r_i)}{\pi r_i^2} \right)^2, \tag{3.6}$$

where vector $\mathbf{x}^N = \{x^1, y^1, \ldots, x^N, y^N\}$ stands for the position of particle centers of the periodic unit cell; $x^i$ and $y^i$ correspond to $x$ and $y$ coordinates of the $i$-th particle, $H_1$ and $H_2$ are the dimensions of the unit cell (see Fig. (3.5a)), $K_0(r_i)$ is the value of $K$ function corresponding to the original medium calculated in the point $r_i$ and $N_m$ is the number of points, in which both



(a)



(b)

Figure 3.5: (a) Geometry of a periodic unit cell, (b) An example of the objective function.

functions are evaluated. Throughout this study, we assume square periodic unit cell ($H_1 = H_2$) and determine its dimensions in such a way that the volume fraction of the fiber phase in the periodic unit cell is the same as in the original micrograph. An example of the objective function is shown in Fig. 3.5(b).

## *3.3 Applied methods*

During last few years, we have developed and tested several Evolutionary Algorithms that are based on both binary/integer and real-valued representation of searched variables. Each of them has been primarily applied to one particular optimization problem of the four introduced above. These methods are (in order of appearance):

- Differential Evolution (**DE**), developed by R. Storn and K. Price [Storn and Price, 1995] to solve the Chebychev trial polynomial problem.
- Simplified Atavistic Differential Evolution (**SADE**), developed by the authors for research on high-dimensional problems [Hrstka and Kučerová, 2000],[Hrstka, WWWb].
- Integer Augmented Simulated Annealing (**IASA**), a combination of an integer-coded Genetic Algorithm and Simulated Annealing; it was primarily applied to the reinforced concrete beam layout optimization problem.
- Real-coded Augmented Simulated Annealing (**RASA**), a combination of a real-coded Genetic Algorithm by Michalewicz [Michalewicz et al., 1994] and Simulated Annealing; it was developed for solving the periodic unit cell problem.

### *3.3.1 Differential Evolution (*DE*)*

Differential Evolution was invented as the solution method for the Chebychev trial polynomial problem by R. Storn and K. Price [Storn, 1996]. It operates directly on real valued chromosomes and uses the so-called differential operator, which works with real numbers in natural manner and fulfills the same purpose as the cross-over operator in a Simple Genetic Algorithm.

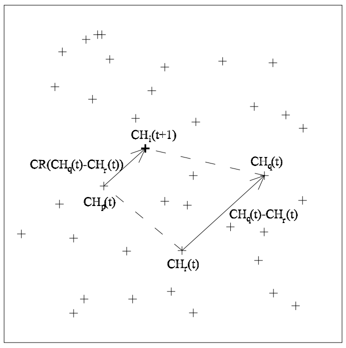


Figure 3.6: The geometric meaning of a certain subtype of the differential operator.

The differential operator has a sequential character: Let $CH_i(t)$ be the *i*-th chromosome of generation *t*

$$CH_i(t) = (ch_{i1}(t), ch_{i2}(t), ..., ch_{in}(t)), \tag{3.7}$$

where $n$ is the chromosome length (which means the number of variables of the fitness function in the real encoded case). Next, let $\Lambda$ be a subset[1] of $\{1, 2, ..., n\}$. Then for each $j \in \Lambda$ holds

$$\begin{aligned} ch_{ij}(t+1) = ch_{ij}(t) \quad &+ \quad F_1\left(ch_{pj}(t) - ch_{qj}(t)\right) \\ &+ \quad F_2\left(ch_{\mathrm{best}j}(t) - ch_{ij}(t)\right), \end{aligned} \tag{3.8}$$

and for each $j \notin \Lambda$ we get

$$ch_{ij}(t+1) = ch_{ij}(t), \tag{3.9}$$

where $ch_{pj}$ and $ch_{qj}$ are the *j*-th coordinates of two randomly chosen chromosomes and $ch_{\mathrm{best}j}$ is the *j*-th coordinate of the best chromosome in generation *t*. $F_1$ and $F_2$ are then coefficients usually taken from interval $(0, 1)$. Fig. 3.6 shows the geometrical meaning of this operator.

The method can be understood as a stand-alone evolutionary method or it can be taken as a special case of a Genetic Algorithm. The algorithmic scheme is similar to the Genetic Algorithms but it is much simpler:

1. At the beginning an initial population is randomly created and the fitness function value is assigned to each individual.

2. For each chromosome in the population, its possible replacement is created using the differential operator as described above.

3. Each chromosome in the population has to be compared with its possible replacement and if an improvement occurs, it is replaced.

4. Steps 2 and 3 are repeated until some stopping criterion is reached.

As it can be seen, there are certain features that distinguish this method from the Simple Genetic Algorithm, namely:

[1] The determination of $\Lambda$ is influenced by the parameter called *crossrate* ($CR$), see [Storn, 1996].

- the crossing-over is performed by applying the differential operator (3.8), (3.9),
- the selection operation like the roulette wheel, for example, is not performed, the individuals that are going to be affected by the differential operator, are chosen purely randomly,
- selection of individuals to survive is simplified to the mentioned fashion: each chromosome has its possible replacement and if an improvement occurs, it is replaced,
- the mutation operator is not introduced as the authors of **DE** claim that the differential operator is able to replace both mutation and uniform crossover known from basic **GA**s.

Further details together with the source codes of **DE** can be obtained from the web page [Storn, WWW].

And finally, in a sketchy form presented in Section 2.2.1, the algorithm can be written as

$$opt_{DE}({}^{t+1}I^{\mu}) = \mathbf{sel}^{\mu}_{\mu+1}({}^{t}I, \mathbf{rec}^{1}_{4}({}^{t}I, {}^{t}I_{best})) \tag{3.10}$$

and can be also understood as a nice example of a $(\mu+1)$-optimization algorithm.

### *3.3.2 Simplified Atavistic Differential Evolution (*SADE*)*

This method was proposed as an adaptation of the differential evolution in order to acquire an algorithm which will be able to solve optimization problems on real domains with a high number of variables. This algorithm combines features of Differential Evolution with traditional Genetic Algorithms. It uses the differential operator in the simplified form and an algorithmic scheme similar to Evolution Strategies.

The differential operator has been taken from the Differential Evolution in a simplified version for the same purpose as the cross-over is used in a Simple Genetic Algorithm. This operator has the following fashion: Let (again) $CH_i(t)$ be the *i*-th chromosome in generation *t*,

$$CH_i(t) = (ch_{i1}(t), ch_{i2}(t), ..., ch_{in}(t)), \tag{3.11}$$

where $n$ is the number of variables of the fitness function. Then, the simplified differential operator can be written as

$$ch_{ij}(t+1) = ch_{pj}(t) + CR\left(ch_{qj}(t) - ch_{rj}(t)\right), \tag{3.12}$$

where $ch_{pj}$, $ch_{qj}$ and $ch_{rj}$ are the *j*-th coordinates of three randomly chosen chromosomes and $CR$ is the so-called *cross-rate*. Due to its simplicity this operator can be rewritten also in the vector form:

$$CH_i(t+1) = CH_p(t) + CR(CH_q(t) - CH_r(t)). \tag{3.13}$$

Contrary to the Differential Evolution, this method uses an algorithmic scheme very similar to Evolution Strategies:

1. As the first step, the initial population is generated randomly and the fitness function value is assigned to all chromosomes in the population.

2. Several new chromosomes are created using the mutation operators - the mutation and the local mutation (number of them depends on a value a of parameter called *radioactivity*, which gives the mutation probability).

3. Other new chromosomes are created using the simplified differential operator as was described above; the whole amount of chromosomes in the population doubles.

4. The fitness function values are assigned to all the newly created chromosomes.

5. The selection operator is applied to the double-sized population, so the amount of individuals is decreased to its original value.

6. Steps 2-5 are repeated until some stopping criterion is reached.

And again, using the introduced notation, the **SADE** method can be easily written as

$$opt_{SADE}(^{t+1}I^{\mu}) = \mathbf{sel}^{\mu}_{2\mu}(\mathbf{rec}^{1}_{3}(^{t}I), \mathbf{mut}^{1}_{1}(^{t}I)) \qquad (3.14)$$

and can be characterized as a $(\mu+\mu)$-optimization algorithm.

Next, we introduce the description of the mentioned operators in detail:

**Mutation:** If a certain chromosome $CH_i(t)$ is chosen to be mutated, a new random chromosome $RP$ is generated and the mutated one $CH_k(t+1)$ is computed using the following relation:

$$CH_k(t+1) = CH_i(t) + MR(RP - CH_i(t)), \qquad (3.15)$$

where $MR$ is a parameter called *mutation-rate*.

**Local mutation:** If a certain chromosome is chosen to be locally mutated, all its coordinates have to be altered by a random value from a given (usually very small) range.

**Selection:** This method uses modified tournament strategy to reduce the population size: two chromosomes are randomly chosen, compared and the worse of them is rejected, so the population size is decreased by $1$; this step is repeated until the population size reaches its original size[2].

Detailed description of the **SADE** method including source codes in C/C++ and the tests documentation for the high-dimensional problems can be obtained directly on the web-page [Hrstka, WWWb] and also from the article [Hrstka and Kučerová, 2000] .

[2] Contrary to the traditional tournament strategy this approach can ensure that the best chromosome will not be lost even if it was not chosen to any tournament.

### *3.3.3 Real-valued Augmented Simulated Annealing (*RASA*)*

The Augmented Simulated Annealing method is the combination of two stochastic optimization techniques – a Genetic Algorithm and Simulated Annealing. It uses basic principles of Genetic Algorithms ( selection, recombination by genetic operators ), but controls replacement of parents by the Metropolis criterion (see Eq. (3.17)). This increases the robustness of the method, since we allow a worse child to replace its parent and thus escape from local minima, which is in contrary with **DE** methods described in Section 3.3.1.

The algorithmic scheme of the present implementation is summarized as follows.

1. Randomly generate an initial population and assign fitness to each individual. The initial temperature is set to $T_0 = T_{max} = \texttt{T\_frac} F_{avg}$ and the minimal temperature is determined as $T_{min} = \texttt{T\_frac\_min} F_{avg}$ , where $F_{avg}$ is the average fitness value of the initial population.

2. Select an appropriate operator. Each operator is assigned a certain probability of selection.

3. Select an appropriate number of individuals (according to the operator) and generate possible replacements. To select individuals, we apply *normalized geometric ranking* scheme ([Houck et al., 1995]): The probability of selection of the $i$-th individual is given by

$$p_i = q'(1 - \mathtt{q})^{r-1}, \qquad q' = \frac{\mathtt{q}}{1 - (1 - \mathtt{q})^{\mathtt{pop\_size}}}, \tag{3.16}$$

   where `q` is the probability of selecting the best individual in the population, $r$ is the rank of the $i$-th individual with respect to its fitness, and `pop_size` is the population size.

4. Apply operators to selected parent(s) to obtain possible replacement(s).

4a. Look for an individual identical to possible replacement(s) in the population. If such individual(s) exists, no replacement is performed.

4b. Replace an old individual if

$$u(0,1) \leq e^{(F(I_{\rm old}) - F(I_{\rm new}))/T_t}, \tag{3.17}$$

   where $F(\cdot)$ is fitness of a given individual, $T_t$ is the actual temperature and $u(\cdot,\cdot)$ is a random number with the uniform distribution on a given interval.

5. Steps 2–4 are performed until the number of successfully accepted individuals reaches `success_max` or selected number of steps reaches `counter_max`.

6. Decrease temperature

$$T_{t+1} = \mathtt{T\_mult} T_t. \tag{3.18}$$

   If actual temperature $T_{t+1}$ is smaller than $T_{min}$, perform *reannealing* – i.e. perform step #1 for one half of the population.

7. Steps 2–6 are repeated until the termination condition is attained.

This algorithm can be therefore seen also as a $(\mu+\mu)$-Evolutionary Algorithm and its simplified notation can be formulated as

$$opt_{RASA}({}^{t+1}I^{\mu}) = \mathbf{sel}^{\mu}_{2\mu}({}^{t}I, \mathbf{rec}({}^{t}I)||\mathbf{mut}({}^{t}I)) \ . \tag{3.19}$$

**List of operators** The set of real-valued operators proposed in [Michalewicz et al., 1994] was implemented as follows. In the sequel, we will denote $L$ and $U$ as vectors of lower/upper bounds on unknown variables, $u(a, b)$ and $u[a, b]$ as a real or integer random variable with the uniform distribution on a closed interval $\langle a, b\rangle$. Otherwise we use the same notation as employed in the previous sections.

**Uniform mutation:** Let $k = [1, n]$

$$ch_{ij}(t+1) = \begin{cases} u(L_j, U_j), & \text{if } j = k \\ ch_{ij}(t), & \text{otherwise,} \end{cases} \tag{3.20}$$

**Boundary mutation:** Let $k = u[1, n]$, $p = u(0, 1)$ and set:

$$ch_{ij}(t+1) = \begin{cases} L_j, & \text{if } j = k, p < .5 \\ U_j, & \text{if } j = k, p \geq .5 \\ ch_{ij}(t), & \text{otherwise} \end{cases} \tag{3.21}$$

**Non-uniform mutation:** Let $k = [1, n]$, $p = u(0, 1)$ and set:

$$ch_{ij}(t+1) = \begin{cases} ch_{ij}(t) + (L_j - ch_{ij}(t))f, & \text{if } j = k, p < .5 \\ ch_{ij}(t) + (U_j - ch_{ij}(t))f, & \text{if } j = k, p \geq .5 \\ ch_{ij}(t), & \text{otherwise} \end{cases} \tag{3.22}$$

where $f = u(0, 1)(T_t/T_0)^{\mathsf{b}}$ and $\mathsf{b}$ is the shape parameter.

**Multi-non-uniform mutation:** Apply non-uniform mutation to all variables of $CH_i$.

**Simple crossover:** Let $k = [1, n]$ and set:

$$ch_{il}(t+1) = \begin{cases} ch_{il}(t), & \text{if } l < k \\ ch_{jl}(t), & \text{otherwise} \end{cases} \qquad ch_{jl}(t+1) = \begin{cases} ch_{jl}(t), & \text{if } l < k \\ ch_{il}(t), & \text{otherwise} \end{cases}$$

**Simple arithmetic crossover:** Let $k = u[1, n]$, $p = u(0, 1)$ and set:

$$ch_{il}(t+1) = \begin{cases} pch_{il}(t) + (1-p)ch_{jl}(t), & \text{if } l = k \\ ch_{il}(t), & \text{otherwise} \end{cases} \tag{3.23}$$

$$ch_{jl}(t+1) = \begin{cases} pch_{jl}(t) + (1-p)ch_{il}(t), & \text{if } l = k \\ ch_{jl}(t), & \text{otherwise} \end{cases} \tag{3.24}$$

**Whole arithmetic crossover:** Simple arithmetic crossover applied to all variables of $CH_i$ and $CH_j$.

**Heuristic crossover:** Let $p = u(0, 1)$, $j = [1, n]$ and $k = [1, n]$ such that $j \neq k$ and set:

$$CH_i(t+1) = CH_i(t) + p(CH_j(t) - CH_k(t)). \tag{3.25}$$

If $CH_i(t+1)$ is not feasible then a new random number $p$ is generated until the feasibility condition is met or the maximum number of heuristic crossover applications `num_heu_max` is exceeded.

### *3.3.4 Integer Augmented Simulated Annealing (*IASA*)*

The Integer Augmented Simulated Annealing method is based on the same ideas as the previously mentioned **RASA** algorithm. This procedure effectively exploits the essentials of **GA**s (a population of chromosomes, rather then a single point in space is optimized) together with the basic concept of a Simulated Annealing method guiding the search towards minimal energy states. To avoid well-known problems with binary coding the integer coding was used, see Section 2.4.1. Together with new operators such as differential crossover and a new mutation operator encouraging results were obtained.

The description of the algorithm does not substantially differ from the **RASA** algorithm, but for the sake of completeness all steps are briefly reviewed here.

1. The initial population consisting of `OldSize` individuals is created randomly and fitnesses are assigned to each individual. Starting and ending temperatures `T_min` and `T_max` are set by the user.

2. If a real random number $p = u(0,1)$ is smaller than `CrossoverProb` the crossover is used, otherwise the mutation is applied. This step is repeated until the number `NewSize` of new solutions is obtained.

3. For each individual in a "new" population one "parent" from "old" part is selected. The "old" solution is replaced if
$$u(0,1) \leq \frac{1}{1+e^{(F(I_{\text{old}})-F(I_{\text{new}}))/T_t}}\ , \tag{3.26}$$
where $F(\cdot)$, $T_t$ and $u(\cdot,\cdot)$ have the same meaning as in the previous section. Equation (3.26) ensures the 50% probability of survival when comparing two solutions with same fitness.

4. Steps 2–3 are performed until the number of successfully accepted individuals reaches `SuccessMax` or the selected number of steps reaches `CounterMax`.

5. The actual temperature is decreased by
$$T_{t+1} = T_t \left(\frac{\texttt{T\_min}}{\texttt{T\_max}}\right)^{\left(\frac{\texttt{CounterMax}}{\texttt{TminAtCallsRate} * \texttt{MaxCalls}}\right)}\ , \tag{3.27}$$
where `TminAtCallsRate` determines a fraction of the maximum allowable number of function calls `MaxCalls` in which the minimum temperature `T_min` will occur. The *reannealing* step is represented here by setting actual temperature $T_{t+1}$ equal to `T_max`.

6. Steps 2–5 are repeated until the termination condition is attained.

This algorithm can be added to the group of $(\mu{+}\lambda)$-Evolutionary Algorithms and its notation can be formulated as
$$opt_{IASA}({}^{t+1}I^{\mu}) = \mathbf{sel}^{\mu}_{\mu+\lambda}({}^{t}I, \mathbf{rec}^{1}_{3}({}^{t}I), \mathbf{mut}^{1}_{2}({}^{t}I)) \tag{3.28}$$
and in connection with the operators presented in previous sections, integer operators within the **IASA** algorithm have the following form:

**Differential crossover:** This operator is inspired by the **DE**. A new individual $CH_j(t)$ is created from three randomly selected solutions $CH_p(t)$, $CH_q(t)$ and $CH_r(t)$ according to

$$CH_j(t+1) = CH_p(t) + u(0.0, CR)(CH_q(t) - CH_r(t)). \tag{3.29}$$

Note that all vectors $CH_i$ are integer numbers and also that the influence of the difference on the right-hand side randomly varies between zero and *cross-rate* $CR$.

**Mutation:** The mutation operator is provided by modifying each variable in $CH_j(t)$ to

$$ch_{ij}(t+1) = ch_{ij}(t) + N(0, \frac{\mid ch_{ij}(t) - ch_{pj}(t) \mid}{2} + 1)\ , \tag{3.30}$$

where $N(\cdot,\cdot)$ is a random integer number derived from the Gauss normal distribution and $ch_{pj}(t)$ is the $j$-th variable of a randomly selected vector $CH_p(t)$.

## *3.4 Test computations and results*

Each of the methods introduced in the previous section has been tested on all of the presented optimization problems. The methodology that has been used for our computations is based on the following criteria:

- For each problem and each method the computation was run 100 times to avoid an influence of random circumstances.
- For all cases, the number of successful runs (which can be traded as the probability of success or the reliability of the method) is presented.
- If the number of successful runs is non-zero, the average number of fitness calls of all successful runs is also presented.

Further details of individual function settings and methodology for results evaluation can be found in the next subsections.

### *3.4.1 Results for the Chebychev problem*

| **Method** | **IASA** | **RASA** | **DE** | **SADE** |
|---|---|---|---|---|
| Successful runs | 100 | 100 | 100 | 100 |
| Average number of fitness calls | 10342 | 47151 | 25910 | 24016 |

Table 3.1: Results for the Chebychev polynomial problem.

The computations were performed for the Chebychev problem with a degree of the polynomial expression $n = 8$ (the T8 problem), which corresponds to the nine dimensions of the problem. The computation was terminated if the algorithm reached a value of the objective function smaller then $10^{-5}$ or the number of function evaluations exceeded $100,000$. Upper bounds on individual coefficients were set to $512$, while lower bounds were equal to $-512$. The results of individual algorithms are shown in Table 3.1 and Figure 3.8.

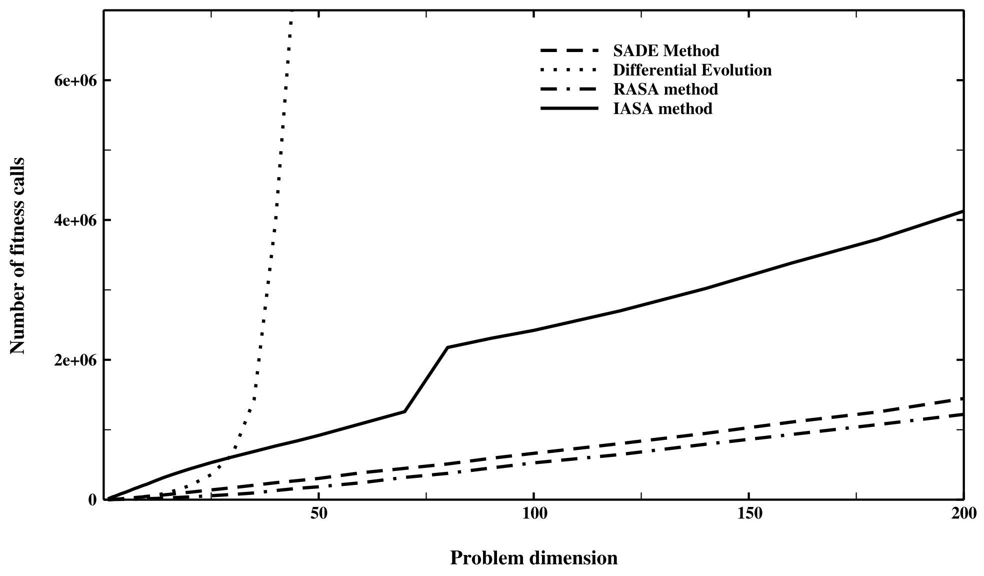


Figure 3.7: A comparison of results for the *type 0* function.

### *3.4.2 Results for the* **type 0** *trial function*

Test computations for the *type 0* problem were performed for a wide set of problem dimensions, ranging from $1$ to $200$. The upper bound on each variable was set to $400$, while the lower bound value was $-400$. For each run, the position of the extreme was randomly generated within these bounds and the height of the peak $y_0$ was generated from the range $0$–$50$. The parameter $r_0$ was set to $1$. The computation was terminated when the value of the objective function was found with a precision greater than $10^{-3}$. The results are given in the form of the growth of computational complexity with respect to the problem dimension. For each dimension, the computation was run 100 times and the average number of fitness calls was recorded (see Fig. 3.7 and Table 3.2).

| Problem dimension | IASA | RASA | DE | SADE |
|---|---|---|---|---|
| 10 | 246,120 | 13,113 | 39,340 | 46,956 |
| 30 | 611,760 | 74,375 | 653,600 | 171,539 |
| 50 | 926,100 | 183,882 | N/A | 304,327 |
| 100 | 2,284,590 | 526,492 | N/A | 663,084 |
| 140 | 3,192,800 | 793,036 | N/A | 948,197 |
| 200 | 4,184,200 | 1,220,513 | N/A | 1,446,540 |

Table 3.2: Average number of fitness calls for the type-$0$ function

### *3.4.3 Results for the reinforced concrete beam layout problem*

The basic parameters subjected to optimization were the beam width $b$, which was assumed to take discrete values between $0.15$ m and $0.45$ m with the step $0.025$ m and the beam height $h$ ranging from $0.15$ m to $0.85$ m with the step $0.025$ m. For each of the three parts of a beam, the diameter and the number of longitudinal reinforcing bars located at the bottom and the top of the beam, spacing and the diameter of stirrups and the length of the corresponding part were optimized. Lower bounds were selected for the sake of structural requirements; solutions exceeding upper bounds were considered to be irrelevant for the studied examples. However, from the optimization point of view, bounds can be easily adjusted to any reasonable value.The number of longitudinal bars was restricted to the range $0$–$15$, the spacing of stirrups was assumed to vary from $0.05$ m to $0.40$ m with the $0.025$ m step. The profiles of longitudinal bars were drawn from the list of $16$ entries while for the stirrups, only $4$ diameters were considered. This finally results in $18$ independent variables. Note that the maximal number of longitudinal bars presents only the upper bound on the searched variable; the specific restrictions given by Codes of Practice are directly incorporated in the objective function. For more details see [Lepš and Šejnoha, 2000, Matouš et al., 2000]. The computation was terminated if an individual with a price smaller than $573.5$ CZK was found or the number of objective function calls exceeded $1,000,000$. Table 3.3 stores the obtained results of different optimization algorithms, see also Figure 3.8.

| **Method** | **IASA** | **RASA** | **DE** | **SADE** |
|---|---:|---:|---:|---:|
| Successful runs | 100 | 100 | 100 | 100 |
| Average number of fitness calls | 108732 | 131495 | 196451 | 185819 |

Table 3.3: Results for the reinforced concrete beam layout

### *3.4.4 Results for the periodic unit cell problem*

Test computations for the periodic unit cell construction were performed for the $10$-fiber unit cell (i.e. the dimension of the problem was $20$). The computation was terminated if the algorithm returned value smaller than $6 \times 10^{-5}$ or the number of function calls exceeded $400,000$. Variables were constrained to the box $0 \leq x_i \leq H_1 \approx 25.8$ (see Section (2.4)) . The required numbers of function are stored in Table 3.4 and displayed in Figure 3.8.

| **Method** | **IASA** | **RASA** | **DE** | **SADE** |
|---|---:|---:|---:|---:|
| Successful runs | 100 | 100 | 100 | 100 |
| Average number of fitness calls | 13641 | 12919 | 93464 | 55262 |

Table 3.4: Results for the periodic unit cell problem

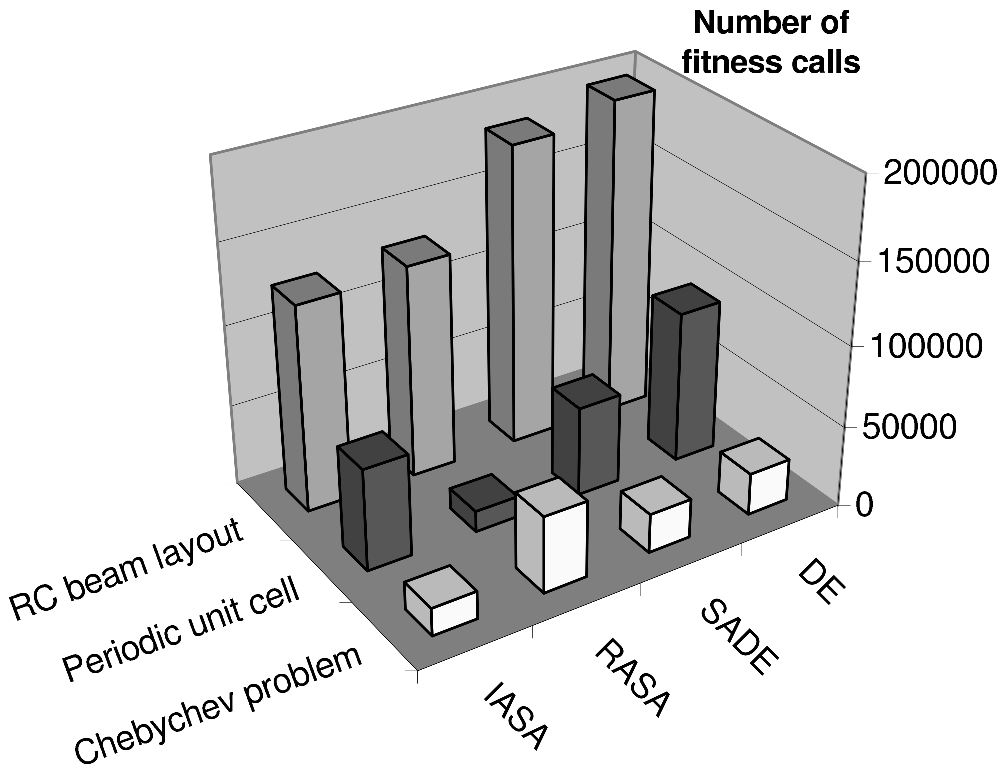


Figure 3.8: A comparison of results for Chebychev polynomial, reinforced concrete beam layout and periodic unit cell problems.

## *3.5 Conclusions for comparison*

**Differential Evolution** The Differential Evolution algorithm showed to be very efficient and robust for moderate-sized problems, but its performance for higher dimensions deteriorated. Moreover, the small number of parameters, see Table C.2, is another advantage of this method[3]. However, the results suggest that the absence of mutation-type operator(s) is a weak point of the algorithm.

**Simplified Atavistic Differential Evolution** The **SADE** algorithm was able to solve all problems of our test set with high reliability and speed. Although it needed a larger number of function calls than other methods (see Table 3.5), the differences are only marginal and do not present any serious disadvantage. Another attractive feature of this method is the relatively small number of parameters, see Table C.3.

**Real-valued Augmented Simulated Annealing** The **RASA** algorithm was successful for all presented problems; the average number of function calls was comparable to the other methods.

[3] This is the base-stone of the popularity of this method and it is probably a reason why the **DE** was modified also for multi-objective problems, see [Kukkonen and Lampinen, 2004].

The obvious disadvantage of this algorithm is a large number of parameters (Table C.1), which can result in a tedious tuning procedure. On the other hand, as follows from Appendix C, only two types of parameter settings were necessary – one for the continuous and one for the discrete functions.

**Integer Augmented Simulated Annealing** The **IASA** algorithm was the most successful and fastest method on problems with small dimensions. But on the problems with larger dimensions and with a higher number of local minima, the algorithm suffers from premature convergence and limited precision due to integer coding of variables. In addition, initial tuning of individual parameters, see Table C.4, presents another drawback of this method.

| **Method** | IASA | RASA | DE | SADE |
|---|---|---|---|---|
| **Chebychev problem** | 1 | 4 | 3 | 2 |
| **Type 0 test function** | 3 | 1 | $4^*$ | 2 |
| **Concrete beam layout** | 1 | 2 | 4 | 3 |
| **Periodic unit cell** | 3 | 1 | 4 | 2 |
| $\Sigma$ | 8 | 8 | 14 | 9 |

Table 3.5: Overall performance of methods. ($^*$ : Not successful for all runs)

The summary results are given in Table 3.5 to quantify the overall performance of the individual methods. Each of the method is ranked primarily with respect to its success rate and secondary with respect to the average number of fitness calls. The sum then reveals the overall performance of the method.

**Final comments** In our opinion, several interesting conclusions and suggestions can be made from the presented results. Each of them is discussed in more detail.

- The performance and robustness of the **SADE** method was distinguishly better than for the **DE** algorithm. This supports an important role of a mutation operator(s) in the optimization process.

- Although algorithms were developed independently, all use some form of differential operator. This shows the remarkable performance of this operator for both real-valued and discrete optimization problems.

- The most successful methods, the **SADE** and **RASA** algorithms, both employ a variant of "local mutation". This operator seems to be extremely important for higher-dimensional *type-0* functions, where these methods clearly outperform the others.

- Slightly better results of the **RASA** method can be most probably attributed to the re-annealing/restarting phase of the algorithm (a trivial but efficient tool for dealing with local minima) and to the search for an identical individual. The procedure for local minima assessment was implemented to the **SADE** method (see [Hrstka and Kučerová, 2004, Hrstka, WWWa] for results), incorporation into the **IASA** algorithm is under development.

- When comparing methods based on the discrete coding of variables with real-encoded ones it becomes clear that for continuous functions the methods with the real coding perform better. Nevertheless, after implementing new features, like those mentioned before, the performance is expected to be similar. On the other hand, the advantage of the **IASA** algorithm is the possibility of its use for discrete combinatorial problems like the Traveling salesman problem.

Therefore, from the practical point of view, the **SADE** method seems to be the most flexible alternative due to its simplicity and small number of parameters.

Chapter 4

# DESIGN OF REINFORCED CONCRETE FRAMES

For it is unworthy of excellent men to lose hours like slaves in the labour of calculation which would safely be relegated to anyone else if machines were used.

Gottfried Wilhelm Von Leibniz

Previous Chapter 3 deals mainly with problems that can be understood as black-box functions, even though for the periodic unit cell problem some knowledge can be added into an optimization process and the performance of optimization can be significantly improved, see e.g. [Matouš et al., 2000]. A humble goal of this chapter is to show that adding more information into the system of a design can enable us to optimize such type of structures which was not manageable in previous decades.

## *4.1 Introduction*

From the global point of view and following the classification introduced in Section 1.2, every design of a frame structure can be seen as a simultaneous topology, sizing and shape optimization problem. The designer can vary several possible topologies of the desired structure with different cross-sectional shapes, which can be selected from a predefined list of available profiles. Still our current knowledge and computer equipment do not enable us to solve this specific task in its full generality. On the other hand, the width, the height of a building and even the position of columns are given a long time before the final design starts. Therefore the topology optimization is not studied within this work and attention will be only paid to the shape and/or sizing tasks.

Research within the design of steel structures seems to be almost complete, see e.g. the works by D. Greiner [Greiner et al., 2001] and [Greiner et al., 2003]. On the other hand, the actual state of the art in the design of reinforced concrete (RC) structures is not so clear. The main reason is that steel structures design requirements typically belong to the pure sizing area, while reinforced concrete structures are on a half-way to shape optimization problems. Steel structures are characterized by a small set of available, usually standardized, cross-sections, on the other side, combinations of reinforcement within almost arbitrary shapes of a concrete structure lead to an enormous number of possible solutions. The objective of this chapter is therefore to fill in several gaps within the research of the design of reinforced concrete framed structures.

At this point a step-by-step description for the rest of an optimization-design process of frame structures will be presented and several possible solutions will be proposed. Logically, the first variable studied is the **shape** of members that the frame structure is composed of[1]. The shapes will be most likely selected from a small list of available sections limited by form-work abilities and moreover the economical aspect will usually lead to mostly uniform forms within one building. This reduction of the searched space therefore enables this part of design to be rationally tackled by the size optimization methods.

To continue the above mentioned ideas, let us assume that the shape is fixed. Then, the second and possibly the most challenging task is the **placement of reinforcing bars** within concrete members, often called **detailing**. And again, from the optimization point of view, this task generally belongs to the field of topology optimization, where the number of bars, their shape and material and even their mutual space position are searched for. The type and form of a chosen parameterization of the shape will determine the computational complexity of the problem in question. Although this solution is the most straightforward one in terms of both analysis and the design phase, it is obvious that this approach is unmanageable with the current computational resources and has to be solved by simpler methods[2].

Therefore the usual scenario is to divide the reinforcement design into the shear and bending parts. The optimization of shear reinforcement is not a difficult task especially if we limit the use of reinforcement to stirrups only. For comparison of different standard shear-design methodologies see [ASCE-ACI, 1998]. Also several papers dealing with optimization of shear reinforcement can be found, see e.g. the contribution [Lepš and Šejnoha, 2003]. Hence the main effort will be put on the design of longitudinal, also called flexural, reinforcement. From the design point of view, the most important spot is the most critical cross-section within each structural member. Therefore this task is reduced to the mere detailing of the limited number of cross-sections and as such is often solved by many researchers.

### *4.1.1 Design of reinforced concrete cross-sections*

It is probably a historical consequence that many researchers who have been solving the design of a RC cross-section by optimization methods have followed procedures defined in the appropriate Codes of Practice, see e.g. [Hadi, 2001] and [Coello et al., 1997]. The uniting characteristic of this approach is the use of a steel area instead of individual reinforcing bars. As a consequence, this methodology is adequate for analysis but not for practical use. This is nicely formulated in [Koumousis et al., 1996]:

> In practice, there are many cases where the design is excellent from the analysis point of view, but its constructibility is poor or impossible.

Even though these approaches do not seem to be appropriate for implementation into practice, they can lead to new designing and optimization methods for different problems of research, see e.g. [Rizzo et al., 2000] and [Chen et al., 2001], or can serve as benchmarks for optimization algorithms [Coello et al., 1997].

Therefore, the methodology presented in the paper [Choi and Kwak, 1990] (and a very similar procedure [Rafiq and Southcombe, 1998]), called the *declarative approach*, looks like

[1] To propose general methodology all possible shapes are considered, the only limiting condition being the availability of their parameterization.

[2] Nevertheless, a similar methodology can be found in the so-called *strut and tie* approaches usually used for non-linear analysis of reinforced concrete structures, see e.g. [Abdallaa et al., 1996] and [ASCE-ACI, 1998].

a breakthrough into the usual system of a design. Firstly, all possibilities of designing a cross-section are defined to meet structural requirements like steel cover or mutual distances between reinforcing bars, and after that the optimal solutions are selected or searched. This can be done by the brute force method or by any available optimization algorithms presented earlier. Hence this approach ensures not only the fulfillment of the load-bearing conditions but also the constructibility of the structure. Also the automation of this procedure by computers can be efficiently used.

### *4.1.2 Design of frame members*

If we follow the above mentioned ideas, the problem of combining the designs of individual cross-sections must be now solved along with the search for the parameters of cross-sectional shapes. All these variables come from a discrete domain and therefore some combinatorial algorithm, like presented **EA**s, can be used. Hence the declarative approach can move the design of the RC frame structure from the topology or mixed continuous-discrete to the pure sizing optimization, which can be relatively easily managed by the available computational resources. One particular and very promising example of the declarative approach can be found in [Koumousis et al., 1996], where the database of possible cross-sectional designs was combined with logical programming to produce a practical design of a spatial frame structure up to a detailed drawing in a CAD program.

## *4.2 Proposed design procedure*

In this work, the multi-objective version of a hybrid optimization procedure is applied to the design and consequent detailing of structural members within steel-reinforced concrete frames. The implemented strategy relies on splitting this difficult optimization task into two parts. Firstly, the detailing of a reinforced concrete cross-section is solved by the “brute force” method using an efficient procedure for fast evaluation of internal forces for a polygonal cross-section and an arbitrary stress-strain relationship. Secondly, the optimization of a whole structure in terms of basic structural characteristics, e.g. types of materials, dimensions or profiles, is tackled with a multi-objective stochastic optimization method. For this purpose, two important objectives are selected and defined - the total price of a resulting structure and the maximum deflection of structural members. As a result of the presented research, the Pareto-optimal solutions can be plotted to demonstrate the non-linearity of this design problem and also to show the applicability of this approach in Civil Engineering practice.

### *4.2.1 Design parameterization*

As already mentioned above, we search for a frame structure simultaneously considering price of the structure and maximum deflection as the objectives of optimization. For simplicity, we limit our attention to 2D problems and elements with rectangular cross-sections. Hence, we consider frame structures located in the $xz$ plane and our interest is restricted to internal forces acting in this plane: the bending moment $M_y$, the normal force $N_x$ and the shear force $V_z$.

From the construction point of view as well as optimization itself it appears to be advantageous to decompose the whole structure into $n_d$ *design elements* (see Fig. 4.1). These user-defined blocks are parts of a structure which a-priori possess identical optimized parameters

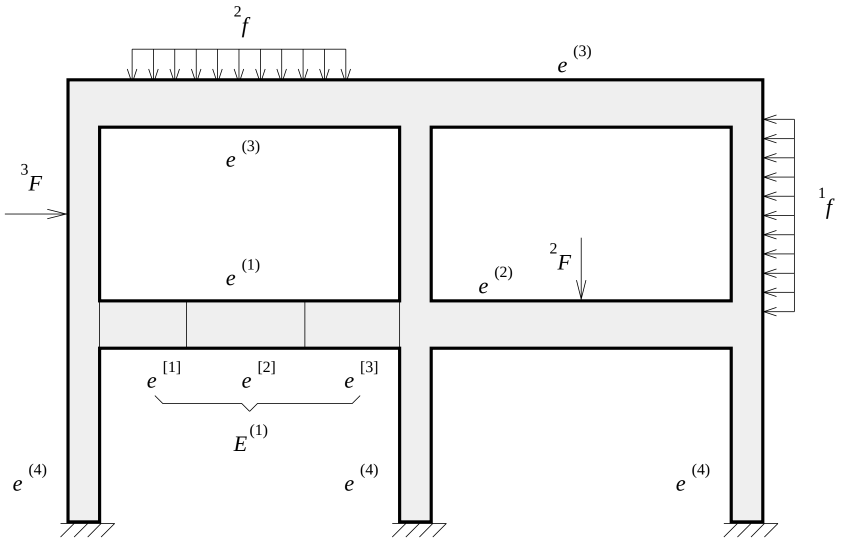


Figure 4.1: An example of a frame structure

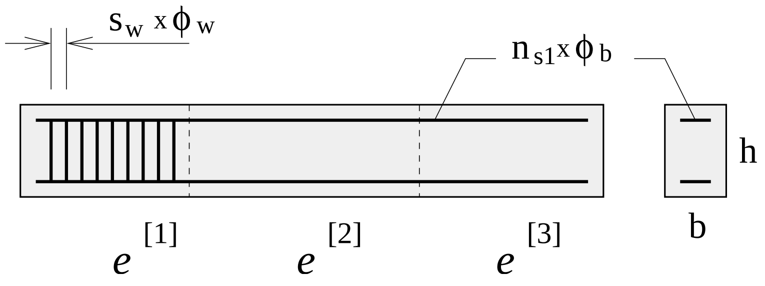


Figure 4.2: An example of a design element

like dimensions of the cross-section, the area and the diameter of the bending reinforcement etc. In addition, we assume that the structure is discretized into $n_e$ ***finite elements***, used for the determination of an internal forces distribution. In the sequel, we will denote a quantity $X$ related to the $i$-th finite element as $X^{[i]}$ while a quantity related to the $i$-th design element as $X^{(i)}$, i.e., values related to finite elements are indexed by square brackets, while quantities connected with design elements are denoted by round ones. Further, $e^{[i]}$ and $e^{(i)}$ are used for the $i$-th finite and design elements, respectively. The symbol $E^{(i)}$ is reserved for the set of finite elements, related to the $i$-th design element, i.e.,

$$E^{(i)} = \left\{ e^{[j]} : e^{[j]} \cap e^{(i)} \neq \emptyset, j = 1, \ldots, n_e \right\}. \tag{4.1}$$

Furthermore, the analyzed structure is supposed to be loaded by $n_c$ user-supplied load cases. A quantity $X$ related to the $i$-th design element and the $c$-th load case is denoted as ${}^{c}X^{(i)}$.

As described in the previous paragraph, the design element is used for the definition of basic optimization parameters (see Fig. 4.2). In our context, the design optimization parameters are the cross-section dimensions $b$ and $h$, the diameter of bending reinforcement $\phi_b$, the number of reinforcing bars located at the upper and the bottom surfaces of the design element denoted by $n_{s1}$ and $n_{s2}$ and, alternatively, the diameter of shear reinforcement $\phi_w$ and the spacing of stirrups $s_w$. We assume that the cross-sectional dimensions and stirrup spacing vary with a given

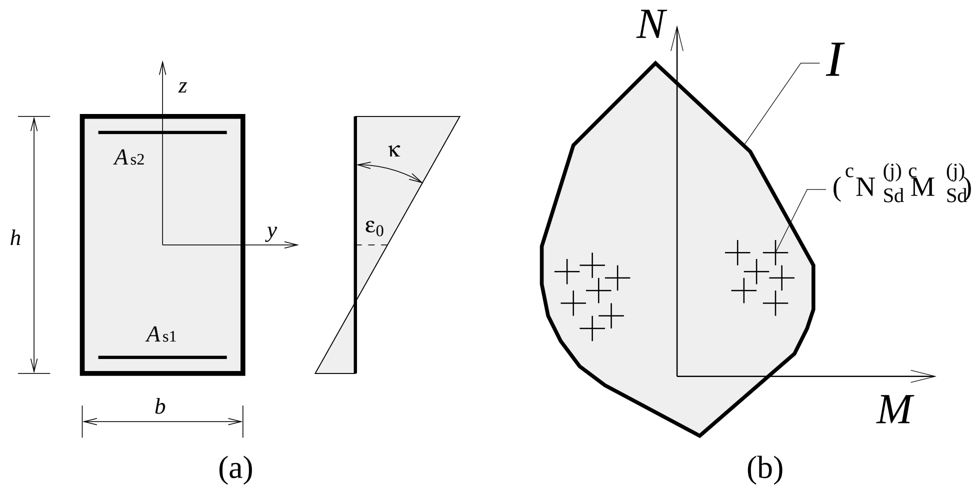


Figure 4.3: The cross-section scheme: (a) a plane of deformation, (b) an interaction diagram

discrete difference (e.g., 0.025 m), while $\phi_b$ and $\phi_w$ are selected from a given list of available dimensions.

### 4.2.2 *Ultimate limit state*

Generally speaking, the structural requirements imposed by a chosen design standard (e.g., EC2 [Eurocode 2, 1991] considered in this work) can be divided into two basic categories: load-bearing capacity and serviceability requirements. In the present work, the load-bearing capacity requirements, discussed in the present section, are directly incorporated into the reinforcement design. The serviceability requirements, on the other hand, are taken into account as the second optimization objective, see Section 4.2.3.2.

In our previous works, see e.g. [Lepš et al., 1999], the optimization of cross-section reinforcement was carried out simultaneously with the determination of geometrical parameters of the structure. This approach, however, does not seem to be feasible for larger structures because it would result in a huge amount of optimized variables, rendering the whole problem unmanageable (see Section 3.2.3, where a design problem of one beam consists of eighteen variables!). Thus, we employ a conceptually simple procedure aimed at the reduction of the problem size based on powerful algorithms for fast evaluation of internal forces that were developed in [Vondráček, 2001].

First of all, we briefly list the basic ideas of the procedure of the evaluation of internal forces employed in this work and refer an interested reader either to Appendix A or to works [Vondráček, 2001] or [Vondráček and Bittnar, 2002] for more detailed discussion. To that end, we assume that a given polygonal cross-section is subjected to a given linear distribution of the $\varepsilon_x$ strain given by

$$\varepsilon_x(z) = \varepsilon_0 + z\kappa, \tag{4.2}$$

where $\varepsilon_0$ is the strain at the coordinate system origin and $\kappa$ is the curvature in the $z$ direction (see Fig. 4.3a). Further, the response of a material is governed by a constitutive equation

$$\sigma_x = \sigma(\varepsilon_x). \tag{4.3}$$

The internal forces $N_x$ and $M_y$ are then provided by the well-known relations

$$N_x = \iint_A \sigma_x \,\mathrm{d}A, \qquad M_y = \iint_A \sigma_x z \,\mathrm{d}A. \tag{4.4}$$

Converting the area integrals (4.4) into boundary integrals by the Gauss-Green formula together with the fact, that the cross-section is polygonal, yield after some manipulations (see again Appendix A)

$$\begin{aligned} N_x &= -\frac{1}{\kappa^2}\sum_{i=0}^{n_p-1}\frac{1}{k_i}\Big(ss(\varepsilon^{(i+1)}) - ss(\varepsilon^{(i)})\Big), & (4.5)\\ M_y &= -\frac{1}{\kappa^3}\sum_{i=0}^{n_p-1}\frac{1}{k_i}\Big[(\xi-\varepsilon_0)ss(\xi) - 2ss(\xi) - 2sss(\xi)\Big]_{\varepsilon^{(i)}}^{\varepsilon^{(i+1)}}, & (4.6)\end{aligned}$$

where $n_p$ is the number of polygon segments, $k_i$ is the tangent of $i$-th polygon segment, $\varepsilon^{(i)}$ is the value of the strain at the $i$-th polygon vertex and values $ss(\cdot)$ and $sss(\cdot)$ follow from recursions

$$sss(\epsilon) = \int_0^{\epsilon} ss(\xi)\,\mathrm{d}\xi = \int_0^{\epsilon}\Big[\int_0^{\xi} s(\eta)\,\mathrm{d}\eta\Big]\mathrm{d}\xi = \int_0^{\epsilon}\Big[\int_0^{\xi}\big[\int_0^{\eta}\sigma(\psi)\,\mathrm{d}\psi\big]\,\mathrm{d}\eta\Big]\mathrm{d}\xi. \tag{4.7}$$

For detailed derivation and discussion of these relations together with the treatment of degenerate cases (i.e., $\kappa \to 0$ or $k_i \to 0$) we again refer to Appendix A and the works [Vondráček, 2001] and [Vondráček and Bittnar, 2002].

Once we are able to evaluate internal forces for a given plane of deformation determined by $\kappa$ and $\varepsilon_0$, the boundary of the *interaction diagram* $\mathcal{I}$ (see Fig. 4.3b) for a given cross-section can be simply constructed by evaluating the values of the bending moment $M_y$ and the normal force $N_x$ for a given set of extremal deformation planes (see Section A.4). Then, the cross-section can sustain the given normal force $N_{Sd}$ and the bending moment $M_{Sd}$ iff

$$(N_{Sd}, M_{Sd}) \in \mathcal{I}. \tag{4.8}$$

In the design procedure, we assume that we are provided with the dimensions of a cross-section $b$ and $h$ and the diameter of the longitudinal reinforcing bars $\phi_b$. Next, the Codes of Practice provide us with the minimum and maximum values of reinforcement areas $A_{s1} + A_{s2}$, which can be easily converted to a minimum/maximum number of reinforcing bars $n_{s,min}$ and $n_{s,max}$. Then, one can find the minimum reinforcement area such that the condition (4.8) holds for all elements and load cases, i.e.

$$({}^{c}N_{Sd}^{[j]}, {}^{c}M_{Sd}^{[j]}) \in \mathcal{I}, j \in E^{(i)}, i = 1,\ldots,n_d, c = 1,\ldots,n_c. \tag{4.9}$$

Although the proposed procedure is extremely simple, it performs satisfactorily thanks to the very efficient implementation of internal forces evaluation. Furthermore, it effectively eliminates infeasible solutions and thus substantially decreases the dimensionality of the problem.

### *4.2.3 Objective functions*

Having defined (and appropriately reduced) the domain of all admissible structures, the most suitable solution from this set is to be selected. For this purpose we need to measure the quality of each structure. As mentioned previously, we have selected both the total price of a structure and the maximum deflection as the objectives to be optimized. Note that some deflection limit usually serves as constraint during an optimization process while here is an objective. In fact, this is the direct application of the methodology presented in Section 2.3.3 - a shift from constraints to objectives.

#### *4.2.3.1 Design economy*

The total price of the structure follows from the expression

$$f(\mathbf{X}) = V_c P_c + W_s P_s + A_c P_{Ac}, \tag{4.10}$$

where $\mathbf{X}$ stands for the vector of design variables, $V_c$ is the volume of concrete, $W_s$ is the weight of steel and $A_c$ is the area of concrete connected with form-work; $P_c$, $P_s$ are the prices of concrete per unit volume and steel per kilogram, $P_{Ac}$ is the price of form-work per square meter, which is added to simulate construction costs[3].

#### *4.2.3.2 Serviceability limit state*

As the second objective of the optimization, the maximum deflection of the analyzed structure will be considered. In the current implementation, the maximum sagging of the $i$-th design element due to the $c$-th load case is determined on the basis of a simple numerical integration algorithm. To this end, suppose for simplicity that the internal forces distribution for the given load case and design element is known from the elastic analysis. For the given values of the bending moment $M_y$ and the normal force $N_x$[4], the parameters of the deformation plane $\varepsilon_0$ and $\kappa$, recall Fig. 4.3a, can be efficiently found by the Newton-Raphson algorithm [Vondráček, 2001, Chapter 5]. Then, under the assumptions of small deformations and small initial curvature, the deflection curve follows from the familiar relation,

$$\frac{\mathrm{d}^2 w(x)}{\mathrm{d}x^2} = -\kappa(M_y(x)), \tag{4.11}$$

which yields, after integrating Eq. (4.11) twice,

$$w(x) = -\int_0^x \Big[ \int_0^\xi \kappa(M_y(\eta))\,\mathrm{d}\eta \Big]\,\mathrm{d}\xi + C_1 x + C_2 \tag{4.12}$$

with integration constants $C_1$ and $C_2$ determined from boundary conditions for a given design element. Also the analyzed element can be split into several equidistant parts with the length $\Delta x$ and Eqs. (4.11) and (4.12) can be replaced by their discretized counterparts. The maximum deflection of the design element is then straightforwardly determined as the extremal value found for all load cases.

[3] See the work [Sarma and Adeli, 1998] for more than seventy references dealing with cost optimization of reinforced concrete structures.

[4] Note that for the notational simplicity, indices $c$ and $i$ are omitted in the present section.

### 4.2.4 Multi-objective optimization algorithm

The Strength Pareto Evolutionary Algorithm (SPEA), firstly introduced by Zitzler and Thiele [Zitzler and Thiele, 1999] in 1999, was selected as the multi-objective optimizer in the present study. The key ideas of this algorithm can be summarized as [Zitzler and Thiele, 1999]: storing non-dominated solutions externally in a second, continuously updated population, fitness assignment with respect to the number of external non-dominated points that dominate it, preserving population diversity using the Pareto dominance relationship and incorporating a clustering procedure for the reduction of the non-dominated set. Moreover, all these features are actually independent of the form of crossover and mutation operators. Therefore, it is possible to use operators developed for the single-objective optimization problem [Lepš and Šejnoha, 2000] without any changes. Last, but certainly not least, advantage of this algorithm is its conceptual simplicity and freely available C++ source code. An interested reader is referred to the article [Zitzler and Thiele, 1999] and the Ph.D. thesis [Zitzler, 1999] for more detailed description of the algorithm as well as extensive numerical investigation of its performance.

## 4.3 Examples and results

We demonstrate the aforementioned design procedure on the benchmark problems, already considered in [Laníková, 1999]. In particular, two different statically determined structures are examined.

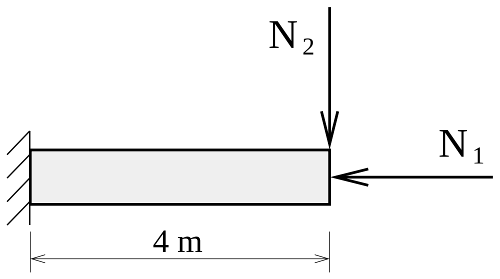


Figure 4.4: First example - a cantilever beam.

### 4.3.1 A cantilever beam

Firstly, a cantilever beam, see Fig. 4.4, with the 4.0 meter span was studied. A concrete model with cylindrical ultimate strength equal to 20 MPa (Class C 16/20) was considered with steel model with the 410 MPa yield stress (Class V 10 425). The cantilever was loaded with two loading cases: (${}^1N_1 = 1800$ kN, ${}^1N_2 = 100$ kN) and (${}^2N_1 = 300$ kN, ${}^2N_2 = 100$ kN). The theoretical cover of steel reinforcement was set to $0.05$ m and the supposed diameter of shear reinforcement was $0.06$ m. In the design procedure, the beam width was restricted to $b \in \{0.3, 0.35, 0.4, 0.45\}$ m while the heights $h \in \{0.4, 0.5, 0.6\}$ m were considered. The longitudinal reinforcement profiles were selected from the list $\phi_b \in \{$10, 12, 14, 16, 18, 20, 22, 25, 28, 32, 36$\}$ mm. The individual unit prices appearing in Eq. (4.10) were considered $P_c = 2,500$ CZK/m$^3$, $P_s = 25$ CZK/kg and $P_{Ac} = 1,250$ CZK/m$^2$, respectively[5]. Finally, the

[5] The symbol CZK stands for Czech Crowns.

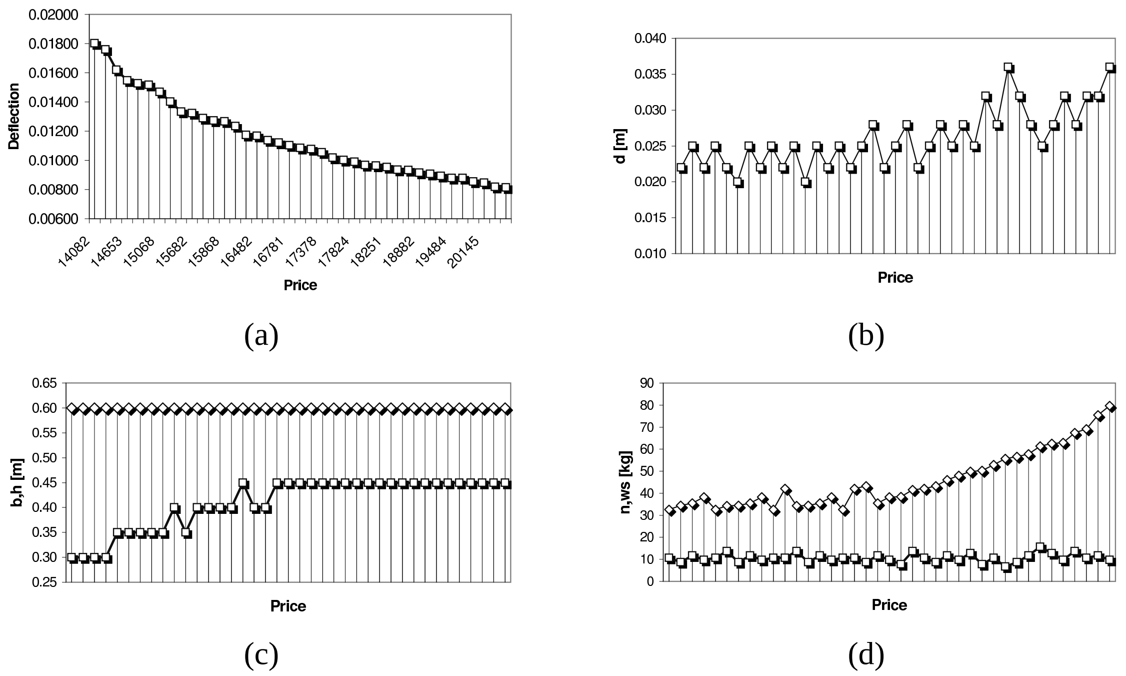


Figure 4.5: Results for the cantilever beam example: (a) Pareto-front and Pareto sets - (b) steel profiles $d$, (c) the width $b$ and the height $h$, (d) the number of steel bars $n$ and the amount of steel $w_s$.

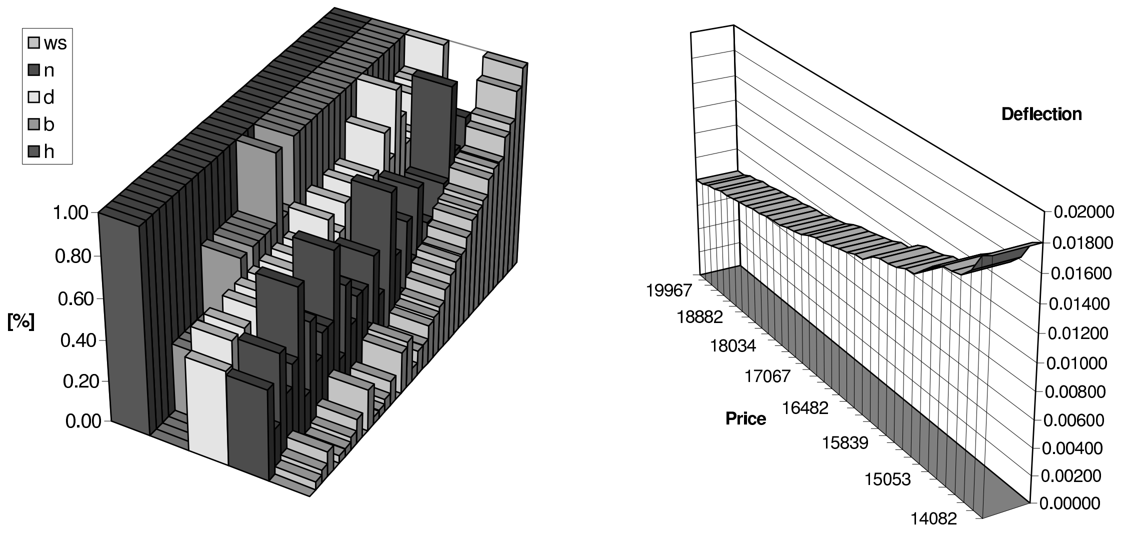


Figure 4.6: Results for the cantilever beam example depicted in 3D.

integration step $\Delta x = 0.25$ m was considered for the deflection analysis, see Section 4.2.3.2.

The results are shown in Fig. 4.5 and Fig. 4.6 by the methodology presented in Appendix B. It can be seen that there are 39 non-dominated solutions, which are characterized by the maximal value of the height of the beam $h$ and by non-monotonously increasing amount of steel, see Fig. 4.5(d). It is also important, that solutions are not created by the small steel profiles which are probably not able to sustain applied internal forces.

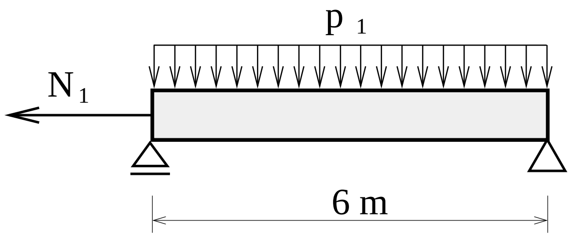


Figure 4.7: Second example - a simply supported beam.

### *4.3.2 A simply supported beam*

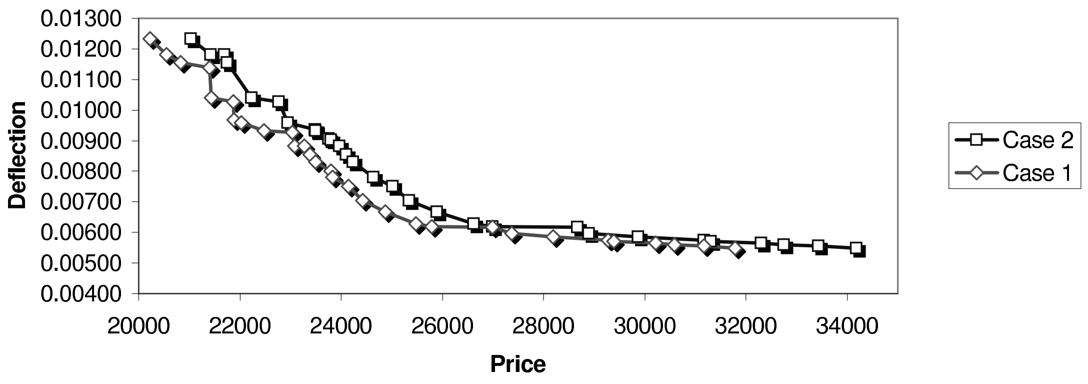
The second example studied was a simply supported beam, see Fig. 4.7. The span was considered 6 m. The concrete and the steel models were the same as in the previous example, as well as a reinforcement cover, a shear reinforcement profile, geometrical parameters $b$, $h$ and $\phi_b$. The beam was loaded with three loading cases: ($^1p_1 = 62.5$ kN/m, $^1N_1 = -240$ kN), ($^2p_1 = 62.5$ kN/m, $^2N_1 = -1440$ kN) and ($^3p_1 = 62.5$ kN/m, $^3N_1 = 480$ kN).

In this example, we simulated the scenario of a growing price of steel. The question placed here is: “What will happen if a price of steel grows by 20%?”. Therefore, Case 1 is characterized by unit prices $P_c = 2,500$ CZK/m$^3$, $P_s = 25$ CZK/kg and $P_{Ac} = 1,250$ CZK/m$^2$ and Case 2 by the same values for $P_c$ and $P_{Ac}$, but the value of $P_s$ is set to $30$ CZK/kg.

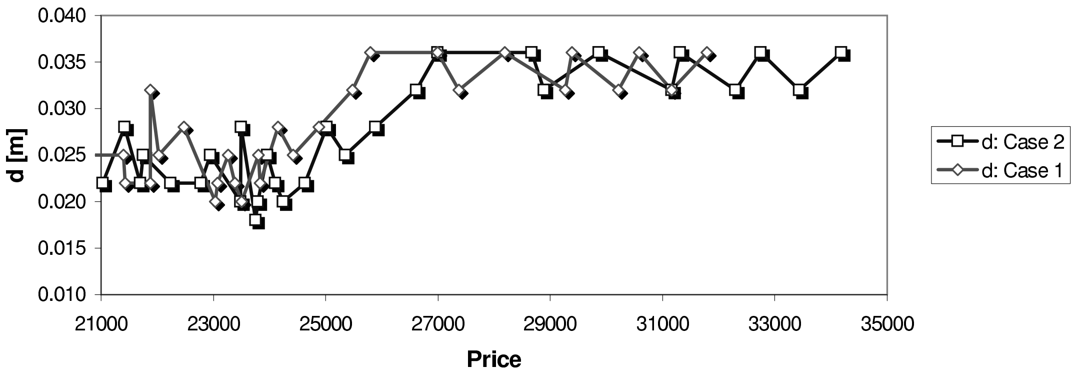
Results are shown again in 2D in Fig. 4.8 and in 3D in Fig. 4.9 and Fig. 4.10. Case 1 is created by 30 non-dominated solutions and Case 2 by 29 and both cases are characterized by the maximal value of the height of the beam $h$. At first sight, the growth of the steel price shifts the Pareto-front of the more expensive Case 2 to the right, see Fig. 4.8(a). But still, there are some designs, where both cases meet each other. The next interesting point is the decrease of the amount of steel, as can be visible in Fig. 4.8(d). And finally, by inspecting both Pareto-sets it comes that the last 15 solution are the same - they differ only in the price. Thus, such optimal designs can be seen as stable (or at least less sensitive) with respect to perturbation of steel price and hence more “robust” from the practical point of view.

## *4.4 Conclusion for the design of RC frames*

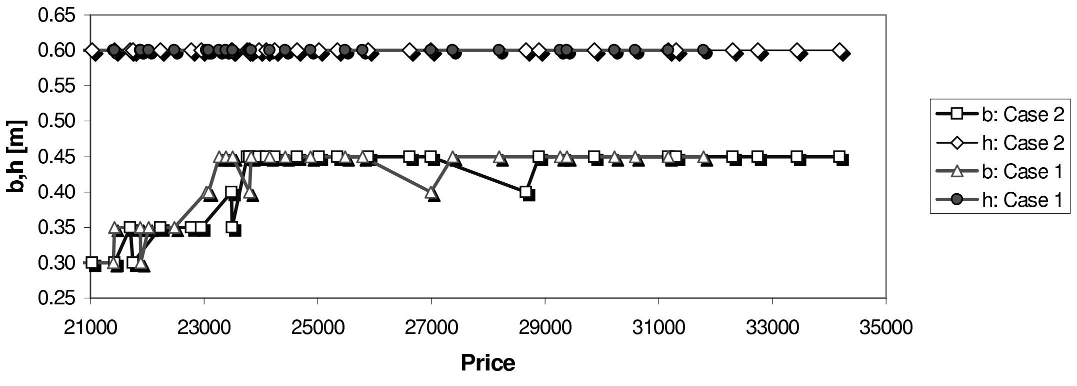
The results of the SPEA algorithm have revealed that there are 39, 30 and 29 non-dominated solutions for the cantilever and simply supported beam problems, respectively. The trade-off surfaces for both problems appear in Fig. 4.5(a) and Fig. 4.8(a). It is clearly visible that even for these rather elementary design tasks, both Pareto-optimal fronts are non-convex and non-smooth due to the discrete nature of the optimization problem. This fact justifies the choice of the selected optimization strategy and suggests its applicability to more complex structural design problems.



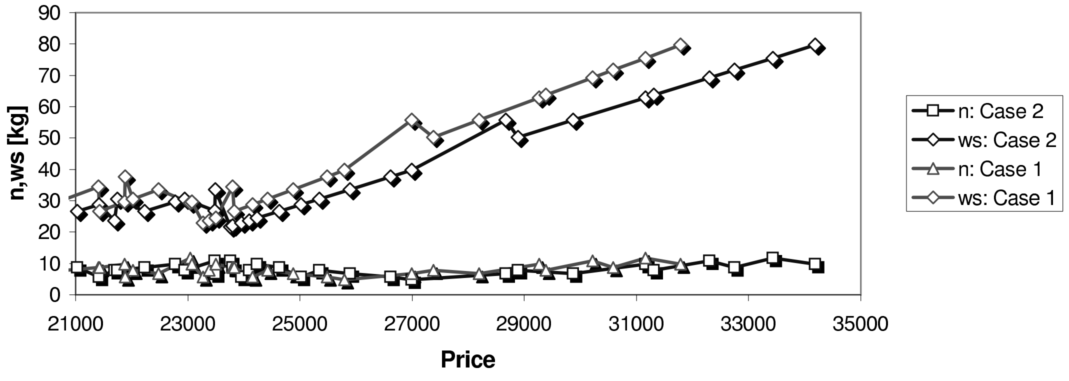
(a)



(b)



(c)



(d)

Figure 4.8: Results for the simply supported beam example: (a) Pareto-fronts and Pareto sets - (b) steel profiles $d$, (c) the widths $b$ and the heights $h$, (d) the number of steel bars $n$ and the amount of steel $w_s$.

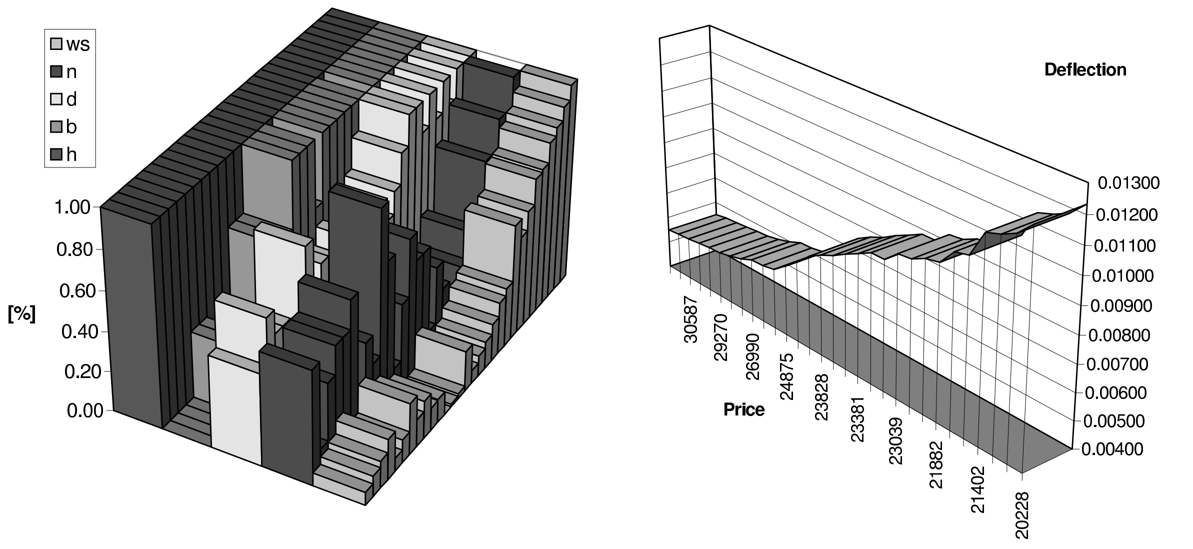


Figure 4.9: Results for the simply supported beam example depicted in 3D, cheaper Case 1.

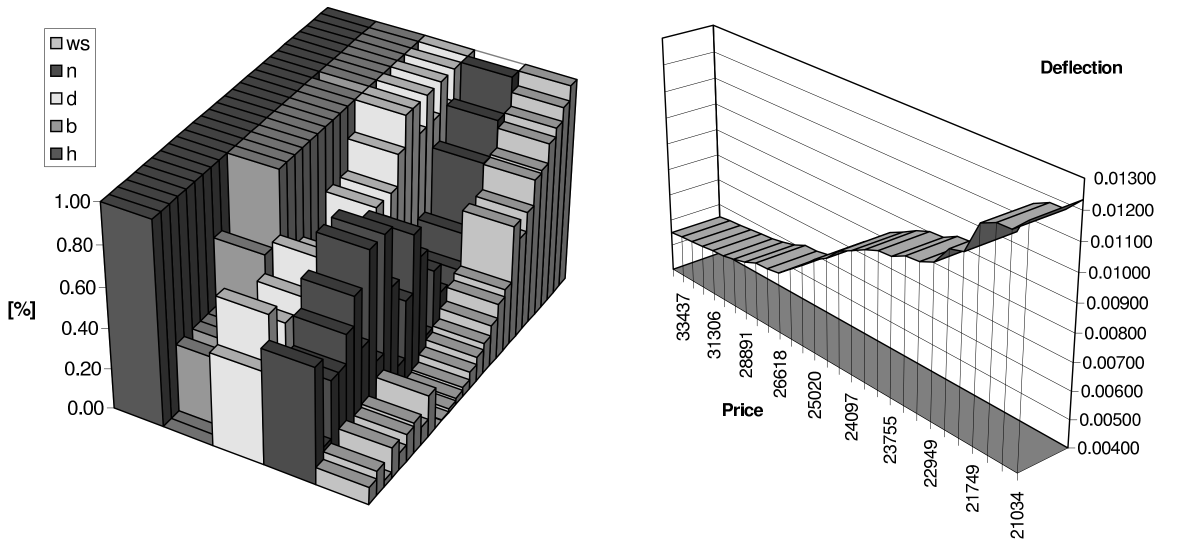


Figure 4.10: Results for the simply supported beam example depicted in 3D, more expensive Case 2.

Chapter 5

# SOFT-COMPUTING METHODS APPLIED ON MICROPLANE MODEL PARAMETERS IDENTIFICATION

The purpose of computation is insight, not numbers.

Richard Hamming

## *5.1 Introduction*

The great class of technical and scientific tasks leads to problems, described by a system of partial differential equations. In the last few decades the so-called artificial intelligence or soft-computing methods have been developed as alternatives to the traditional solutions of problems, which are difficult to be defined, described or resolved using traditional methods. Our research deals particularly with one part of soft-computing methods called artificial neural networks (NN)[1] [Tsoukalas and Uhrig, 1997, Yagawa and Okuda, 1996]. They were developed to simulate the processes in a human brain but later on it was discovered that they can be effectively used for many problems like pattern recognition, different approximations and predictions, control of systems, etc. In this work, they will be used "only" as general approximation tools.

The behavior of such neural network is determined by an initial training process. It consists of finding the so-called synaptic weights, which have influence on the response of a neural network, depending on the different components of an input signal. The training of a neural network could be considered as an optimization process, because it can be seen as a minimization of neural network output error. Therefore, Evolutionary Algorithms, see Section 2.2.1, can be used and can outperform traditional gradient-based methods that are usually applied here. The current implementation is then easy, because the synaptic weights of a neural network act as variables of the Evolutionary Algorithm's fitness function.

Using Evolutionary Algorithms for training neural networks is not a new idea, see e.g. the work [Tsoukalas and Uhrig, 1997]. The results show that due to their ability to avoid local extremes, it is an efficient way. This work also deals with the **SADE** algorithm (see Section 3.3.2) in the training process. First, we compare **SADE** training with the traditional Backpropagation method on a simple task. Then we will use **SADE** training as well as the **SADE** algorithm alone for solving a much more complicated Civil Engineering problem - an estimation of parameters of a constitutive model for concrete called microplane model. One way to do this is to fit these values using an experimenter's own experience. As one of more up-to-date approaches to estimate these parameters the neural network simulation could be employed.

This chapter is organized as follows. Section 5.2 presents a comparison of the Backpropagation and the **SADE** algorithm for neural network training. In Section 5.3, the microplane

[1] Hereafter we will use only the term *neural network* instead of *artificial neural network* for the sake of simplicity.

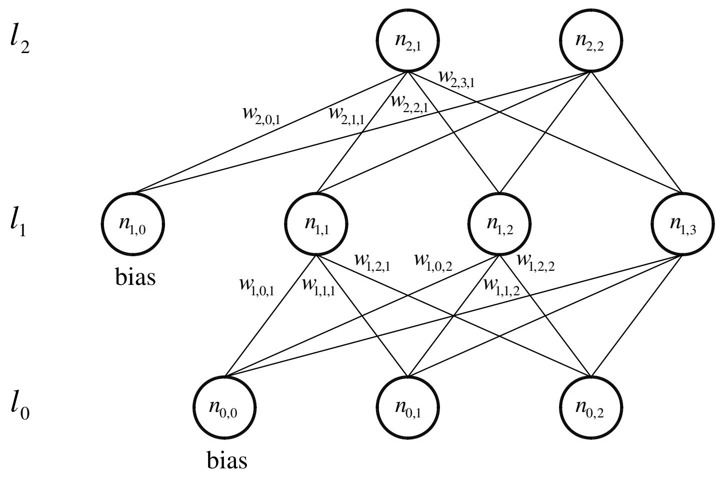


Figure 5.1: A neural network architecture.

material model will be introduced. Section 5.3.1 brings an estimation of microplane parameters by a neural network trained on approximations of the stress-strain curves. It shows that some properties can be predicted well, especially Young's modulus $E$, the parameter $k_1$ and coefficient $c_{20}$. In Section 5.3.2, a parallel version of the algorithm **SADE** is directly applied to obtain required parameters by varying them within a nonlinear finite element analysis (see also the conference paper [Kučerová et al., 2003] for more details). The first main result is that this time consuming analysis can be solved by a parallel analysis in reasonable time. The second outcome is the fact that the objective function corresponding to the tuning problem has several local minima, which are characterized by similar values but are far from each other. To solve the above mentioned obstacles and in the view of recent research in this domain [Lehký, 2003], a new methodology is presented in Section 5.3.3. Particularly, an application of the Latin Hypercube Sampling method is applied to investigate the influence of individual material model parameters. Finally, several results along with some concluding remarks are presented.

## 5.2 Optimizing synaptic weights of neural networks

This section (already presented as a conference paper [Drchal et al., 2002]) deals with an application of the **SADE** algorithm introduced earlier in the process of training a neural network (NN). We would like to show, that this problem is multi-modal and hence an Evolutionary Algorithm is much better than the traditionally used gradient-based Backpropagation method. This work does not aim at systematic research on neural networks and therefore the description of a neural network will be presented only in a sketchy form.

### 5.2.1 Description of a neural network

Here, an artificial neural network, which will be later used for material parameters prediction, is presented. More precisely, a layered fully connected feed-forward neural network with bias neurons [Tsoukalas and Uhrig, 1997] is used. Its architecture is shown in Figure 5.1.

In general, a neural network is created to map the input vector $I = (I_0, I_1, \ldots, I_m)$ on a target vector $T = (T_0, T_1, \ldots, T_n)$. There are $L$ layers denoted as $l_0, l_1, \ldots, l_{L-1}$, where $l_0$ is the input layer and $l_{L-1}$ is the output layer. The layer $l_i$ has $N_i$ neurons denoted as $n_{i,1}, n_{i,2}, \ldots, n_{i,N_i}$. Each layer except the output layer has the bias neuron $n_{i,0}$. The connections are given the so-called synaptic weights $w_{l,i,j}$, where $l$ denotes a layer, $i = 0, 1, \ldots, N_{l-1}$ is the index number of a neuron in the preceding layer ($i = 0$ for bias neurons) and $j = 1, 2, \ldots, N_l$ is the index number of a neuron in the layer $l$. The output of the neuron $n_{l,j}$ is then defined as

$$\begin{aligned} O_{l,j} &= f_{\text{act}}\left(\sum_{i=0}^{N_{l-1}} O_{l-1,i} \cdot w_{l,i,j}\right) , \quad l = 1, 2, \ldots, L-1 , \quad j = 1, 2, \ldots, N_l , && (5.1) \\ O_{0,j} &= I_j , \quad j = 1, 2, \ldots, N_0 , && (5.2) \\ O_{l,0} &= 1 , \quad l = 0, 1, \ldots, L-1 , && (5.3) \end{aligned}$$

where $f_{\text{act}}$ is an activation function. In our current implementation the activation function has the following form:

$$f_{\text{act}}(\Sigma) = \frac{1}{(1 + e^{-\alpha/\Sigma})} , \tag{5.4}$$

where $\alpha$ is the gain of the $f_{\text{act}}$. The value $\alpha = 0.8$ is used in the next calculations. The output vector of each layer $l_i$ is denoted as $O_i = (O_{i,1}, O_{i,2}, \ldots, O_{i,N_i})$. Finally, the neural network is propagated as follows:

1. Let $l = 1$.
2. Calculate $O_{l,i}$ for $i = 1, 2, \ldots, N_l$.
3. $l = l + 1$.
4. If $l < L$ go to 2, else $O_{L-1}$ is the network's approximation of $T$.

The output error, which is used as a measure of training accuracy, is defined as

$$\varepsilon = \frac{1}{2} \sum_{i=1}^{N_{L-1}} (T_i - O_{L-1,i})^2 . \tag{5.5}$$

### *5.2.2 Backpropagation*

For the training of a neural network, we employed the well-known Backpropagation algorithm. More specifically, the momentum version [Tsoukalas and Uhrig, 1997] is used to speed up the convergence. A short description of the method follows. Note that the error connected with the output of the neuron $n_{l,i}$ is devoted as $e_{l,i}$. The algorithm could be written in the following form:

1. for $i = 1, 2, \ldots, N_{L-1}$ calculate $e_{L-1,i} = \alpha \cdot O_{L-1,i} \cdot (1 - O_{L-1,i}) \cdot (T_i - O_{L-1,i})$.
2. Set $l = L - 1$.
3. $e_{l-1,i} = \alpha \cdot O_{l-1,i} \cdot (1 - O_{l-1,i}) \cdot \left(\sum_{j=1}^{N_l} w_{l,i,j} \cdot e_{l,j}\right)$ for each neuron $i$ in ($l - 1$)$^{th}$ layer.

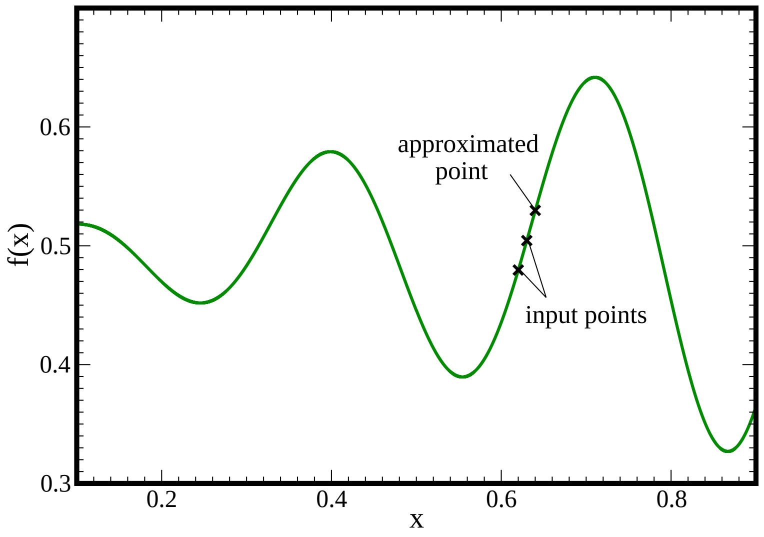


Figure 5.2: A function used for testing: $f(x) = 0.2x\ \sin(20x) + 0.5$.

4. $l = l - 1$.
5. While $l > 1$ go to 3.
6. All the weights are adjusted: $w_{l,i,j} = w_{l,i,j} + \Delta w_{l,i,j} + \mu \cdot \Delta w'_{l,i,j}$, where $\Delta w_{l,i,j} = \eta \cdot e_{l,j} \cdot O_{l-1,i}$ .

The term $\Delta w'_{l,i,j}$ stands for the value from the previous training iteration, $\eta$ is the learning constant, and $\mu$ is the momentum coefficient. The typical values are $\eta = 0.5$ and $\mu = 0.9$.

### *5.2.3* SADE *training vs. Backpropagation*

As competitor, the algorithm **SADE** was selected - not only because it is directly based on real numbers, but also for its ability to solve problems with high number of variables, see Chapter 3.

The training ability was tested on a simple task. A goniometric function

$$f(x) = ax\ \sin(bx) + c \tag{5.6}$$

was used. The following sequence of consecutive points $x_1, x_2, \dots, x_n$ was generated:

$$\begin{aligned} x_1 &= \text{a random number from training interval}\,, \\ x_i &= x_{i-1} + d\,, \quad i = 2, 3, \dots, n\ , \quad d \text{ is a constant}\,, \end{aligned} \tag{5.7}$$

i.e., all points are equidistant and the sequence starts at a random point. A network input vector was created as $I = (f(x_1), f(x_2), \dots, f(x_{n-1}))$ and the next sequential point $f(x_n)$ was expected on the output. Typically, two input points ($n = 3$) were used. The three-layered neural network had two neurons in the input layer, three neurons in the hidden layer and one neuron in the output layer, and, as an addition, bias neurons (see Section 5.2.1). The output error according to (5.5) is

$$\varepsilon = \frac{1}{2}\left(f(x_n) - O_{2,1}\right)^2\ . \tag{5.8}$$

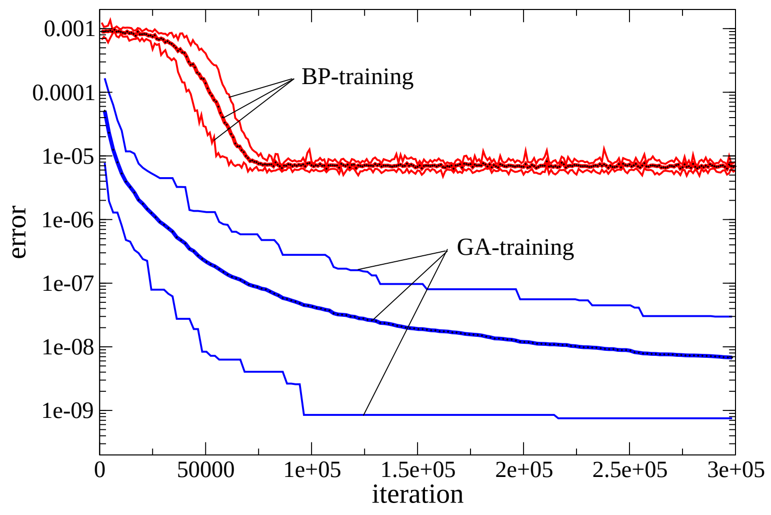


Figure 5.3: An error function in estimation of neural network weights during an optimization process.

The constants were set to $a = 0.2,\ b = 20,\ c = 0.5$ in order to avoid the necessity of normalization (the network output ranges from $0$ to $1$). Other functions were also tested and the results were similar. The situation is shown in Figure 5.2.

We compared both optimization methods in $100$ runs of testing computations, each starting from a new random generator seed. Each run comprised $300,000$ iterations. The value of an error function was saved every $1000^{th}$ iteration. The minimum, maximum, and average values of the error function (calculated from $100$ independent testing runs) are shown in Figure 5.3 for both the **SADE** and the Backpropagation method. The graph shows that **SADE** training clearly outperforms the Backpropagation methodology. Figure 5.4 also shows the distribution of the error in $f(x)$ approximation on the interval $\langle 0.1; 0.9 \rangle$.

## *5.3 Microplane model parameters prediction*

Concrete as a man-made heterogeneous material shows very complex non-linear behavior, which is extremely difficult to model both theoretically and numerically. The microplane material model [Bažant et al., 2000], [Jirásek and Bažant., 2001] is a fully three-dimensional material law that includes tensional and compressive softening, damage of the material, different combinations of loading, unloading and cyclic loading. It can also describe the development of anisotropy within the material. As a result, it is fully capable of predicting behavior of real-world concrete structures [Němeček et al., 2002] once provided with proper input data.

The major disadvantage of this model is, however, a large number of phenomenological material parameters. Although the authors of the model proposed a heuristic calibration procedure [Caner and Bažant, 2000], it is based on the trial-and-error method and provides guide to determination of only selected material parameters. Therefore, a reliable procedure for parameters identification is on demand. In particular, a certain type of concrete is described by eight parameters: Young's modulus $E$, Poisson's ratio $\nu$, and six other parameters ($k_1$, $k_2$, $k_3$, $k_4$, $c_3$, $c_{20}$), which do not have a simple physical interpretation, and therefore it is difficult to determine their values from experiments.

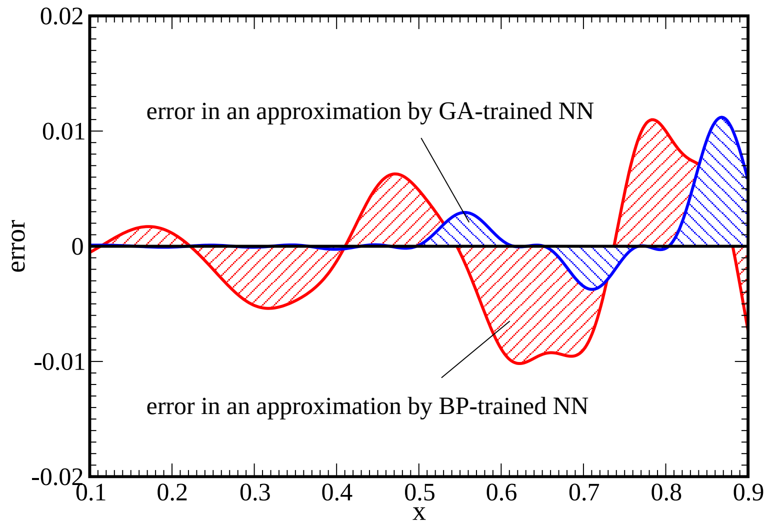


Figure 5.4: An error distribution in the approximation of $f(x) = 0.2x\ \sin(20x) + 0.5$.

The common practice for an experimenter is to employ the trial and error method to tune stress-strain diagrams by varying individual model parameters, see [Caner and Bažant, 2000] or [Němeček, 2000]. This is not a trivial task because of highly non-linear behavior. Nevertheless, several limits can be found in the literature for these parameters. In the current implementation, the suggested appropriate bounds were set to the values shown in Table 5.1.

$$\begin{array}{rcl}
E & \in & \langle 20.0, 50.0\rangle \text{ GPa} \\
\nu & \in & \langle 0.1, 0.3\rangle \\
k_1 & \in & \langle 0.00008, 0.00025\rangle \\
k_2 & \in & \langle 100.0, 1000.0\rangle \\
k_3 & \in & \langle 5.0, 15.0\rangle \\
k_4 & \in & \langle 30.0, 200.0\rangle \\
c_3 & \in & \langle 3.0, 5.0\rangle \\
c_{20} & \in & \langle 0.2, 5.0\rangle
\end{array}$$

Table 5.1: Boundaries for the microplane model parameters.

To define the problem more formally, the goal is to find microplane parameters from the stress-strain diagram of a test specimen in a uniaxial compression, see Figure 5.5.

The rather severe disadvantage of the microplane model is an extreme demand of computational time. As it was shown e.g. in [Němeček, 2000], the above presented example of a uniaxial test (Fig. 5.5) consumed more than 23 hours on a single processor PC with the Pentium II Xeon 400 MHz processor and 512 MB RAM. Therefore, a single finite element is used instead of the whole specimen. It was demonstrated in [Němeček, 2000] that this simplified model is not so unrealistic as it may appear at first sight; the differences in fitted parameters found by these two approaches were not significant.

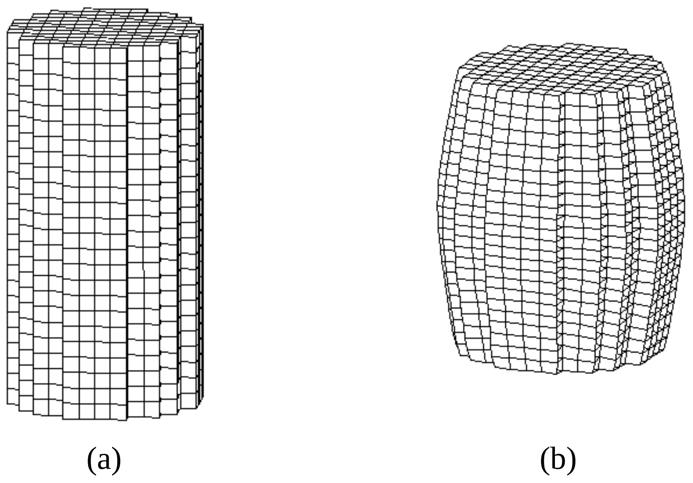


Figure 5.5: The computational model of a compression test a) at the start and b) at the end of loading.

### *5.3.1 Application of a NN on parameterized diagrams*

To start neural network training, it is necessary to prepare a sufficiently rich set of training data - these are the strain-stress diagrams for different kinds of a material, for which the corresponding values of the microplane model parameters are already known. For satisfactory training, such a set of training data should contain several hundred patterns. The results of the microplane model computer simulations were used as these patterns. The trial parameters were generated randomly from given intervals, see Tab. 5.1.

Next, the Pearson product moment correlation coefficient[2], defined as

$$cor = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y})^2}} \,, \tag{5.9}$$

is used as an sensitivity measure to investigate the influence of individual parameters to a structural response. From the results in the twenty consecutive points, see Fig. 5.6, the impact of Young's modulus $E$ especially in the elastic part of the curves is clearly visible. Also two parameters, $k_1$ and $c_{20}$, seem to be important. An effect of the other parameters is small and hence, for a neural network only hardly recognizable. This is why the following computations were focused only on an estimation of these two mentioned parameters.

At this point, a question arises, which representation of data should serve as an input for a neural network to produce its optimal performance. The first option that came into sight was to take the function values in determined points. This approach seemed to be inappropriate because the shapes of the strain-stress curves differ one from another too much (see Section 5.3.3 for such implementation). Also the size of the input layer of a neural network must correspond to the number of approximation points. Therefore if the number of approximation points is high enough to represent all the complexity of the function graph, the size of the neural network grows significantly and it leads to an enormous computation complexity.

An approximation of the strain-stress graph using a parametric function with a simple analytic description can be considered as another option. In this way, the parameters of the analytic

[2] For the comparison of influence of different types of correlation coefficients, see Appendix D.

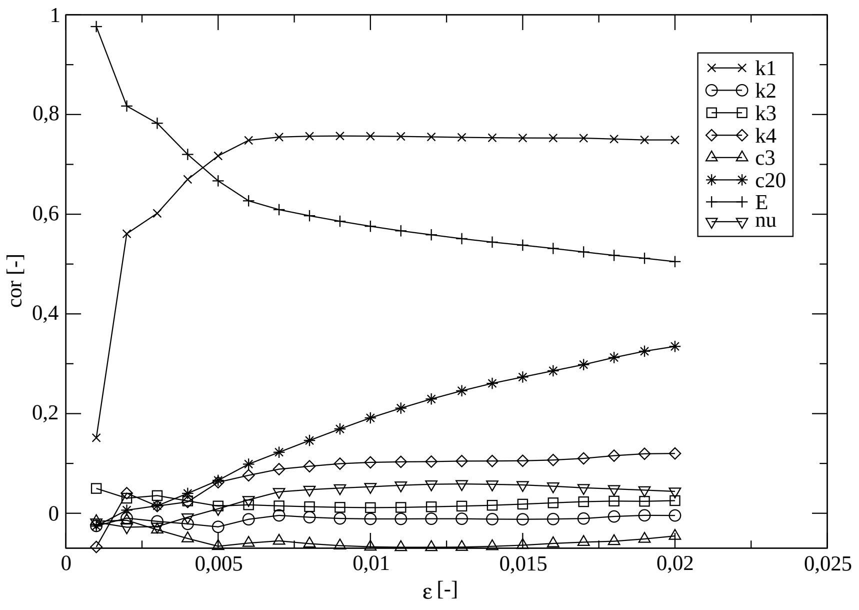


Figure 5.6: Pearson product moment correlation coefficient as sensitivity parameter of individual parameters to the stress-strain curve.

function can be used as an input vector for a neural network. A simple choice is to use a low order polynomial function. For certain patterns it was discovered that the polynomial function oscillates in the descending part of the graph which makes this approach of a less value. An approximation that consists of several partial elementary functions has proved to be more effective. In more detail, $3$ linear and $1$ exponential functions were used in the approximation of the strain-stress function in the intervals between points $0$, $\epsilon_1$, $\epsilon_2$, $\epsilon_3$ and $\epsilon_{last}$. The analytic description of these elementary functions is as follows:

$$\begin{aligned} \epsilon \in \langle 0; \epsilon_1 \rangle : &\quad \sigma_1(\epsilon) = k_1 \epsilon \ , \\ \epsilon \in \langle \epsilon_1; \epsilon_2 \rangle : &\quad \sigma_2(\epsilon) = k_2 \epsilon + q_2 \ , \\ \epsilon \in \langle \epsilon_2; \epsilon_3 \rangle : &\quad \sigma_3(\epsilon) = k_3 \epsilon + q_3 \ , \\ \epsilon \in \langle \epsilon_3; \epsilon_{last} \rangle : &\quad \sigma_4(\epsilon) = c e^{-a\epsilon + b} + d \ . \end{aligned}$$

In the following, we call this approximation *combined approximation*. It uses $12$ parameters: $\epsilon_1, \epsilon_2, \epsilon_3, k_1, k_2, q_2, k_3, q_3, a, b, c, d$. Three of them could be eliminated using conditions of continuity in the connecting points:

$$\begin{aligned} \epsilon_1 : &\quad \sigma_1(\epsilon_1) = \sigma_2(\epsilon_1) \rightarrow q_2 = (k_1 - k_2)\epsilon_1 \ , \\ \epsilon_2 : &\quad \sigma_2(\epsilon_2) = \sigma_3(\epsilon_2) \rightarrow q_3 = (k_2 - k_3)\epsilon_2 + q_2 \ , \\ \epsilon_3 : &\quad \sigma_3(\epsilon_3) = \sigma_4(\epsilon_3) \rightarrow d = k_3 \epsilon_3 + q_3 - c e^{-a\epsilon_3 + b} \ . \end{aligned}$$

The approximation is thus given by nine independent parameters $\epsilon_1$, $\epsilon_2$, $\epsilon_3$, $k_1$, $k_2$, $k_3$, $a$, $b$, $c$. The **SADE** algorithm was used to search for the values of these parameters. The optimized function was the least square function, which contained the difference among values of a stress-strain curve and an approximation function. Fig. 5.7 shows an example of a computed strain-stress curve, its *combined approximation* and (for comparison) also its approximation using a polynomial function of the eighth degree, which has also nine parameters.

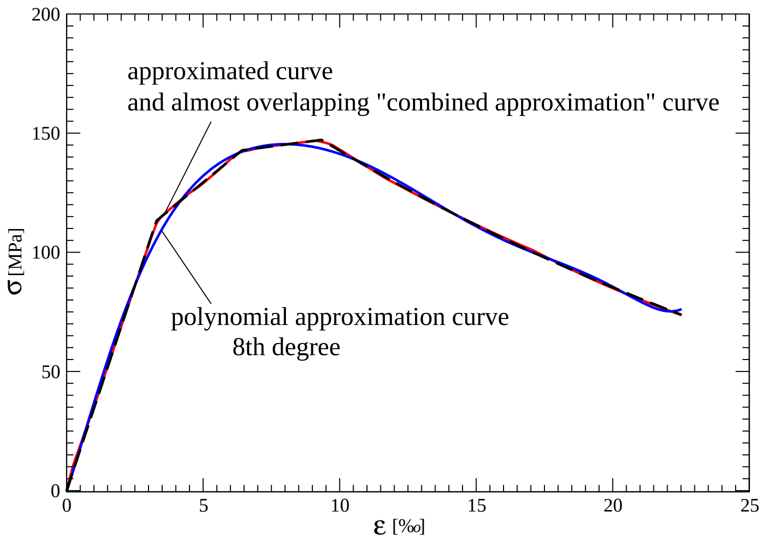


Figure 5.7: Approximations of a strain-stress curve.

The nine parameters of the *combined approximation* are used as the input data for a neural network, but they also contain some interesting information about the material as well. For instance, the slopes of linear sections have the meaning of the secant moduli. For neural network training itself, the input data as well as the target data (the known microplane model parameters) have to be normalized to the interval $\langle 0; 1\rangle$.

It is supposed that the neural network training process should run only once. Hence emphasis is put on the reliability of the estimation. Two different networks were trained - each for estimating one parameter and each of these networks had three layers. The size of the input layer was nine, which corresponds to the number of parameters of the *combined approximation*. In the second (hidden) layer, there were three neurons while there was only one neuron in the output layer. One bias neuron was added to the first and the second layer. This neural network layout contains thirty-one synaptic weights which means thirty-one variables of the optimized function. The training data set consisted of $606$ patterns[3]. During the training of neural networks, one single evaluation of the optimized function is equal to an average error of estimation of the target value on one hundred patterns. The training was stopped after a given number of iterations (set to one million) and then the average error of an estimation of the target values was computed on all the $606$ patterns. To avoid the stochasticity of the **SADE** algorithm, the whole training process was run hundred times. Tab. 5.2 shows the average error and the standard deviation in estimating the parameters $K_1$ and $C_{20}$ for hundred runs after one million iterations. Because the outputs of the neural network are from the interval $\langle 0; 1\rangle$, it was easier to evaluate the error based on these scaled values.

The influence of the error in the estimation by the neural network is shown in Figs. 5.8 and 5.9. The curves differ only by the values of the estimated parameter. Fig. 5.8 shows the curves for two different values of the parameter $K_1$ which differ just by $1.3525\%$, which is the average error of the estimation of this parameter. Fig. 5.9 shows four curves with varying value of the parameter $C_{20}$, two curves represent the boundary values of $C_{20}$ and the other two differ by

[3] Originally, there were $1000$ sets, but the missing parameters constitute a non-realistic material that was not able to describe a compression test.

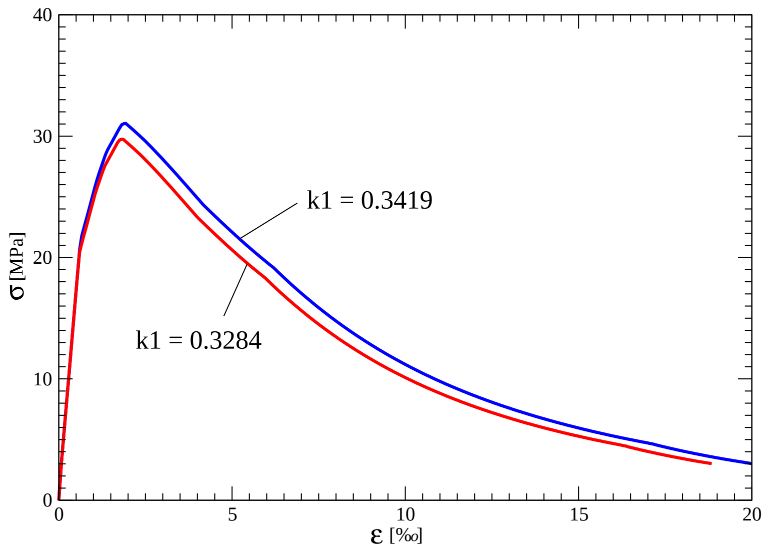


Figure 5.8: The influence of the error in the estimation of the parameter $K_1$ using the neural network.

| parameter | Avg. error [%] | Std. [%] |
|---|---|---|
| $K_1$ | 1,3525 | 0,2962 |
| $C_{20}$ | 6,9758 | 0,8216 |

Table 5.2: An error in estimating the microplane model parameters employing the neural network using the *combined approximation*.

$6.9758\%$ which is the average error for this parameter.

For comparison, the computation was made also for the approximation using the polynomial function of the eighth degree. The nine parameters of the polynomial function for the corresponding strain-stress curve were used as an input vector for the neural network. The results of this computation are shown in Tab. 5.3 in the same format as in Tab. 5.2.

| parameter | Avg. error [%] | Std. [%] |
|---|---|---|
| $K_1$ | 1,2480 | 0,2458 |
| $C_{20}$ | 20,875 | 2,2510 |

Table 5.3: The errors in the estimation of the microplane model parameters employing the polynomial approximation.

### 5.3.2 *Direct application of the parallel* SADE *algorithm*

In the previous section, the microplane models parameters were predicted from already known stress-strain curves. But the whole problem can be seen as minimizing the difference between an experiment and an output from static analysis. Therefore, the objective is to minimize the

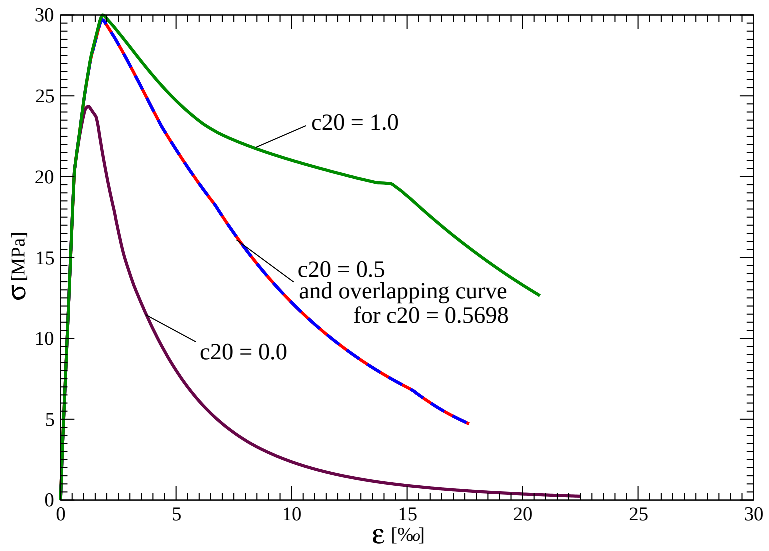


Figure 5.9: The influence of the error in the estimation of the parameter $C_{20}$ using the neural network.

least square error function, which contains the difference between values of a known stress-strain curve (from an experiment or from structural analysis) and values from the microplane model simulation.

As was mentioned previously, we will use only one microplane element, but still, the total computational time is not negligible. Results for one element non-linear analysis on different types of single processor PCs are presented in Table 5.4. The computations were executed one hundred times with randomly chosen material parameters and minimum, average and maximum values are shown. Note the big differences between minimum and maximum times, which result from the non-linear response of the structure.

| Processor | Min. [s] | Avg. [s] | Max. [s] |
|---|---|---|---|
| Intel Pentium III 450 MHz | 47.96 | 89.76 | 213.32 |
| Intel Pentium III 1000 MHz | 22.45 | 47.95 | 97.73 |
| Intel Xeon 1700 MHz | 16.80 | 38.63 | 87.81 |

Table 5.4: Obtained times of one computation on different processors.

The whole problem is therefore single-objective, but is very multi-modal, i.e. there are many local minima with approximately the same profit (see also Section 2.2.4). Figure 5.10 shows an example of the “real” stress-strain curve from an experiment, and its five, locally optimal, approximations with a microplane model. These results were obtained using the **SADE** algorithm after $3,000$ evaluations of the optimization method. The application of the stochastic global optimization method brings here new possibilities as the optimized function apparently possesses several (at least five) local minima, which introduces considerable obstacles to successful application of gradient-based optimization procedures.

When inspecting Tab. 5.4, it is clear that running analysis $3,000$ times on the fastest processor will require (in average) more than $32$ hours of computing. Therefore, we have applied the

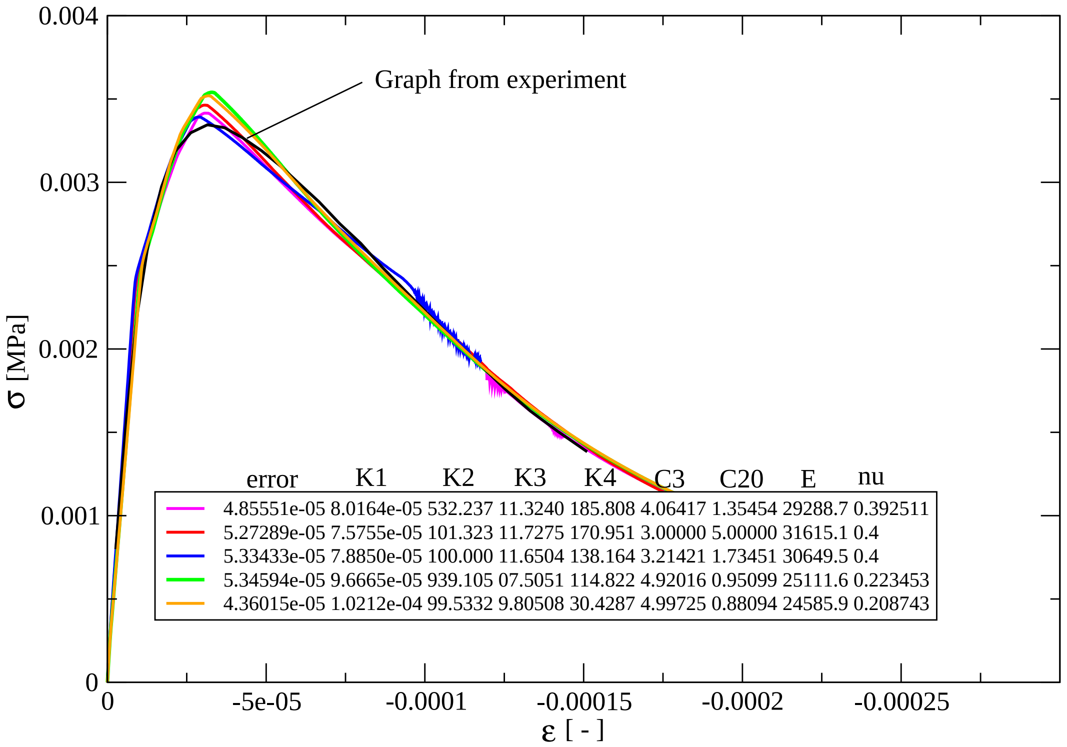


Figure 5.10: Five different results and an original stress-strain curve.

**global parallel model**, see Section 2.2.5, to minimize computational demands. This parallel computing scheme ensures constant distribution of the work among the processors provided that the time spent on evaluation of two solutions does not differ. Although this condition is not strictly met for the current problem (see Table 5.4), it appears that the sufficiently high number of solutions assigned to each processor eliminates this disadvantage.

To be sure that the parallel algorithm is well-designed with respect to the number of available processors, the optimum amount of processors is checked. It can be simply estimated by Eq. (2.16)

$$P^* = \sqrt{\frac{nT_f}{T_c}} \doteq 440 \quad \text{processors} \tag{5.10}$$

where

| | |
|---|---|
| $P^*$ | is the optimal number of processors, |
| $n$ | is the number of solutions in population, in our case $80$, |
| $T_f$ | is time for one evaluation of a function, set to $47.95$ s, see Table 5.4, |
| $T_c$ | is latency time - hardware dependent variable, which is spent on creating communication between two processors, in our case equal to $20$ ms. |

It is clear that in this particular case, the linear speedup can be expected even for substantially higher number of processors than it is available at the author's research department. Therefore, the obtained near-linear speedup is nothing surprising (see Fig. 5.11).

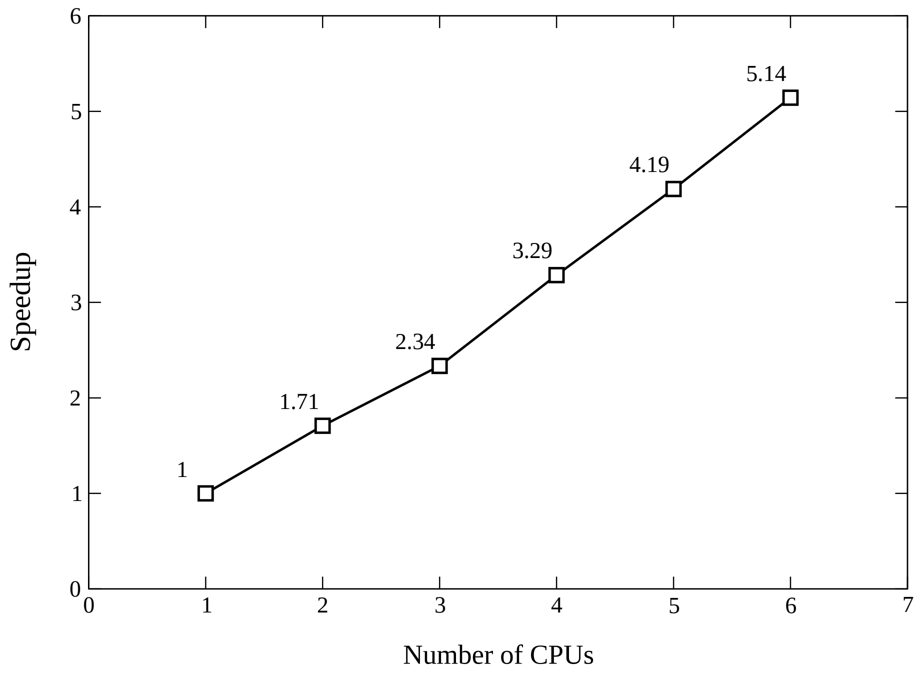


Figure 5.11: Speedup of the parallel SADE algorithm on a cluster of PCs.

Resulting times were obtained by running the presented optimization problem on a PC cluster, installed at the Department of Structural Mechanics, Faculty of Civil Engineering, CTU in Prague. The PC cluster consists of ten two-processor DELL workstations. Parameters of the three used computers (the fastest ones) are described in Table 5.5. The workstations are connected by Fast Ethernet 100 Mb network using 3Com Superstack II switch. Note that this cluster represents a heterogeneous parallel computing platform.

| Processor | No. processors | Memory [MB] |
|---|---|---|
| Intel Pentium III 1000 MHz | 2 | 512 |
| Intel Pentium III 1000 MHz | 2 | 2048 |
| Intel Xeon 1700 MHz | 2 | 1024 |

Table 5.5: Parameters of used computers.

The total times for the present problem are summarized in Fig. 5.12. The results were obtained for 880 evaluations (10 generations per 80 solutions, the first generation needs twice more data). It is obvious that for obtaining useful results in a reasonable time, the number of processors needs to be much higher.

### 5.3.3 *Application of the Latin Hypercube Sampling method (LHS)*

To enrich the previous work on this subject, a neural network presented in Section 5.2.1 is enhanced by the application of the Latin Hypercube Sampling method [Iman and Conover, 1980], which is used to generate training sets for a neural network. This procedure enables to minimize an amount of needed simulations to reliably train a neural network, see e.g. [Lehký, 2003] or

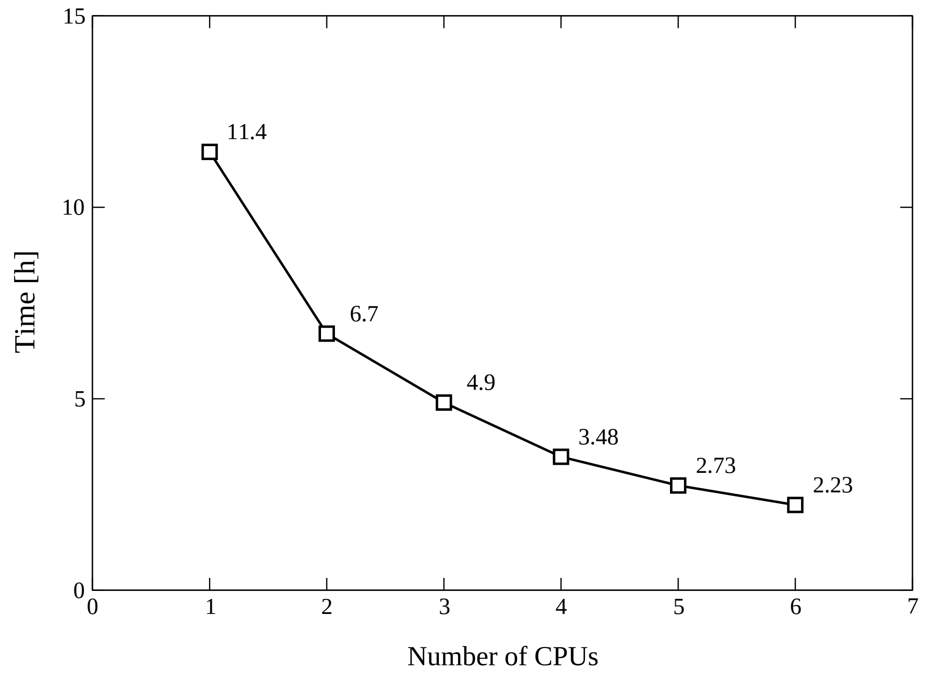


Figure 5.12: Obtained times during $10$ generations of the parallel SADE algorithm.

[Lehký and Novák, 2004]. Moreover, the Simulated Annealing optimization method (see also Section 2.2.1) available in the software package FREET [Novák et al., 2003], or the most recent work [Vořechovský, 2004, Appendix A], is used to ensure the independence among individual samples. As results from this procedure, outputs from the stochastic non-linear analysis can be seen in the Figure 5.13.

At this point, the neural network presented earlier in this paper is trained by the optimization strategy **SADE** on the thirty simulations from the LHS method. The level of trained accuracy is tested on the set of ten different independent curves. The precision of predicted parameters is shown in Table 5.6. The stress-strain curves generated from these predicted parameters are depicted in the Figs. 5.14 and 5.15 and from the same pictures they can be easily compared with the corresponding original stress-strain diagrams.

## *5.4 Conclusions*

The test results have shown that **SADE** algorithm-lead training is a method fully capable to train a neural network. The number of iterations needed to achieve the same output error is significantly lower than with the Backpropagation method. Also the minimal output error is by about three orders lower, which could be explained by the Evolutionary Algorithm's higher resistance to falling into local extremes.

Section 5.3 and the later sections demonstrate the neural network utilization in one part within the Civil Engineering. The results of computations show that the neural network trained by the **SADE** algorithm has the ability to predict the microplane model parameters with a satisfying level of precision. The parameters of the strain-stress curve approximation were chosen as the input data for the prediction. The *combined approximation* by three linear functions and one exponential seems to be an effective way. In the engineering practice, a neural network could

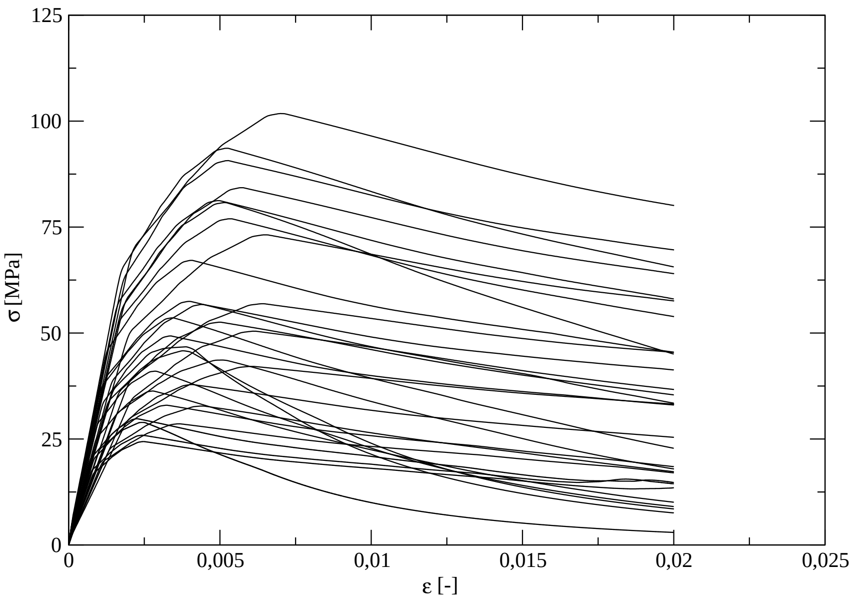


Figure 5.13: Stress-strain curves as results from 30 simulations generated by the Latin Hypercube Sampling method.

save a lot of experimenter's time needed for searching parameters of the microplane model by the trial and error method. It is supposed that when using the neural network for a parameter prediction of a real material and its measured strain-stress curve, this curve must fit the set of modeled examples on which the network was trained.

The main outcome of the present study, however, is the conclusion that the determination of the microplane model parameters needs at least two test cases rather than a sole uniaxial compression test. Indeed, this conclusion directly follows from the large scatter of values of Young's modulus $E$ and Poisson's ratio $\nu$ among identified local minima during direct optimization (see the embedded table in the Fig. 5.10). Since these two parameters are usually the only known values in engineering practice, such a difference is not acceptable and additional data must be supplied to reliably classify individual local minima. Therefore, in this optimization problem,

Table 5.6: Errors in the estimated parameters obtained from ten independent tests by the trained neural network.

| parameter | relative error [%] | |
|---|---|---|
| | average | maximal |
| $E$ | 2.84 | 5.30 |
| $\nu$ | 36.33 | 117.94 |
| $k_1$ | 1.70 | 4.93 |
| $k_2$ | 45.82 | 102.78 |
| $k_3$ | 36.51 | 224.68 |
| $k_4$ | 64.08 | 153.22 |
| $c_3$ | 33.74 | 78.68 |
| $c_{20}$ | 22.27 | 38.00 |

the application of the optimization procedure does produce not only (sub)-optimal values of optimized parameters but also provides deeper insight into the analyzed problem.

The parallel solution then appears to be an appropriate tool for tackling the enormous computational demand of the microplane material model. The obtained nearly-linear speedup together with the possibility to use many more processors promise new interesting results and potential applications of the presented method in the future.

When using a neural network, a number of needed simulations can be reduced by the application of the Latin Hypercube Sampling method. The sensitivity analysis shows not only the influence of individual parameters, see Figure 5.6, but also approximately predicts the errors produced by the neural network, see Table 5.6. Although the obtained predictions are not identical, they can be further improved by much longer training process and/or by changing the topology of a neural network. Nevertheless, the main advantage of this approach can be still employed - the trained neural network can be used for the next estimation phase in the future without the need of expensive numerical simulations.

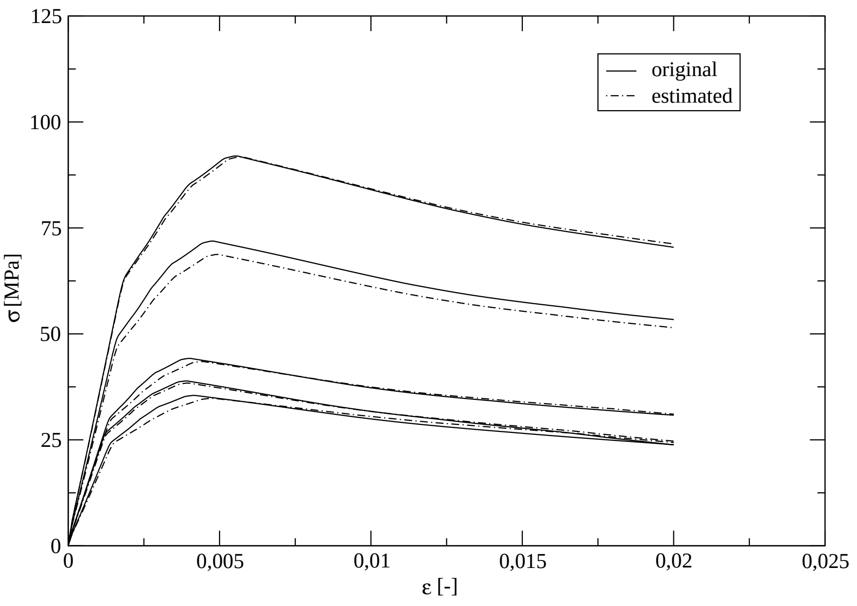


Figure 5.14: A comparison of the first five stress-strain diagrams for the original and for the estimated parameters.

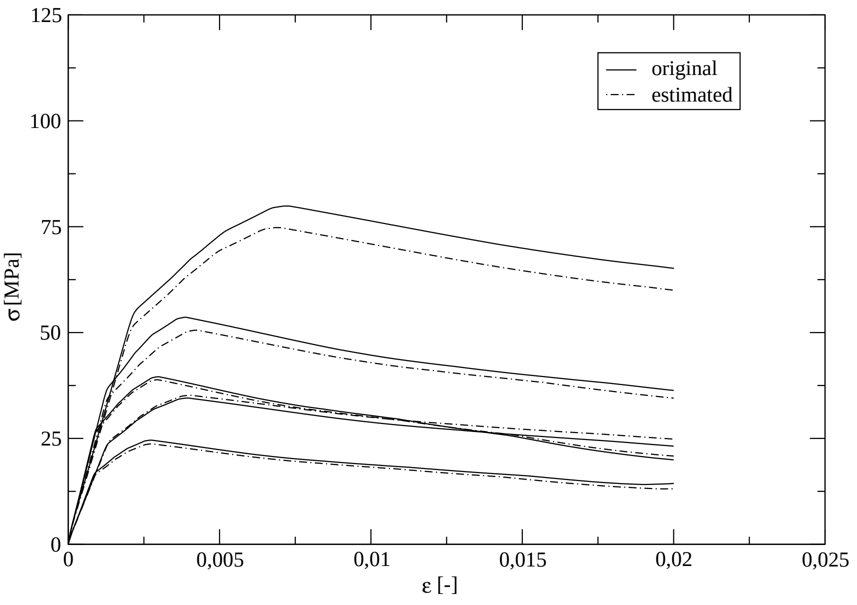


Figure 5.15: A comparison of the second five stress-strain diagrams for the original and for the estimated parameters.

Chapter 6

# CONCLUSIONS AND FUTURE PERSPECTIVES

Nothing is older than yesterday's success.

Robert M. Knight

The proposed thesis brings an insight into global optimization methods applied to several Civil Engineering tasks. To describe problems, which are usually encountered in engineering practice as well as science, basic notation and classification is introduced. Namely, the Global and Structural optimizations are presented and the latter one is divided into four categories, which, hopefully, cover all structural optimization tasks from the Civil Engineering area.

Next, a new classification for Evolutionary Algorithms (**EA**s) is presented. It is based on the well-known notation developed for the Evolution Strategies (**ES**s) [Bäck and Schwefel, 1995] and appropriately modified for single- as well as multi-objective optimization algorithms. The leading idea is that every **EA** can be described by a combination of three basic operations, namely recombination, mutation and selection mechanisms. Based on this notation, the most popular algorithms from the Global optimization area are introduced and described.

In engineering practice, we usually deal with multi-objective, constrained and often mixed integer-continuous optimization problems. Solutions for all these phenomena are presented: the multi-objective nature can be solved by Pareto-optimality approaches, constraints by penalty functions and different types of variables by an appropriate encoding. Several other possibilities are discussed in the text as well.

Although research within the Evolutionary Algorithms domain seems to be almost finished, still there are several phenomena that need to be studied. Recently, multi-objective algorithms have drawn a lot of attention and new developments in this area can be expected. Also in the domain of the Global optimization, new discoveries have been made. Especially, the No free lunch theorem changed dramatically the view on optimization algorithms. Now comes the question if it is possible to find a superior algorithm for one particular function. The answer for all optimization algorithms is "Not", see e.g. [Macready and Wolpert, 1995], but if the number of available algorithms is limited, then the answer is still missing.

Then, based on the above mentioned notation, four particular examples of **EA**s are described and compared. These optimization algorithms are used to solve several tasks from engineering practice as well as two test functions and advantages and disadvantages of these methods are shown. As a result, the **SADE** algorithm is recommended due to its simplicity and a small number of parameters.

The next part is devoted to the application of the presented optimization methods to the design of reinforced concrete frames. Generally, this task is multi-modal, multi-objective and highly constrained. To solve this problem as a whole, it is shown that this inevitably leads to an integer formulation of the problem and hence presented qualities of Evolutionary Algorithms are utilized. As an illustrative result, typical examples are solved and the Pareto-fronts in terms of the total price of a structure against its deflection are depicted. A new system of visualization is also presented as an addition to the multi-objective optimization domain.

As an example of a single-objective optimization problem, training of an artificial neural network and its use for a microplane material model parameters prediction is presented. The Evolutionary Algorithm can be used here for the same purpose and has shown that the obtained errors are much lower than the outputs obtained from the Backpropagation algorithm. Moreover, an identification of the microplane material model is investigated. Several approaches are tested to solve the introduced problem. An estimation by an artificial neural network trained on approximations of stress-strain curves shows that some properties can be predicted well but a significant error in other coefficients is obtained. One reason can be an inappropriate description of a load-deflection curve. A new possibility here is to use Genetic Programming methods [Koza, 1992]. As was shown in [Rudnicki et al., 2003], it is nowadays possible to find the most suitable approximation even as an analytical expression in a closed form[1].

Next, a parallel version of the **SADE** algorithm is directly used to obtain the required parameters. The first main result is that this time consuming analysis can be solved by a parallel analysis in a reasonable time. The second outcome is the fact that the objective function corresponding to the identification problem has several local minima, which are characterized by similar values but are far from each other. This is usually solved by niching methods which have still not been investigated enough. The opposite is true for a general parallelization, which is nowadays no more a research topic, but an everyday approach. Today's most promising area is the domain of approximate models, where either a simpler computational tool can be used, e.g. a static instead of non-linear static or non-linear static instead of non-linear dynamic analysis can be utilized, etc., or a domain of a general approximation, usually called as Response Surface methods [Lee and Hajela, 2001], Diffuse Approximations [Ibrahimbegović et al., 2004] or Surrogate models [Karakasis and Giannakoglou, 2004]. If such a model exists, then different and usually a hybrid parallelization scheme can be effectively applied, see e.g. [Wang et al., 2002] and [González et al., 2004]. Moreover, traveling between an approximate and original functions can be also seen as a multi-objective problem. One recent example of such application can be found e.g. in [Quagliarella, 2003].

To solve the above mentioned obstacles in microplane parameters identification and in the view of recent research in this domain, a new methodology is also presented: an application of the Latin Hypercube Sampling method (LHS) as well as a sensitivity analysis are applied not only to investigate the influence of individual material model parameters, but also to minimize the need of training samples for an artificial neural network. This application of the LHS method along with the Simulated Annealing seems to be too robust, because it is primarily aimed at fulfilling correlation dependencies rather than independence of training samples, see e.g. [Vořechovský et al., 2002] or [Vořechovský, 2004]. This can be done by simpler methods, for instance, the so-called Quasi-random generators seem to be a proper choice. Recently two authors have been investigating these methods - Heikki Maaranen has compared several Quasi-random generators and their influence on the behavior of Genetic Algorithms, see the works [Maaranen et al., 2004a] and [Maaranen et al., 2004b], and Felix F. Tong has studied the same topic directly within neural network training [Tong and Liu, 2004] and both authors report interesting and promising results.

[1] By the way, the Genetic Programming can be also seen as a multi-objective problem, see e.g. the work [Bleuler et al., 2001] for more details.

# BIBLIOGRAPHY

[Abdallaa et al., 1996] Abdallaa, K. M., Alshegeirb, A., and Chenc, W. F. (1996). Analysis and design of mushroom slabs with a strut-tie model. *Computers & Structures*, 58(2):429–434.

[Adeli and Kamal, 1986] Adeli, H. and Kamal, O. (1986). Efficient optimization of space trusses. *Computers & Structures*, 24(3):501–511.

[Adleman, 1994] Adleman, L. M. (1994). Molecular computation of solutions to combinatorial problems. *Science*, 266:1021–1024.

[Alander, 1995] Alander, J. T. (1995). Indexed bibliography of distributed genetic algorithms. Report 94-1-PARA, University of Vaasa, Department of Information Technology and Production Economics. Available at `ftp://ftp.uwasa.fi/cs/report94-1/-gaPARAbib.ps.Z`.

[Allaire, 2002] Allaire, G. (2002). *Shape Optimization by the Homogenization Method*. Springer-Verlag, New-York Berlin Heidelberg.

[Andre et al., 2000] Andre, J., Siarry, P., and Dognon, T. (2000). An improvement of the standard genetic algorithm fighting premature convergence in continuous optimization. *Advances in Engineering Software*, 32(1):49–60.

[Annicchiarico, 2003] Annicchiarico, W. D. (2003). Three dimensional shape optimization of complex model by using distributed micro genetic algorithms. In [Bugeda et al., 2003].

[ASCE-ACI, 1998] ASCE-ACI (1998). ASCE-ACI Committee 445 on Shear and Torsion. Recent approaches to shear design of structural concrete. *Journal of Structural Engineering*, 124(12):1375–1417.

[Bäck and Schwefel, 1995] Bäck, T. and Schwefel, H.-P. (1995). Evolution Strategies I: Variants and their computational implementation. In Periaux and Winter, editors, *Genetic Algorithms in Engineering and Computer Science*, chapter 6, pages 111–126. John Wiley & Sons Ltd.

[Bauer and Gutkowski, 1995] Bauer, J. and Gutkowski, W. (1995). Discrete structural optimization: A review. In [Olhoff and Rozvany, 1995], pages 901–908.

[Bažant et al., 2000] Bažant, Z. P., Caner, F. C., Carol, I., Adley, M. D., and Akers, S. A. (September 2000). Microplane model M4 for concrete. Part I: Formulation with work-conjugate deviatoric stress. *Journal of Engineering Mechanics*, 126(9):944–953.

[Belegundu, 1982] Belegundu, A. D. (1982). *A Study of Mathematical Programming Methods for Structural Optimization*. PhD thesis, University of Iowa, Dept. of Civil and Environmental Engineering.

[Bendsøe and Sigmund, 2003] Bendsøe, M. P. and Sigmund, O. (2003). *Topology Optimization: Theory, Methods and Applications*. Springer-Verlag.

[Bleuler et al., 2001] Bleuler, S., Brack, M., Thiele, L., and Zitzler, E. (2001). Multiobjective Genetic Programming: Reducing Bloat Using SPEA2. In *Proceedings of the Congress on Evolutionary Computation 2001 (CEC'2001)*, volume 1, pages 536–543, Piscataway, New Jersey. IEEE Service Center.

[Bonet et al., 2002] Bonet, J. L., Miguel, P. F., Romero, M. L., and Fernandez, M. A. (2002). A modified algorithm for reinforced concrete cross section integration. In Topping, B. H. V. and Bittnar, Z., editors, *Proceedings of the Sixth International Conference on Computational Structures Technology*, Stirling, United Kingdom. Civil-Comp Press.

[Bugeda et al., 2003] Bugeda, G., Désidéri, J.-A., Périaux, J., Schoenauer, M., and Winter, G., editors (2003). *Evolutionary Methods for Design, Optimization and Control: Applications to Industrial and Societal Problems*, Eurogen 2003. International Center for Numerical Methods in Engineering (CIMNE).

[Caner and Bažant, 2000] Caner, F. C. and Bažant, Z. P. (September 2000). Microplane model M4 for concrete. Part II: Algorithm and calibration. *Journal of Engineering Mechanics*, 126(9):954–961.

[Cantú-Paz, 1997] Cantú-Paz, E. (1997). A survey of parallel genetic algorithms (Illi-GAL Report No. 97003). Technical report, Urbana, IL: University of Illinois at Urbana-Champaign.

[Cantú-Paz, 2001] Cantú-Paz, E. (2001). *Efficient and Accurate Parallel Genetic Algorithms*. Kluwer Academic Publishers.

[Ceranic et al., 2001] Ceranic, B., Fryer, C., and Baines, R. W. (2001). An application of simulated annealing to the optimum design of reinforced concrete retaining structures. *Computers & Structures*, 79:1569–1581.

[Charalampakis and Koumousis, 2004] Charalampakis, A. and Koumousis, V. (2004). A generic fiber model algorithm for the analysis of arbitrary cross sections under biaxial bending and axial load. In Topping, B. and Soares, C. M., editors, *Proceedings of the Seventh International Conference on Computational Structures Technology*. Civil-Comp Press, Stirling, Scotland.

[Chen et al., 2001] Chen, S. F., Teng, J. G., and Chan, S. L. (June 2001). Design of biaxially loaded short composite columns of arbitrary section. *Journal of Structural Engineering*, 127(6):678–685.

[Cherkaev, 2000] Cherkaev, A. V. (2000). *Variational Methods for Structural Optimization*. Springer-Verlag, New-York Berlin Heidelberg.

[Chiba et al., 2003] Chiba, K., Obayashi, S., Nakahashi, K., Giotis, A. P., and Giannakoglou, K. C. (2003). Design optimization of the wing shape for the RLV booster stage using evolutionary algorithms and Navier-Stokes computations on unstructured grids. In [Bugeda et al., 2003].

[Choi and Kwak, 1990] Choi, C.-K. and Kwak, H.-G. (1990). Optimum RC member design with predetermined discrete sections. *Journal of Structural Engineering*, 116(10):2634–2655.

[Coello, 2000a] Coello, C. A. C. (2000a). Constraint-handling using an evolutionary multiobjective optimization technique. *Civil Engineering and Environmental Systems*, 17:319–346.

[Coello, 2000b] Coello, C. A. C. (2000b). Handling Preferences in Evolutionary Multiobjective Optimization: A Survey. In *2000 Congress on Evolutionary Computation*, volume 1, pages 30–37, Piscataway, New Jersey. IEEE Service Center.

[Coello, 2000c] Coello, C. A. C. (2000c). Treating Constraints as Objectives for Single-Objective Evolutionary Optimization. *Engineering Optimization*, 32(3):275–308.

[Coello, 2003] Coello, C. A. C. (2003). Evolutionary Multiobjective Optimization: Current and Future Challenges. In Benitez, J., Cordon, O., Hoffmann, F., and Roy, R., editors, *Advances in Soft Computing—Engineering, Design and Manufacturing*, pages 243–256. Springer-Verlag.

[Coello, 2004] Coello, C. A. C. (2004). List of references on evolutionary multiobjective optimization. `http://www.lania.mx/~ccoello/EMOO/EMOObib.html`.

[Coello and Pulido, 2001] Coello, C. A. C. and Pulido, G. T. (2001). Multiobjective Optimization using a Micro-Genetic Algorithm. In Spector, L., Goodman, E. D., Wu, A., Langdon, W., Voigt, H.-M., Gen, M., Sen, S., Dorigo, M., Pezeshk, S., Garzon, M. H., and Burke, E., editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001)*, pages 274–282, San Francisco, California. Morgan Kaufmann Publishers.

[Coello et al., 1997] Coello, C. C., Hernández, F. S., and Farrera, F. A. (1997). Optimal design of reinforced concrete beams using genetic algorithms. *Expert Systems with Applications*, 12(1):101–108.

[Corne and Knowles, 2003] Corne, D. and Knowles, J. (2003). No free lunch and free leftovers theorems for multiobjective optimization. In *Proceedings of the Second International Conference on Evolutionary Multi-Criterion Optimization (EMO 2003), Faro, Portugal*, pages 327–341.

[Deb, 1999] Deb, K. (1999). Evolutionary algorithms for multi-criterion optimization in engineering design. In [Miettinen et al., 1999], pages 135–161.

[Deb et al., 2000] Deb, K., Agrawal, S., Pratab, A., and Meyarivan, T. (2000). A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. In Schoenauer, M., Deb, K., Rudolph, G., Yao, X., Lutton, E., Merelo, J. J., and Schwefel, H.-P., editors, *Proceedings of the Parallel Problem Solving from Nature VI Conference*, pages 849–858, Paris, France. Springer. Lecture Notes in Computer Science No. 1917.

[Drchal et al., 2002] Drchal, J., Kučerová, A., and Němeček, J. (2002). Optimizing synaptic weights of neural networks. In Topping, B. and Bittnar, Z., editors, *Proceedings of the Third International Conference on Engineering Computational Technology*, Stirling, United Kingdom. Civil-Comp Press.

[Eiben et al., 1999] Eiben, A. E., Hintering, R., and Michalewicz, Z. (1999). Parameter control in evolutionary algorithms. *IEEE Transations on Evolutionary Computation*, 3(2):124–141.

[Eurocode 2, 1991] Eurocode 2 (1991). *Eurocode 2 Part 1.1, Design of Concrete Structures, ENV 1992 1-1*. CEN, Brussels.

[Fafitis, 2001] Fafitis, A. (July 2001). Interaction surfaces of reinforced-concrete sections in biaxial bending. *Journal of Structural Engineering*, 127(7):840–846.

[Fan et al., 2000] Fan, H.-Y., Lu, J. W.-Z., and Xu, Z.-B. (2000). An empirical comparison of three novel genetic algorithms. *Engineering Computation*, 17(8):981–1001.

[Fogel, 1999] Fogel, D. B. (1999). Some recent important foundational results in evolutionary computation. In [Miettinen et al., 1999], pages 55–71.

[Fonseca and Fleming, 1993] Fonseca, C. M. and Fleming, P. J. (1993). Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In Forrest, S., editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 416–423, San Mateo, California. University of Illinois at Urbana-Champaign, Morgan Kauffman Publishers.

[Gil and Andreu, 2001] Gil, L. and Andreu, A. (2001). Shape and cross-section optimisation of a truss structure. *Computers & Structures*, 79:681–689.

[Goldberg, 1989] Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley.

[González et al., 2004] González, L. F., Whitney, E. J., Périaux, J., Sefrioui, M., and Srinivas, K. (2004). Multidisciplinary aircraft conceptual design and optimisation using a robust evolutionary technique. In [Neittaanmäki et al., 2004].

[Goslingt and Lewist, 1996] Goslingt, P. D. and Lewist, W. J. (1996). Optimal structural membranes-II. Form-finding of prestressed membranes using a curved quadrilateral finite element for surface definition. *Computers & Structures*, 61(5):895–895.

[Greiner et al., 2001] Greiner, D., Winter, G., and Emperador, J. M. (2001). Optimising frame structures by different strategies of genetic algorithms. *Finite Elements in Analysis and Design*, 37:381–402.

[Greiner et al., 2003] Greiner, D., Winter, G., and Emperador, J. M. (2003). Searching for an efficient method in multiobjective frame optimisation using evolutionary algorithms. In Bathe, K. J., editor, *Second MIT Conference on Computational Fluid and Solid Mechanics*, volume 2, pages 2285–2290, Oxford, UK. Elsevier Ltd.

[Hadi, 2001] Hadi, M. N. S. (2001). Optimum design of reinforced concrete continuous beams by genetic algorithms. In Topping, B. H. V., editor, *Proceedings of the Eighth International Conference on Civil and Structural Engineering Computing*, Stirling, United Kingdom. Civil-Comp Press.

[Harik and Lobo, 1999] Harik, G. R. and Lobo, F. G. (1999). A parameter-less genetic algorithm. In Banzhaf, W., Daida, J., Eiben, A. E., Garzon, M. H., Honavar, V., Jakiela, M., and Smith, R. E., editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 1, pages 258–265, Orlando, Florida, USA. Morgan Kaufmann.

[Haslinger and Neittaanmaki, 1996] Haslinger, J. and Neittaanmaki, P. (1996). *Finite element approximation for optimal shape, material and topology design*. John Wiley & Sons.

[Hinterding et al., 1997] Hinterding, R., Michalewicz, Z., and Eiben, A. (1997). Adaptation in evolutionary computation: A survey. In *Proceedings of the 4th IEEE International Conference on Evolutionary Computation, Indianapolis*, pages 65–69.

[Holland, 1975] Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. MIT Press.

[Holmes, 2001] Holmes, P. (2001). Correlation: From picture to formula. *Teching Statistics*, 23(3).

[Horn and Nafpliotis, 1993] Horn, J. and Nafpliotis, N. (1993). Multiobjective Optimization using the Niched Pareto Genetic Algorithm. Technical Report IlliGAl Report 93005, University of Illinois at Urbana-Champaign, Urbana, Illinois, USA.

[Houck et al., 1995] Houck, C. R., Joines, J. A., and Kay, M. G. (1995). A genetic algorithm for function optimization: A Matlab implementation. Technical Report NCSU-IE TR 95-09, North Carolina State University.

[Hrstka, WWWa] Hrstka, O. (WWWa). Comparing different types of evolutionary methods from the point of view of robustness and convergence rate.
`http://klobouk.fsv.cvut.cz/~ondra/about_ga`.

[Hrstka, WWWb] Hrstka, O. (WWWb). Homepage of SADE.
`http://klobouk.fsv.cvut.cz/~ondra/sade/sade.html`.

[Hrstka and Kučerová, 2000] Hrstka, O. and Kučerová, A. (2000). Searching for optimization method on multidimensional real domains. In *Contributions to Mechanics of Materials and Structures*, volume 4 of *CTU Reports*, pages 87–104. Czech Technical University in Prague.

[Hrstka and Kučerová, 2004] Hrstka, O. and Kučerová, A. (2004). Improvements of real coded genetic algorithms based on differential operators preventing the premature convergence. *Advances in Engineering Software*, 35(3–4):237–246.

[Hrstka et al., 2001] Hrstka, O., Kučerová, A., Lepš, M., and Zeman, J. (2001). A competitive comparison of different types of evolutionary algorithms. In *Proceedings of the Sixth International Conference of Artificial Intelligence to Civil and Structural Engineering*, pages on CD–ROM, Eisenstadt, Austria. Civil-Comp Press.

[Hrstka et al., 2003] Hrstka, O., Kučerová, A., Lepš, M., and Zeman, J. (2003). A competitive comparison of different types of evolutionary algorithms. *Computers & Structures*, 81(18–19):1979–1990.

[Ibrahimbegović et al., 2004] Ibrahimbegović, A., Knopf-Lenoir, C., Kučerová, A., and Villon, P. (2004). Optimal design and optimal control of elastic structures undergoing finite rotations. *International Journal for Numerical Methods in Engineering*. In print.

[Iman and Conover, 1980] Iman, R. L. and Conover, W. (1980). Small sample sensitivity analysis techniques for computer models, with an application to risk assessment. *Communications in Statistics - Theory and Methods*, 9(17):1749–1842.

[Ingber, 1993] Ingber, L. (1993). Simulated annealing: Practice versus theory. *Mathematical and Computer Modelling*, 18(11):29–57.

[Ingber, 1995] Ingber, L. (1995). Adaptive simulated annealing (ASA): Lessons learned. *Control and Cybernetics*, 25(1):33–54.

[Jirásek and Bažant., 2001] Jirásek, M. E. and Bažant., Z. P. (2001). *Inelastic Analysis of Structures*. John Wiley & Sons.

[Kameshki and Saka, 2001] Kameshki, E. S. and Saka, M. P. (2001). Optimum design of nonlinear steel frames with semi-rigid connections using a genetic algorithm. *Computers & Structures*, 79:1593–1604.

[Karakasis and Giannakoglou, 2004] Karakasis, M. K. and Giannakoglou, K. C. (2004). On the use of surrogate evaluation models in multi-objective evolutionary algorithms. In [Neittaanmäki et al., 2004].

[Kim and Baker, 2002] Kim, H. and Baker, G. (2002). Topology optimization for reinforced concrete design. In [Mang et al., 2002].

[Kirsch, 1995] Kirsch, U. (1995). Layout optimization using reduction and expansion processes. In [Olhoff and Rozvany, 1995], pages 95–102.

[Knowles and Corne, 2000] Knowles, J. D. and Corne, D. W. (2000). Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. *Evolutionary Computation*, 8(2):149–172.

[Knowles et al., 2001] Knowles, J. D., Watson, R. A., and Corne, D. W. (2001). Reducing Local Optima in Single-Objective Problems by Multi-objectivization. In Zitzler, E., Deb, K., Thiele, L., Coello, C. A. C., and Corne, D., editors, *First International Conference on Evolutionary Multi-Criterion Optimization*, pages 268–282. Springer-Verlag. Lecture Notes in Computer Science No. 1993.

[Koumousis et al., 1996] Koumousis, V. K., Arsenis, S. J., and Vasiloglou, V. B. (1996). Detailed design of reinforced concrete buildings using logic programming. *Advances in Engineering Software*, 25:161–176.

[Koza, 1992] Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press.

[Kukkonen and Lampinen, 2004] Kukkonen, S. and Lampinen, J. (2004). Comparison of generalized differential evolution to other multi-objective evolutionary algorithms. In [Neittaanmäki et al., 2004].

[Kučerová et al., 2003] Kučerová, A., Lepš, M., and Němeček, J. (2003). Estimation of microplane model parameters using parallel genetic algorithm. In Topping, B. H. V., editor, *Proceedings of The Seventh International Conference on the Application of Artificial Intelligence to Civil and Structural Engineering*, pages 87–88, Stirling, United Kingdom. Civil-Comp Press.

[Lagaros et al., 2002] Lagaros, N. D., Papadrakakis, M., and Kokossalakis, G. (2002). Structural optimization using evolutionary algorithms. *Computers & Structures*, 80:571–589.

[Laníková, 1999] Laníková, I. (1999). Optimalizovaný návrh výztuže rámových konstrukcí [An optimum design of reinforcement within frame structures]. In *Concrete Days '99*, pages 272–279. The Czech Concrete Society.

[Lee and Hajela, 2001] Lee, J. and Hajela, P. (2001). Application of classifier systems in improving response surface based approximations for design optimization. *Computers & Structures*, 79:333–344.

[Lehký, 2003] Lehký, D. (2003). Material parameters indentification for concrete failure modeling. In Vrba, J., Keršner, Z., and Frantík, P., editors, *2nd PhD Workshop Brno-Wiemar-Prague-Wien$_{BOKU}$, Book of abstracts*, page 20. Brno University of Technology, Czech Republic.

[Lehký and Novák, 2004] Lehký, D. and Novák, D. (2004). Identification of material model parameters using stochastic training of neural network. In *Proceedings of 5th International PhD Symposium in Civil Engineering*, Delft, Netherlands.

[Lepš and Bittnar, 2001] Lepš, M. and Bittnar, Z. (2001). Optimization of RC beams using genetic algorithm. In Atluri, S. N., Nishioka, T., and Kikuchi, M., editors, *Advances in Computational Engineering & Sciences*, pages on CD–ROM. Tech Science Press.

[Lepš et al., 1999] Lepš, M., Matouš, K., and Bittnar, Z. (1999). Genetic algorithms in optimization of reinforced concrete beam. *Acta Polytechnica*, 39(2):145–154.

[Lepš and Šejnoha, 2000] Lepš, M. and Šejnoha, M. (2000). New approach to optimization of reinforced concrete beams. In *Computational Concrete Structures Technology*, pages 143–151, Leuven. Civil-Comp Press.

[Lepš and Šejnoha, 2003] Lepš, M. and Šejnoha, M. (2003). New approach to optimization of reinforced concrete beams. *Computers & Structures*, 81(18–19):1957–1966.

[Lepš et al., 2002] Lepš, M., Zeman, J., and Bittnar, Z. (2002). Hybrid optimization approach to design of reinforced concrete frames. In Topping, B. H. V. and Bittnar, Z., editors, *Proceedings of The Third International Conference on Engineering Computational Technology*, pages 177–178, Prague, Czech Republic. Civil-Comp Press.

[Liang et al., 1999] Liang, Q. Q., Xie, Y. M., and Steven, G. P. (1999). Optimal strut-and-tie models in structural concrete members. In Topping, B. H. V. and Kumar, B., editors, *Optimization and Control in Civil and Structural Engineering*, pages 1–8. Civil-Comp Limited.

[Maaranen et al., 2004a] Maaranen, H., Miettinen, K., and Mäkelä, M. M. (2004a). Quasi-random initial population for genetic algorithms. *Computers and Mathematics with Applications*. In print.

[Maaranen et al., 2004b] Maaranen, H., Miettinen, K., and Penttinen, A. (2004b). How to generate an initial population for genetic algorithms. In [Neittaanmäki et al., 2004].

[Macready and Wolpert, 1995] Macready, W. G. and Wolpert, D. H. (1995). What makes an optimization problem hard? Technical Report SFI-TR-95-05-046, Santa Fe, NM.

[Mahfoud and Goldberg, 1995] Mahfoud, S. and Goldberg, D. E. (1995). Parallel recombinative simulated annealing - A genetic algorithm. *Parallel Computing*, 21(1):1–28.

[Mahfoud, 1995a] Mahfoud, S. W. (1995a). A comparison of parallel and sequential niching methods. In Eshelman, L., editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 136–143, San Francisco, CA. Morgan Kaufmann.

[Mahfoud, 1995b] Mahfoud, S. W. (1995b). *Niching methods for genetic algorithms*. PhD thesis, University of Illinois at Urbana-Champaign, Urbana, IL, USA.

[Mang et al., 2002] Mang, H., Rammerstorfer, F., and Eberhardsteiner, J., editors (2002). *Fifth World Congress on Computational Mechanics (WCCM V)*. Vienna University of Technology, Austria, `http://wccm.tuwien.ac.at`

[Marler and Arora, 2003] Marler, R. T. and Arora, J. S. (2003). Review of multi-objective optimization concepts and algorithms for engineering. Technical Report ODL-01.03, University of Iowa, Optimal Design Laboratory.

[Marler and Arora, 2004] Marler, R. T. and Arora, J. S. (2004). Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, 26(6):369–395.

[Matouš et al., 2000] Matouš, K., Lepš, M., Zeman, J., and Šejnoha, M. (2000). Applying genetic algorithms to selected topics commonly encountered in engineering practice. *Computer Methods in Applied Mechanics and Engineering*, 190(13–14):1629–1650.

[Mattheck and Burkhardt, 1990] Mattheck, C. and Burkhardt, S. (1990). A new method for structural shape optimisation based on biological growth. *International Journal of Fatigue*, 12(3):185–190.

[Michalewicz, 1999] Michalewicz, Z. (1999). *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, $3^{rd}$ edition.

[Michalewicz et al., 1997] Michalewicz, Z., Hinterding, R., and Michalewicz, M. (1997). Evolutionary algorithms, Chapter 2. In Pedrycz, W., editor, *Fuzzy Evolutionary Computation*. Kluwer Academic.

[Michalewicz et al., 1994] Michalewicz, Z., Logan, T., and Swaminathan, S. (1994). Evolutionary operators for continuous convex parameter spaces. In Sebald, A. and Fogel, L., editors, *Proceedings of the 3rd Annual Conference on Evolutionary Programming*, pages 84–97. World Scientific Publishing, River Edge, NJ.

[Michalewicz and Nazhiyath, 1995] Michalewicz, Z. and Nazhiyath, G. (1995). Genocop III: A co-evolutionary algorithm for numerical optimization problems with nonlinear constraints. In Fogel, D. B., editor, *Proceedings of the Second IEEE International Conference on Evolutionary Computation*, pages 647–651. IEEE Press.

[Miettinen, 1999] Miettinen, K. (1999). *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, Dordrecht.

[Miettinen and Mäkelä, 2004] Miettinen, K. and Mäkelä, M. (2004). NIMBUS: Interactive multiobjective optimization system. `http://nimbus.mit.jyu.fi/`.

[Miettinen et al., 1999] Miettinen, K., Neittaanmäki, P., Mäkelä, M. M., and Périaux, J., editors (1999). *Evolutionary Algorithms in Engineering and Computer Science*. John Wiley & Sons.

[Neittaanmäki et al., 2004] Neittaanmäki, P., Rossi, T., Korotov, S., Oñate, E., Périaux, P., and Knörzer, D., editors (2004). *European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS 2004)*, Jyväskylä.

[NEOS Guide, 1996] NEOS Guide (1996). Optimization Tree, Online guide to the field of numerical optimization. `http://www-fp.mcs.anl.gov/otc/Guide/OptWeb/`.

[Neumaier, 2003] Neumaier, A. (2003). Online archive of information on (almost exclusively non-commercial) global optimization, and somewhat less comprehensive on local optimization. `http://www.mat.univie.ac.at/∼neum/glopt.html`.

[Neumaier, 2004] Neumaier, A. (2004). Complete search in continuous global optimization and constraint satisfaction. In Iserles, A., editor, *Acta Numerica*, pages 271–369. Cambridge University Press.

[Novák et al., 2003] Novák, D., Vořechovský, M., and Rusina, R. (2003). Small-sample probabilistic assessment - software FREET. In *Proceedings of 9th International Conference on Applications of Statistic and Prabability in Civil Engineering - ICASP 9*, pages 91–96, San Francisco, USA. Millpress, Rotterdam.

[Němeček, 2000] Němeček, J. (2000). Modelling of compressive softening of concrete. *CTU Reports*, 4(6). Ph.D. Thesis.

[Němeček et al., 2002] Němeček, J., Patzák, B., Rypl, D., and Bittnar, Z. (2002). Microplane models: Computational aspects and proposed parallel algorithm. *Computers & Structures*, 80(27–30):2099–2108.

[Olhoff, 1996] Olhoff, N. (1996). Multicriterion structural optimization via bound formulation and mathematical programming. *Structural Optimization*, 1:11–17.

[Olhoff and Rozvany, 1995] Olhoff, N. and Rozvany, G. I. N., editors (1995). *First World Congress of Structural and Multidisciplinary Optimization*, Essen University, Essen, Germany.

[Pareto, 1896] Pareto, V. (1896). *Cours D'Economie Politique*, volume 1. Lausanne: F. Rouge.

[Pedersen, 2000] Pedersen, P. (2000). On optimal shapes in materials and structures. *Structural and Multidisciplinary Optimization*, 19:169–182.

[Pincus, 1970] Pincus, M. (1970). A Monte-Carlo method for the approximate solution of certain types of constrained optimization problems. *Operations Research*, 18:1225–1228.

[Quagliarella, 2003] Quagliarella, D. (2003). Airfoil design using Navier-Stokes equations and an asymmetric multi-objective genetic algorithm. In [Bugeda et al., 2003].

[Rafiq and Southcombe, 1998] Rafiq, M. Y. and Southcombe, C. (1998). Genetic algorithms in optimal design and detailing of reinforced concrete biaxial columns supported by a declarative approach for capacity checking. *Computers & Structures*, 69:443–457.

[Rechenberg, 1973] Rechenberg, I. (1973). *Evolution strategy: Optimization of technical systems by means of biological evolution*. Fromman-Holzboog, Stuttgart.

[Ripley, 1977] Ripley, B. (1977). Modelling of spatial patterns. *Journal of the Royal Statistical Society - Series B (methodological)*, 39(2):172–192.

[Rizzo et al., 2000] Rizzo, S., Spallino, R., and Giambanco, G. (2000). Shakedown optimal design of reinforced concrete structures by evolution strategies. *Engineering Computations*, 17(4):440–458.

[Romero et al., 2002] Romero, M. L., Miguel, P. F., and Cano, J. J. (2002). A parallel procedure for nonlinear analysis of reinforced concrete three-dimensional frames. *Computers & Structures*, 80(16–17):1337–1456.

[Rong et al., 2001] Rong, J. H., Xie, Y. M., and Yang, X. Y. (2001). An improved method for evolutionary structural optimisation against buckling. *Computers & Structures*, 79:253–263.

[Rotter, 1985] Rotter, J. M. (1985). Rapid exact inelastic biaxial bending analysis. *Journal of Structural Engineering*, 111(12):2659–2674.

[Rudnicki et al., 2003] Rudnicki, M., Bieniek, P., and Suski, P. (2003). Calculating indefinite integrals by means of genetic programming. In [Bugeda et al., 2003].

[Saka, 2003] Saka, M. (2003). Optimum design of skeletal structures: A review. In Topping, B., editor, *Progress in Civil and Structural Engineering Computing*, chapter 10, pages 237–284. Saxe-Coburg Publications.

[Saravanan et al., 1995] Saravanan, N., Fogel, D., and Nelson, K. (1995). A comparison of methods for self-adaptation in evolutionary algorithms. *BioSystems*, 36:157–166.

[Sarma and Adeli, 1998] Sarma, K. C. and Adeli, H. (1998). Cost optimization of concrete structures. *Journal of Structural Engineering*, 124(5):570–578.

[Schaffer, 1984] Schaffer, J. D. (1984). *Multiple Objective Optimization with Vector Evaluated Genetic Algorithms*. PhD thesis, Vanderbilt University.

[Schoenauer and Michalewicz, 1997] Schoenauer, M. and Michalewicz, Z. (July 19-23, 1997). Boundary operators for constrained parameter optimization problems. In *Proceedings of the $7^{th}$ International Conference on Genetic Algorithms*, pages 320–329, East Lansing, Michigan.

[Schwarz et al., 2001] Schwarz, S., Maute, K., and Ramm, E. (2001). Topology and shape optimization for elastoplastic structural response. *Computer Methods in Applied Mechanics and Engineering*, 190:2135–2155.

[Schwefel and Bäck, 1995] Schwefel, H.-P. and Bäck, T. (1995). Evolution strategies II: Theoretical aspects. In Périaux, J. and Winter, G., editors, *Genetic Algorithms in Engineering and Computer Science*. John Wiley & Sons Ltd, Chichester.

[Sfakianakis, 2002] Sfakianakis, M. G. (2002). Biaxial bending with axial force of reinforced, composite and repaired concrete sections of arbitrary shape by fiber model and computer graphics. *Advances in Engineering Software*, 33(4):181–243.

[Sokolowski and Zolesio, 1992] Sokolowski, J. and Zolesio, J. (1992). *Introduction to shape optimization, Shape sensitivity analysis*. Springer-Verlag, Berlin.

[Srinivas and Deb, 1994] Srinivas, N. and Deb, K. (1994). Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, 2(3):221–248.

[Steven, 2003] Steven, G. (2003). Product and system optimization in engineering simulation. FENet Newsletter, January 2003.

[Storn, 1996] Storn, R. (1996). On the usage of differential evolution for function optimization. In *NAPHIS 1996*, pages 519–523. Berkeley.

[Storn, WWW] Storn, R. (WWW). Homepage of Differential Evolution. `http://www.icsi.berkeley.edu/∼storn/code.html`.

[Storn and Price, 1995] Storn, R. and Price, K. (1995). Differential Evolution : A simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, University of Berkeley.

[Teplý and Novák, 1999] Teplý, B. and Novák, D. (1999). *Spolehlivost stavebních konstrukcí [Reliability of structures]*. CERM, s.r.o., Brno, Brno University of Technology. Lecture notes.

[Tong and Liu, 2004] Tong, F. and Liu, X. (2004). On training sample selection for artificial neural networks using number-theoretic methods. In Topping, B. and Soares, C. M., editors, *Proceedings of the Fourth International Conference on Engineering Computational Technology*. Civil-Comp Press, Stirling, Scotland.

[Tsoukalas and Uhrig, 1997] Tsoukalas, L. H. and Uhrig, R. E. (1997). *Fuzzy and neural aproaches in engineering*. John Wiley, New York.

[Van Iwaarden, 1996] Van Iwaarden, R. J. (1996). *An Improved unconstrained Global Optimization Algorithm*. PhD thesis, University of Colorado at Denver.

[Vidal, 1993] Vidal, R. V. V., editor (1993). *Applied Simulated Annealing*, volume 396 of *Lecture Notes in Economics and Mathematical Systems*. Springer-Verlag.

[von Neumann, 1966] von Neumann, J. (1966). *Theory of Self-Reproducing Automata*. ed. Burks A.W, Univ. Illinois Press, Urbana.

[Vondráček, 2001] Vondráček, R. (2001). Numerical methods in nonlinear concrete design. Master's thesis, Czech Technical University in Prague.

[Vondráček and Bittnar, 2002] Vondráček, R. and Bittnar, Z. (2002). Area integral of a stress function over a beam cross-section. In Topping, B. H. V. and Bittnar, Z., editors, *Proceedings of The Sixth International Conference on Computational Structures Technology*, pages 11–12, Prague, Czech Republic. Civil-Comp Press.

[Vořechovský, 2004] Vořechovský, M. (2004). *Stochastic Fracture Mechanics and Size Effect*. PhD thesis, Brno University of Technology.

[Vořechovský et al., 2002] Vořechovský, M., Novák, D., and Rusina, R. (2002). A new efficient technique for samples correlation in Latin Hypercube Sampling. In *Proceedings of the VII Scientific International Conference, Applied Mathematics*, Fakulta stavební, Košice, Slovensko.

[Wang et al., 2002] Wang, J., Periaux, J., and Sefrioui, M. (2002). Parallel evolutionary algorithms for optimization problems in aerospace engineering. *Journal of Computational and Applied Mathematics*, 149:155–169.

[Wiles and Tonkes, 2002] Wiles, J. and Tonkes, B. (2002). Visualisation of hierarchical cost surfaces for evolutionary computing. In *Visualisation of Hierarchical Cost Surfaces for Evolutionary Computing*, CEC '02. IEEE.

[Wolfram, 2002] Wolfram, S. (2002). *Cellular Automata and Complexity*. Perseus Publishing.

[Wolpert and Macready, 1997] Wolpert, D. H. and Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computations*, 1:67–82.

[Xie and Steven, 1997] Xie, Y.-M. and Steven, G. (1997). *Evolutionary Structural Optimization*. London: Springer-Verlag.

[Yagawa and Okuda, 1996] Yagawa, G. and Okuda, H. (1996). *Neural networks in computational mechanics*. CIMNE.

[Yoshimura and Inoue, 1995] Yoshimura, M. and Inoue, N. (1995). Optimization strategy for detailed designs of reinforced column structures using pareto optimum solutions of ideal models. In [Olhoff and Rozvany, 1995], pages 783–788.

[Yoshimura et al., 2002] Yoshimura, S., Dennis, B., and Kawai, H. (2002). Generalized approach to parallel shape optimization with millions DOF finite element model. In [Mang et al., 2002].

[Youssef et al., 2001] Youssef, H., Sait, S. M., and Adiche, H. (April 2001). Evolutionary algorithms, simulated annealing and tabu search: a comparative study. *Engineering Applications of Artificial Intelligence*, 14(2):167–181.

[Yuret, 1994] Yuret, D. (1994). From genetic algorithms to efficient optimization. Technical Report 1569, Massachusetts Institute of Technology, Artificial Intelligence Laboratory. Also available as `http://www.ai.mit.edu/people/deniz/publications/-aitr1569.ps.gz`.

[Zeman and Šejnoha, 2001] Zeman, J. and Šejnoha, M. (2001). Numerical evaluation of effective properties of graphite fiber tow impregnated by polymer matrix. *Journal of the Mechanics and Physics of Solids*, 49(1):69–90.

[Zhou and Rozvany, 2001] Zhou, M. and Rozvany, G. I. N. (2001). On the validity of ESO type methods in topology optimization. *Structural and Multidisciplinary Optimization*, 21:80–83.

[Zitzler, 1999] Zitzler, E. (1999). *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD thesis, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland.

[Zitzler et al., 2001] Zitzler, E., Laumanns, M., and Thiele, L. (2001). SPEA2: Improving the Strength Pareto Evolutionary Algorithm. In Giannakoglou, K., Tsahalis, D., Periaux, J., Papailou, P., and Fogarty, T., editors, *EUROGEN 2001. Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*, Athens, Greece.

[Zitzler and Thiele, 1999] Zitzler, E. and Thiele, L. (1999). Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271.

Appendix A

# INELASTIC BIAXIAL BENDING ANALYSIS

The aim of this chapter is to introduce the current state of the art within the analysis of biaxial bending of arbitrary composite sections and to utilize the gathered knowledge for the design of reinforced concrete cross-sections. Although the most important algorithm used hereafter - *the contour summation technique* - has been known for more than twenty years, the general knowledge of this methodology is probably not widely known in the engineering practice, because till nowadays many papers are being published about improving not-so-enhanced procedures.

## *A.1 Introduction*

The main goal here is the analysis of beam members loaded dominantly with the combination of axial force and bending moments with regard to the non-linear behavior of the material. There are usually two different tasks to be solved: the construction of **an interaction surface** and the assembly of **a tangent stiffness matrix**. The second task is not presented in this work, but the procedure presented later can be used as well and the resulting equations for the analytical solution can be found e.g. in [Rotter, 1985].

In the case of the interaction surface construction, we have three internal forces in the cross-section: one axial force $N_x$ and two bending moments $M_y$ and $M_z$. These forces are defined as the area integrals over the cross-sectional area $A$:

$$N_x = \iint\limits_A \sigma_x \,\mathrm{d}A \ , \tag{A.1}$$

$$M_y = \iint\limits_A \sigma_x z \,\mathrm{d}A \ , \tag{A.2}$$

$$M_z = -\iint\limits_A \sigma_x y \,\mathrm{d}A \ . \tag{A.3}$$

As long as we are working with an uniaxial strain $(\varepsilon_x \neq 0, \varepsilon_y = \varepsilon_z = \gamma_{xy} = \gamma_{xz} = \gamma_{yz} = 0)$, a material of the cross-section can be described using the following constitutive equation:

$$\sigma_x = \sigma(\varepsilon_x) \ . \tag{A.4}$$

In other words, a stress is a general function of a strain. This function is explicitly defined and known for each particular material and also represents the stress-strain diagram of the material. Moreover, the next assumption hereafter is the Bernoulli-Navier Hypothesis assuming that the plane section remains planar after deformation and perpendicular to the centerline. This also means that strain can be presented as a plane constructed over cross-section, where the track of the plane represents the neutral axis.

## *A.2 Historical overview*

In the history, the usual scenario of computing integrals (A.1) – (A.3) was to use well-known equations for the most common cross-sections and a given stress-strain relationship. However, in the time of fast computers and the finite element method, this procedure becomes too simplistic.

It would certainly be nice if there existed a closed-form solution for any cross-sectional shape as well as for an arbitrary stress-strain diagram. Unfortunately, there is no such to the knowledge of the author and probably does not exist. Therefore, inevitably some simplification or approximation must be used.

Firstly, we can distinguish the limitation on **a shape** or on **a stress function**, respectively. If the precise description of cross-section boundaries is not important, we can simplify them and hence we can have almost any stress-strain relationship. Another possibility is to restrict the description of material behavior to a linear, parabolic, mixed-linear and so on, which enables us to use precise delineation of a cross-section.

The next division can be done for the **closed-form** solutions and for the **numerical integration**. Based on the above-mentioned assumptions, there exist closed-form solutions for special cases of both simplification, the shape as well as material ones. If these premises do not hold, numerical integration is often used. And, of course, the main advantage of the former ones is the speed and easiness in comparison to the latter methods.

Finally, the algorithms can also be divided into two groups: **area-based** and **boundary based**. The former ones follow the second-order area integrals as such by dividing the area of a cross-section into smaller pieces - pixels [Sfakianakis, 2002], fibers [Romero et al., 2002], parallel layers [Bonet et al., 2002] or subsections [Chen et al., 2001], and then summing required quantity among these sub-parts. The boundary based ones are based on Green's Theorem by transforming integrals over the domain into boundary integrals, as will be shown later.

Till nowadays, there have been more than twenty papers dealing with these topics, several of them cited above and below. The most important and cited ones are to be shown here in more detail and their advantages and disadvantages will be discussed.

The first example is the work [Sfakianakis, 2002], which uses small graphical pixels to describe the cross-section and therefore it can be characterized as a shape-approximate, numerical and area-based procedure. Although opponents of this method argue that, except the proximity of these methods, it is slow especially due to the division to pixels, nowadays this argument must be rehabilitated. In this work this obstacle is solved by the usually unexplored computational power of current graphical devices to compose pixels of the composite concrete sections.

Next approach, e.g. [Chen et al., 2001], is nowadays an often used method (see also the work [Charalampakis and Koumousis, 2004] for the most recent references). It is based on the decomposition of the shape into trapezoids and circles. Because the stress-strain relationship is limited here only to linear and/or low-order polynomial curves and their combination, it enables us to find closed-form solutions for these shapes. The final values are then computed as the summation over these smaller pieces. Using our notation, it is limited in the stress-strain description as well as in the form of a shape (enables only lines and circles) and finally, this procedure is a closed-form and area-based method.

If the stress-strain diagram does not fulfill piecewise polynomial description, there still exist procedures to evaluate internal forces. Even though the procedure presented in [Fafitis, 2001] is not the oldest one and also is not the best, it can serve here as the introduction to boundary-based procedures. Green's Theorem is applied here to transform the area integration over an area $A$ into a line integration along the closed line $L$ that encloses the area:

$$\iint\limits_A \big(\partial Q/\partial y - \partial P/\partial z\big)\,\mathrm{d}y\,\mathrm{d}z = \oint\limits_L P\,\mathrm{d}y + \oint\limits_L Q\,\mathrm{d}z\ , \tag{A.5}$$

where $P$ and $Q$ are two arbitrary functions of $y$ and $z$. Now, let us set

$$P = 0 \quad \text{and} \tag{A.6}$$

$$Q = \frac{1}{r+1}\int y^{r+1} z^s g(z)\,\mathrm{d}z\ , \tag{A.7}$$

where $r$ and $s$ are nonnegative integers. Based on this choice the Eq. (A.5) becomes

$$\iint\limits_A y^r z^s g(z)\,\mathrm{d}y\,\mathrm{d}z = \frac{1}{r+1}\oint\limits_L y^{r+1} z^s g(z)\,\mathrm{d}z\ . \tag{A.8}$$

This re-formulation of the original problem is very suitable, because if $g(z) = 1$, the left-hand side represents

- the area $A$, for $r = 0$ and $s = 0$,
- the area moment $S_y$, for $r = 0$ and $s = 1$,
- the area moment $S_z$, for $r = 1$ and $s = 0$,
- the moment of inertia $I_y$, for $r = 0$ and $s = 2$,
- the moment of inertia $I_z$, for $r = 2$ and $s = 0$,
- the moment of inertia $I_{yz}$, for $r = 1$ and $s = 1$

and, moreover, if $g(z) = \sigma(\varepsilon_x(z))$, the left-hand side is

- the axial force $N_x$, for $r = 0$ and $s = 0$,
- the bending moment $M_y$, for $r = 0$ and $s = 1$ and, finally,
- the bending moment $M_z$, for $r = 1$ and $s = 0$.

Moreover, as noted in [Fafitis, 2001], the right-hand side of the Eq. (A.8) is a line integration along the sides of integration area $A$. And, in the case of polygonal boundaries, this integral diminishes into the summation over the individual lines creating the polygon. At this point A. Fafitis has applied numerical integration to evaluate line integrals, which, at least in this author's opinion, is not the right choice to follow. Even A. Fafitis admits that if the stress-strain function $\sigma(z)$ is integrable there exists closed-form formula for this line integral. This way was firstly discovered and developed by [Rotter, 1985]. We think, that closed-form solution is not only the computationally fastest enumeration but also a conceptually very simple method and, therefore, it seems that the work of J.M. Rotter is not widely known in the engineering practice. Otherwise the number of applications will be higher. This fact can be also illustrated by the work of R. Vondráček, see e.g. [Vondráček, 2001] and [Vondráček and Bittnar, 2002], who derived independently the same equations several years later. Next, this procedure is described in detail.

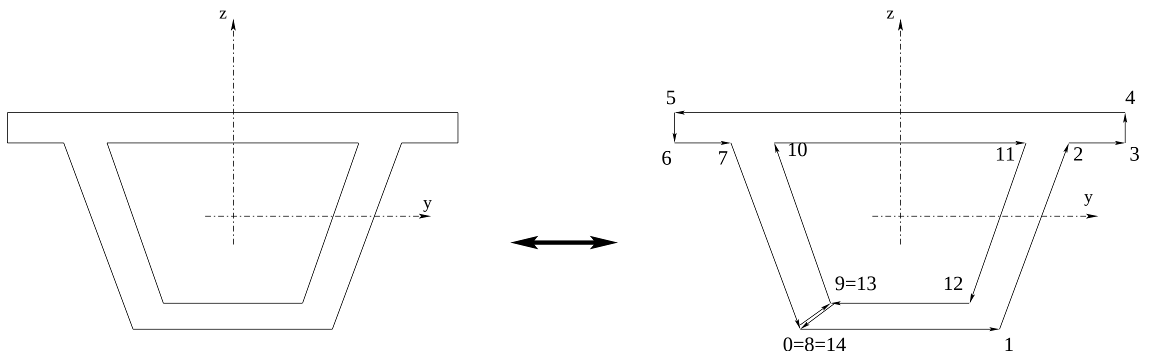


Figure A.1: Modification of a cross-section with openings.

## *A.3 Internal forces evaluation - the contour summation technique*

Let us suppose we have a cross-section of a general beam. This cross-section consists of several subsections, where each subsection is assumed to be of one material.

### *A.3.1 Subsection shape*

The subsection can be either a small simple shape, e.g. a steel bar, which is described by its 2D-coordinate and the area, or the subsection that can be described as an area bounded by a general polygon. This polygon must be closed and oriented counterclockwise. This description can be also used for openings (see A.1). When we run around the whole boundary of a subsection we can add one imaginary edge going from the last vertex of the boundary to the first vertex of the opening. We go along this edge and then we follow the boundary of the opening in clockwise direction. When we reach the last vertex of opening (which is simultaneously the first vertex), we go back along the imaginary edge in the opposite direction. We end up in the starting vertex again.

Finally, we can arrange coordinates of polygon vertexes into a vector $\mathbf{A} = \{A^{(0)}$, $A^{(1)}$,. . ., $A^{(n_p)}\}$, where $A^{(i)} = \{a_y^{(i)}, a_z^{(i)}\}^T$, $n_p$ is a number of the vertexes and $a_y^{(i)}, a_z^{(i)}$ are the Cartesian coordinates in the cross-section local coordinate system.

### *A.3.2 Coordinate system transformation*

According to the Bernoulli-Navier hypothesis mentioned previously, we can write the strain $\varepsilon_x$ as a function of the position in the plane, i.e.

$$\varepsilon_x(y, z) = \varepsilon_0 + z\kappa_z + y\kappa_y \ , \tag{A.9}$$

where $y$, $z$ are the local coordinates of the specific point on the cross-section, $\varepsilon_0$ is the strain in the cross-section origin (sometimes denoted as $u'$), $\kappa_z$, $\kappa_y$ are the curvatures in both directions (sometimes denoted as $v$”, $w$”).

To integrate stress over the cross-section, we also have to consider the fact that $\sigma_x$ is, in first row, function of $\varepsilon_x$, see Eq. (A.4), and that this strain is dependent both on $y$ as well on $z$ coordinate, see (A.9). To solve this obstacle, we define a new coordinate system on the cross-section. This Cartesian system is defined by two axes $u, v$, where $u$ is parallel to the neutral axis. This means that strain is only a function of $v$ coordinate of a specific cross-sectional point

$$\varepsilon_x(u,v) = \varepsilon_x(v) = \varepsilon_0 + v\kappa \ . \tag{A.10}$$

Consequently, stress is also a function only of $v$ coordinate of the specific cross-sectional point. The transformation from $y$, $z$ system to $u$, $v$ system is defined by means of transformation matrix **T** as

$$\left\{ \begin{array}{c} u \\ v \end{array} \right\} = \left[ \begin{array}{cc} \dfrac{\kappa_y}{\kappa} & \dfrac{\kappa_z}{\kappa} \\ -\dfrac{\kappa_z}{\kappa} & \dfrac{\kappa_y}{\kappa} \end{array} \right] \left\{ \begin{array}{c} y \\ z \end{array} \right\} = [\mathbf{T}] \left\{ \begin{array}{c} y \\ z \end{array} \right\} , \tag{A.11}$$

where

$$\kappa = \sqrt{\kappa_y^2 + \kappa_z^2} \ . \tag{A.12}$$

We can express $u$, $v$ coordinates of each vertex of the cross-section also in $u$,$v$ system as well as the whole polygon. If the boundary of the area $A$ consists of $n_p$ straight lines then each can be described parametrically by

$$X^{(i)}(t) = A^{(i)} + tB^{(i)} \ , \quad 0 \le t \le 1 \ , \quad i = 0, \ldots, n_p - 1 \ . \tag{A.13}$$

By substituting

$$B^{(i)} = \ A^{(i+1)} - A^{(i)} \ , \qquad \text{where } B^{(i)} = \{b_u^{(i)}, b_v^{(i)}\} \quad \text{and} \tag{A.14}$$

$$b_u^{(i)} = \ a_u^{(i+1)} - a_u^{(i)} \ , \qquad b_v^{(i)} = a_v^{(i+1)} - a_v^{(i)} \ , \tag{A.15}$$

we obtain

$$u(t) = a_u^{(i)} + tb_u^{(i)} \qquad \mathrm{d}u = b_u^{(i)}\,\mathrm{d}t \tag{A.16}$$

$$v(t) = a_v^{(i)} + tb_v^{(i)} \qquad \mathrm{d}v = b_v^{(i)}\,\mathrm{d}t \tag{A.17}$$

and for each line of the polygon the following relation between differentials is valid

$$\mathrm{d}u = \frac{b_u^{(i)}}{b_v^{(i)}}\,\mathrm{d}v = \frac{1}{k_i}\,\mathrm{d}v \ , \tag{A.18}$$

where $k_i$ is the tangent of $i$-th polygonal segment.

### *A.3.3 Stress integrals substitution*

As the stress $\sigma_x$ is a function of the strain $\varepsilon_x$ and the strain $\varepsilon_x$ is a function of the coordinate $v$, we have, for a given cross-section and a given plane of strain,

$$\sigma_x = \sigma_x(\varepsilon_x) = \sigma_x(\varepsilon_x(v)) \ . \tag{A.19}$$

Finally, we define stress integrals as

$$s(\epsilon) = \int_0^\epsilon \sigma(\xi)\,\mathrm{d}\xi \ , \tag{A.20}$$

$$ss(\epsilon) = \int_0^\epsilon s(\xi)\,\mathrm{d}\xi = \int_0^\epsilon \int_0^\xi \sigma(\eta)\,\mathrm{d}\eta\,\mathrm{d}\xi \ , \tag{A.21}$$

$$sss(\epsilon) = \int_0^\epsilon ss(\xi)\,\mathrm{d}\xi = \int_0^\epsilon \int_0^\xi s(\eta)\,\mathrm{d}\eta\,\mathrm{d}\xi = \int_0^\epsilon \int_0^\xi \int_0^\eta \sigma(\psi)\,\mathrm{d}\psi\,\mathrm{d}\eta\,\mathrm{d}\xi \ . \tag{A.22}$$

Based on this definition, now the right-hand side of the Eq. (A.8) can be expressed analytically and summed over individual lines. The stress resultants are then as follows, including also degenerate cases (i.e., $\kappa \to 0$ or $k_i \to 0$). For a detailed derivation, see again the works [Vondráček, 2001] or [Vondráček and Bittnar, 2002].

#### A.3.3.1 Axial force $N_x$

$$N_x = \sum_{i=0}^{n_p-1} N_x^{(i)} \, , \tag{A.23}$$

$$N_x^{(i)} = -\frac{1}{\kappa^2}\frac{1}{k_i}\left[ss(\epsilon)\right]_{\varepsilon^{(i)}}^{\varepsilon^{(i+1)}} \, , \tag{A.24}$$

$$\lim_{\kappa\to 0} N_x^{(i)} = -\sigma(\varepsilon_0) b_u^{(i)} \left( a_v^{(i)} + \frac{b_v^{(i)}}{2} \right) , \tag{A.25}$$

$$\lim_{b_v^{(i)}\to 0} N_x^{(i)} = -\frac{b_u^{(i)}}{\kappa}\left[s(\varepsilon^{(i)})\right] \, . \tag{A.26}$$

#### A.3.3.2 Bending moment $M_u$

$$M_u = \sum_{i=0}^{n_p-1} M_u^{(i)} \, , \tag{A.27}$$

$$M_u^{(i)} = -\frac{1}{\kappa^3}\frac{1}{k_i}\left[(\varepsilon-\varepsilon_0)ss(\epsilon) - 2sss(\epsilon)\right]_{\varepsilon^{(i)}}^{\varepsilon^{(i+1)}} \, , \tag{A.28}$$

$$\lim_{b_v^{(i)}\to 0} M_u^{(i)} = -\frac{b_v^{(i)}}{\kappa^2}\left[\kappa a_v^{(i)} s(\varepsilon(a_v^{(i)})) - ss(\varepsilon(a_v^{(i)}))\right] \, , \tag{A.29}$$

$$\lim_{\kappa\to 0} M_u = -\frac{1}{6}\sigma(\varepsilon_0)\sum_{i=0}^{n_p-1} b_u^{(i)}\left[3a_v^{(i)}b_v^{(i)} + 3(a_v^{(i)})^2 + (b_v^{(i)})^2\right] \, . \tag{A.30}$$

#### A.3.3.3 Bending moment $M_v$

$$M_v = \sum_{i=0}^{n_p-1} M_v^{(i)} \, , \tag{A.31}$$

$$M_v^{(i)} = -\frac{1}{\kappa}\left[\frac{1}{2}u^2(\varepsilon)s(\varepsilon) - \frac{1}{k_i\kappa}u(\varepsilon)ss(\varepsilon) - \frac{1}{k_i^2\kappa^2}sss(\varepsilon)\right]_{\varepsilon^{(i)}}^{\varepsilon^{(i+1)}} \, , \tag{A.32}$$

$$\lim_{b_v^{(i)}\to 0} M_v^{(i)} = 0 \, , \tag{A.33}$$

$$\lim_{\kappa\to 0} M_v = -\frac{1}{6}\sigma(\varepsilon_0)\sum_{i=0}^{n_p-1} b_v^{(i)}\left[3a_u^{(i)}b_u^{(i)} + 3(a_u^{(i)})^2 + (b_u^{(i)})^2\right] \, , \tag{A.34}$$

where $u(\varepsilon^{(i)})$ (respectively $u(\varepsilon^{(i+1)})$) means the $u$ coordinate of the beginning (respectively end) of the line segment of the boundary polygon.

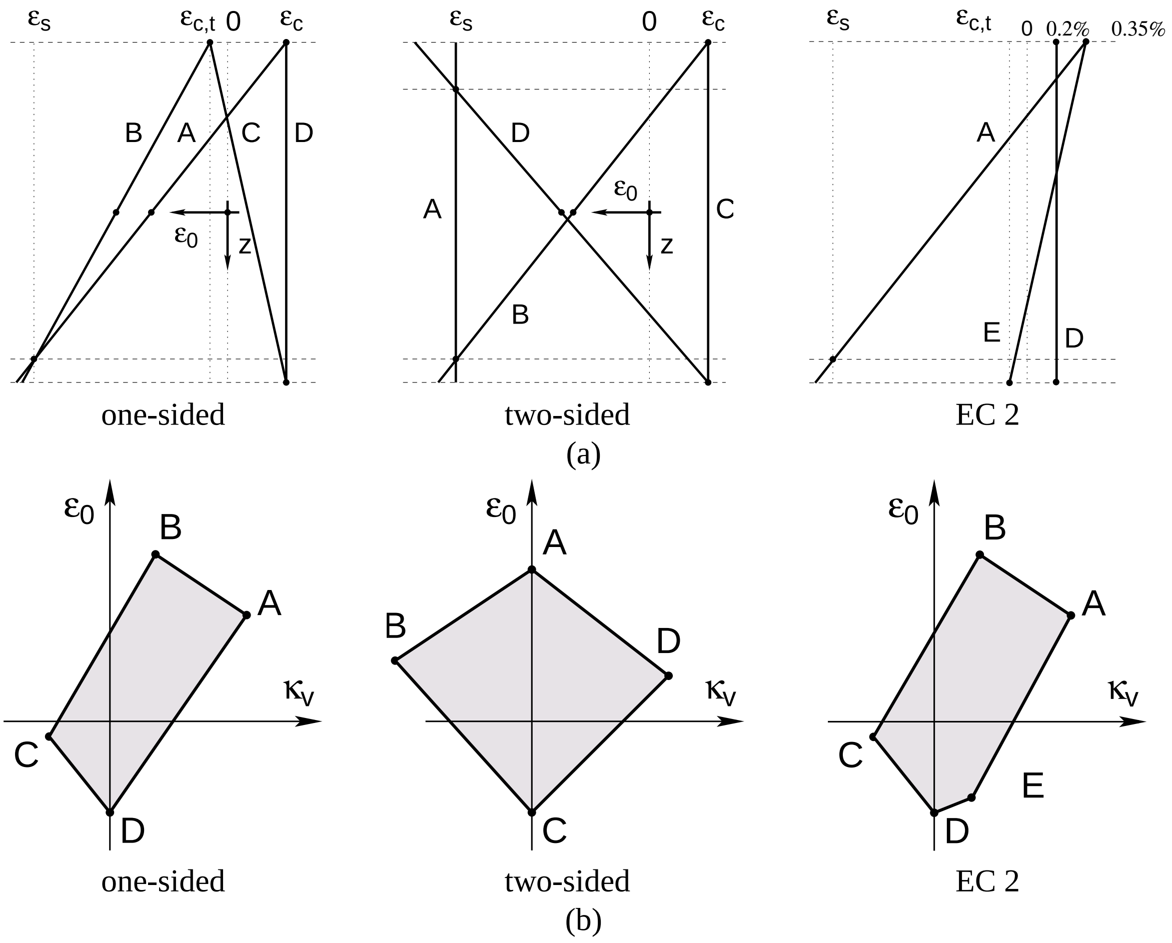


Figure A.2: Construction of the ultimate deformation border for one- and two-sided reinforcement and the EC 2 constraint. (a) Limit deformations, (b) Ultimate borders.

### *A.3.4 Transformation of internal forces*

After the integration we have three internal forces $N_x$, $M_u$ and $M_v$ in $u$,$v$ coordinate system. To get $M_y$ and $M_z$ we have to compute the following transformation using the transformation matrix **T** shown in Eq. (A.11).

$$\left\{ \begin{array}{c} M_y \\ M_z \end{array} \right\} = [\mathbf{T}]^T \left\{ \begin{array}{c} M_u \\ M_v \end{array} \right\} . \tag{A.35}$$

## *A.4 Construction of an interaction diagram*

From the practical point of view, the construction of the interaction surface of a cross-section subject to biaxial bending seems to be the most prominent problem to be solved. The aim of the current section is to employ the general procedure for evaluation of internal forces for a given plane of deformation, Eqs. (A.24) – (A.34), to provide a suitable framework for solving more advanced tasks of reinforced concrete cross-section analysis and design. Note, that for the simplicity we will limit our attention to the 2D problem only, but the procedure presented here is general and can be extended to 3D problems, too.

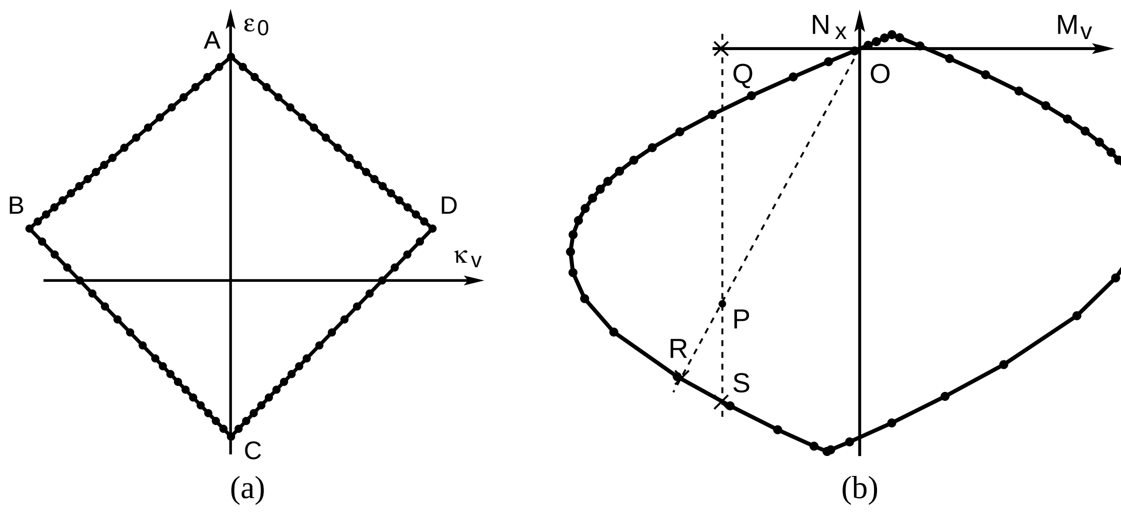


Figure A.3: Construction of interaction surface. (a) Discretization of deformation border, (b) Interaction surface

### *A.4.1 Ultimate deformation border*

Before proceeding with the interaction surface construction, it appears to be advantageous to introduce the notion of an *ultimate deformation border* first. Consider again a cross-section subject to a given normal strain $\varepsilon_0$ and principal curvature $\kappa_v$, see Fig. A.2a.

Then, the cross-section will be in the ultimate state if the normal strain in the concrete parts or the reinforcing members with the largest distance from the neutral axis reaches the limit value. In order to characterize the set of all possible deformation planes, consider the limit deformations A–D, schematically depicted in Fig. A.2a, corresponding to extremal values of normal strain $\varepsilon_0$ and principal curvature $\kappa_v$, respectively. As a consequence of the Bernoulli-Navier hypothesis, all deformations corresponding to the ultimate state can be expressed as a convex combination of the corresponding two limit deformations. Therefore, plotting the points A–D in the $\kappa_v \times \varepsilon_0$ coordinate system defines the polygon representing all possible ultimate deformations of a cross-section, see Fig. A.2b. Note that the introduced construction of the ultimate deformation border can be easily augmented to incorporate additional constraints imposed by design codes. For example, the EC2 standard [Eurocode 2, 1991] suggests a different value of maximal concrete compressive strain for a full compression (dominant axial force) than for a bending. This requirement can be easily incorporated into the ultimate border by inserting an additional point E between vertices A and D (see again Figs. A.2a and A.2b).

### *A.4.2 Construction of interaction surface*

Having introduced the ultimate deformation border, the construction of the interaction surface for a given cross section proceeds in a straightforward manner. To that end, each segment of the ultimate deformation border polygon is divided into a given number of intervals, see Fig. A.3a and for each point of the resulting discretization of ultimate border, the values of the normal force $N_x$ and bending moment $M_v$ are evaluated by Eqs. (A.24) – (A.26) and (A.31) – (A.34) which yields the polygonal approximation to the interaction surface, Fig. A.3b.

Two different indicators can be then defined to quantify the cross-section utilization for a given load corresponding to a point $P$ in the $M_v \times N_x$ coordinate system. In particular, the parameter $\eta_N$ serves to characterize the load-carrying capacity reserve for a load with an increasing normal force and a constant moment $M_v$ while the ratio $\eta_p$ measures the utilization of the cross section for the case of proportional loading. These quantities have a simple geometrical interpretation once the interaction surface has been constructed, see Fig. A.3b,

$$\eta_N = \frac{\|QP\|}{\|QS\|}, \qquad \eta_p = \frac{\|OP\|}{\|OR\|} \tag{A.36}$$

and allow us to determine the admissibility of the design point $P$ simply by checking one of the inequalities $\eta_N \leq 1$ or $\eta_p \leq 1$.

Appendix B

# GRAPHICAL ILLUSTRATION

## *B.1 Introduction*

Without any doubt, the graphical illustration is very important not only for decision making but can also play an important role during the optimization process. While the former is an essential part of both the research work as well as the practical one, the latter can help to understand the behavior of the optimization algorithm. The proper visualization can also bring a new insight into the studied problem, as will be shown in Section B.4.

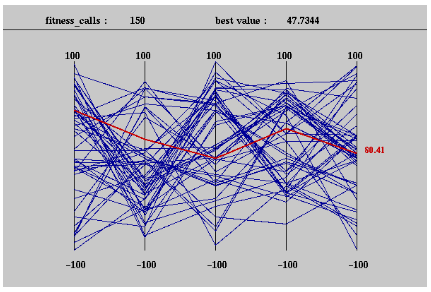
This chapter is organized as follows: Firstly, several types of illustration of data are presented. Secondly, their combination is proposed to build an effective tool for graphical representation of the solutions of multi-objective optimization. The advantage of the proposed methodology is shown in the last section.

## *B.2 Illustrating the pareto optimal set*

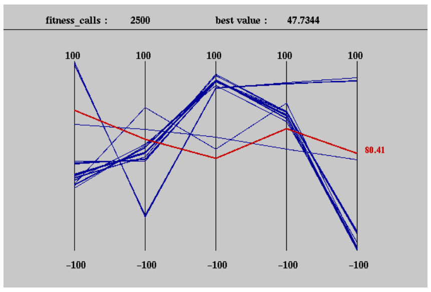
### *B.2.1 Illustrating real vectors*

A lot of researches have dealt with the problem how to display the sets of data to be clearly visible and understandable. During solving the multi-objective problems, the task of visualization of the Pareto-optimal set or Pareto-optimal front must be done. A vector $\mathbf{x} \in \mathbb{R}^2$ can be drawn in a plane, $\mathbf{x} \in \mathbb{R}^3$ in 3D and corresponding axonometries, but for more dimensions this approach can be enhanced only by time, color or sound without loosing general overview, e.g surfaces can be lost. If we limit our attention only to visualizations of vectors, i.e. the surfaces created by these vectors are not important, there exist several ways how to display them. For a comprehensive overview, see e.g. the work done by K. Miettinen [Miettinen, 1999, Chapter III.3], where several possibilities are described in detail. Namely, the *value path* (sometimes called *parallel axis* or *vertical scales*) in relative or absolute values can be used, see e.g. Fig. B.6 (left) and Fig. B.7. Another favorite ones are the *petal diagrams* or *bar charts*, see e.g. Fig. B.5 and Fig. B.6 (right), respectively. It is also worthwhile to mention other possibilities such as *star coordinate systems*, *spider-web charts* or *scatterplot matrixes*, see again the work [Miettinen, 1999] for more details.

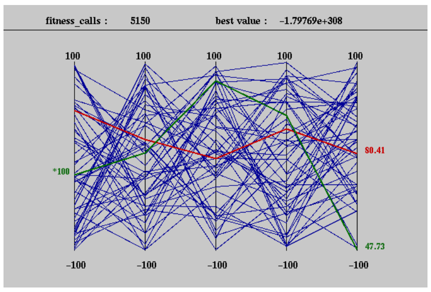
The advantage of visualization in the decision making process needs no comments because its profits are clear. The use of the presented illustration methods during an optimization run will be shown on the example of the *Type 0* function introduced previously in Section 3.2.2. The method used is the SADE algorithm (see Section 3.3.2) together with the CERAF technology for multi-modal optimization [Hrstka and Kučerová, 2004]. The results for five dimensions, i.e. $\mathbf{x} \in \mathbb{R}^5$, nine local and one global optimum are depicted in the Fig. B.1 using the *value path* methodology. Each variable has its own vertical axis with individual scale. The vectors of solutions are characterized by piece-wise linear lines crossing the variables' axes in the corresponding points.



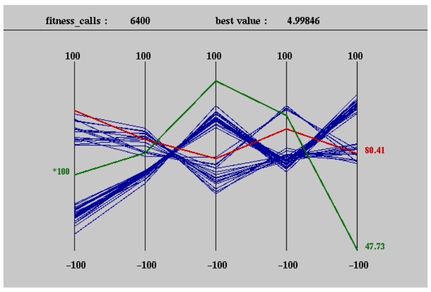
(a) Initial population.



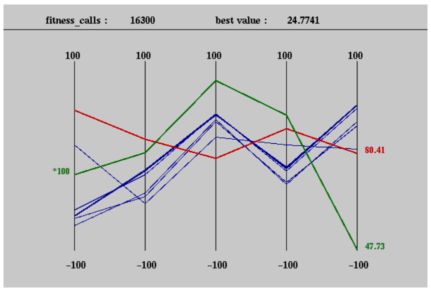
(b) Intermediate population.



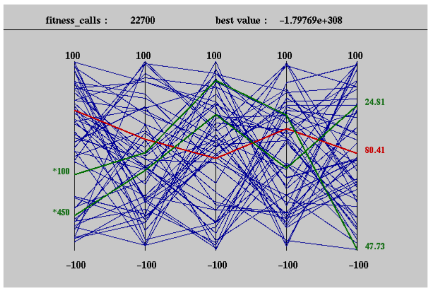
(c) First local minimum found.



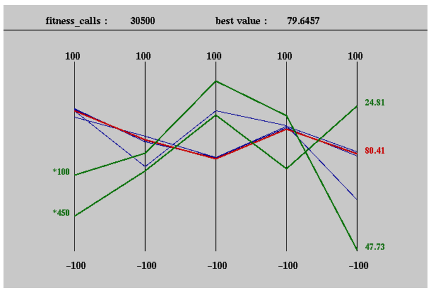
(d) Population at the 125$^{th}$ generation.



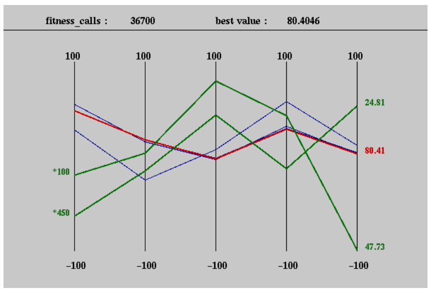
(e) Population at the 323$^{rd}$ generation.



(f) Second local minimum found.



(g) Population at the 607$^{th}$. generation.



(h) Global optimum found.

Figure B.1: A typical convergence of the multi-modal optimization algorithm. Pictures are provided by courtesy of Anna Kučerová.

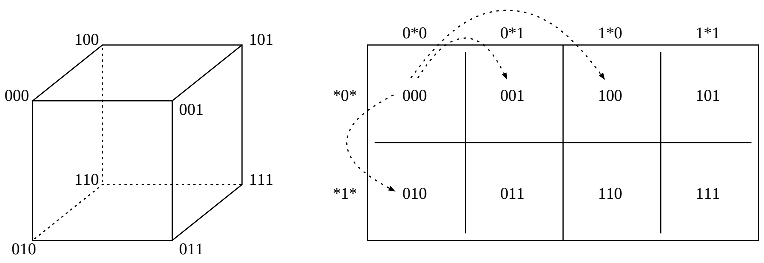


Figure B.2: Binary hypercube (left) and mapping to its corresponding hyperplane (right) proposed in [Wiles and Tonkes, 2002]. Rows and columns of the graph are labelled by their hyperplane templates and dotted arrows show the neighbors of $000$.

Following consecutive generations, the algorithm quickly proceeds from an initial population (a) to the valley of a local minimum (b). In the $100^{th}$ generation, the local minimum is selected (c) and new solutions are created. The population in Fig. B.1 (d) is characterized by the simultaneous search at two valleys. One of them (e) leads to the second minimum (f). And after some iterations (g) we have finally obtained the global minimum (h). This tracking of an optimization algorithm can say a lot of the behavior and is very useful in the developing of a new algorithm.

### *B.2.2 Illustrating discrete vectors*

When dealing with discrete values like integer or binary values, the conditions stated above for displaying vectors are valid too. Moreover, if we limit attention to the finite set of possible values, more options come to view. For instance, in the work [Wiles and Tonkes, 2002] a procedure for displaying moderate-dimensional binary spaces ($\{0, 1\}^n$ for $n \leq 16$) was proposed. It is based on the idea of unfolding a binary hypercube into a plane hypergraph, see Fig. B.2. Authors suggest, that this visualization can show not only values of an objective function in the $n$-dimensional binary space (usually by different colors or the extension into the 3D space), but also fitness surfaces, sizes and shapes of basins of attraction of local minima and other properties can be seen from such graph.

Moreover, this procedure can be easily extended to the integer or discrete sets. This is also more appropriate for engineering problems, where usually bounded values and also discrete ones come under investigation. One particular implementation can be defined on the basis of odd and even positions of variables in the vector: Let $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$, $n \in \mathbb{N}$, $x_i \in D_i$, is the $n$-dimensional vector of discrete values. Then, this spatial vector can be unfolded into the plane hypergraph having $\Pi_{i=1}^{2k-1}|D_i| \times \Pi_{i=2}^{2k}|D_i|$, $k = \lceil n/2 \rceil$, cells. The term $|D_i|$ states for the size of the discrete domain $D_i$. More particularly, odd positions are depicted in columns and their perturbation with even positions is made by the means of raws. For particular implementation, see section B.4 and especially Fig. B.9.

## *B.3 Illustrating a set of alternatives*

When studying multi-objective literature, one may have the feeling that the visualization of Pareto-optimal fronts is enough for an optimization process. Even more, usually only plane plots are shown and the 3D visualizations are very rare, see e.g. the works [Zitzler, 1999]

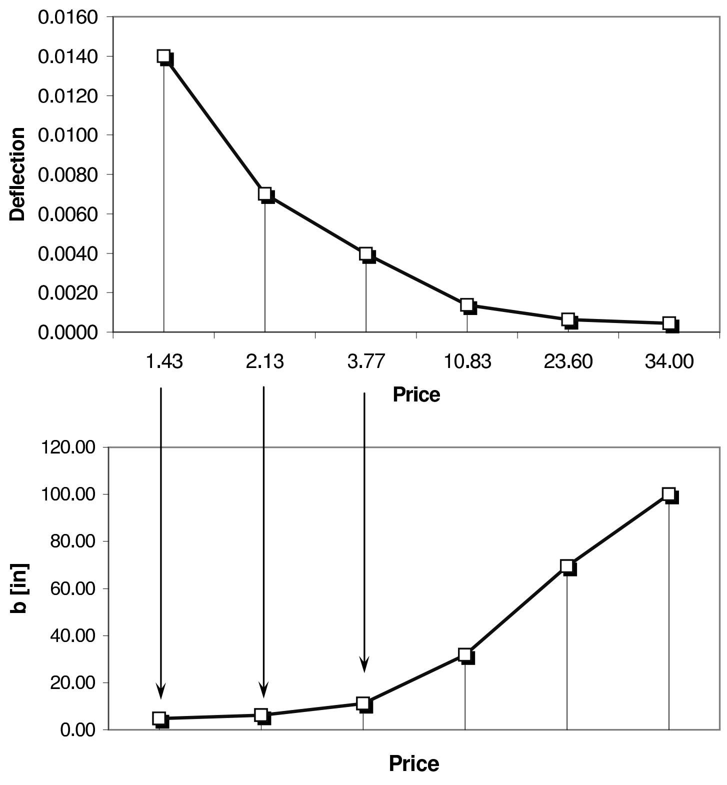


Figure B.3: The Pareto-optimum front (top) and the variable $b$ of the corresponding Pareto-optimal set (bottom).

or [Chiba et al., 2003]. The design/decision space is usually fully omitted. But this does not correspond to the usual practice needs, where the final decision is done in both - in the objective as well as the decision space. As is presented above, both spaces can be suitably drawn, but the connection between them is missing. Therefore the combination of these two spaces is proposed here: When inspecting the Pareto-optimality and all Pareto-optimal fronts, it is obvious, that two solutions can never be in one vertical line, because one would be dominant over the second one. Hence all Pareto-optimal solutions can be sorted in the terms of individual functions and also their $\mathrm{x}$ values can be sorted/drawn in this order. See Fig. B.3, where the Pareto-optimal front of the example studied later in this chapter is depicted above with correspondingly sorted values of the variable $b$ of individual solutions. This picture can give us “sensitivity” information on the variables’ influence on the objective function and, therefore, significantly help the designer to choose the proper solution.

The proposed methodology can be further enhanced be putting particular variables together. This is possible, if all variables have similar scales. In the opposite case of different scales, the relative measure for the values can be used. See e.g. Fig. B.8, where the $0$ % means the minimum and the $100$ % maximum of the variable, respectively. To show individual solutions and their variables together, the whole Pareto-optimal set is plotted in 3D.

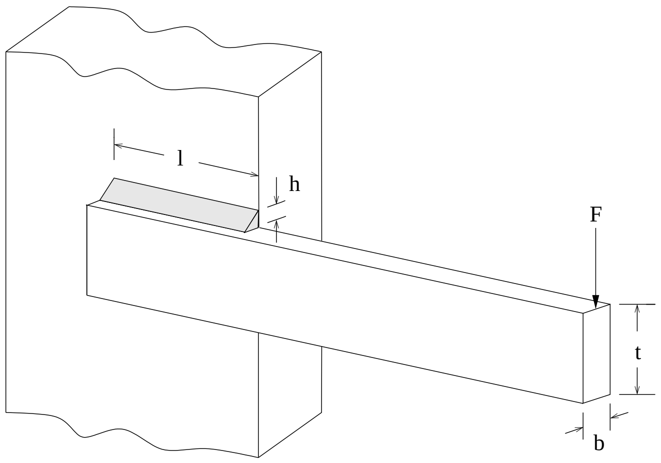


Figure B.4: The welded beam design multi-objective problem.

### *B.4 An engineering example*

To illustrate the advantages of the proposed methodologies, let us assume an often cited engineering example: a weld of two beams, see Fig. B.4. The goal is to minimize the cost of the welded beam and concurrently to minimize the end-deflection of this beam by changing four parameters (thickness of the beam $b$, width of the beam $t$, length of the weld $l$ and weld thickness $h$) formed in the design vector $\vec{x} = \{b, t, l, h\}$. The cantilevered part has a length of $14$ inches and a $6,000$ lb force $F$ is applied at the end of the beam. The variables are bounded by the following constraints: $0.125 \leq h$, $b \leq 5.0$ and $0.1 \leq l$, $t \leq 10.0$, all of them in inches. Following the description presented in [Deb, 1999], the whole problem can be formalized as follows:

$$\begin{aligned}
\text{minimize} \quad & f_1(\vec{x}) && = 1.10471h^2 l + 0.04811tb(14.0 + l)\ , && \text{(B.1)}\\
\text{minimize} \quad & f_2(\vec{x}) && = \delta(\vec{x})\ , && \text{(B.2)}\\
\text{subjected to} \quad & g_1(\vec{x}) && = 13,600 - \tau(\vec{x}) \geq 0\ , && \text{(B.3)}\\
& g_2(\vec{x}) && = 30,000 - \sigma(\vec{x}) \geq 0\ , && \text{(B.4)}\\
& g_3(\vec{x}) && = b - h \geq 0\ , && \text{(B.5)}\\
& g_4(\vec{x}) && = P_c(\vec{x}) - 6,000 \geq 0\ , && \text{(B.6)}
\end{aligned}$$

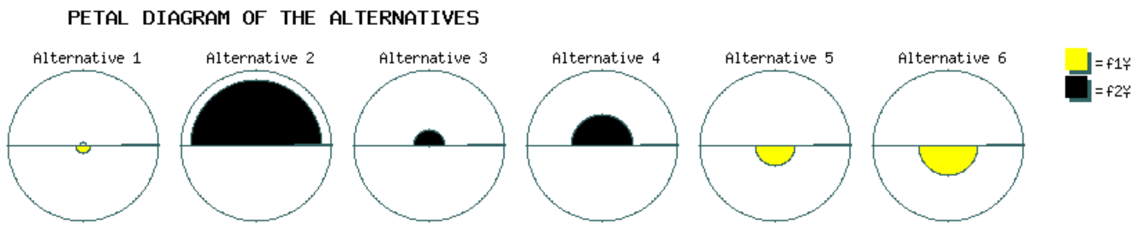
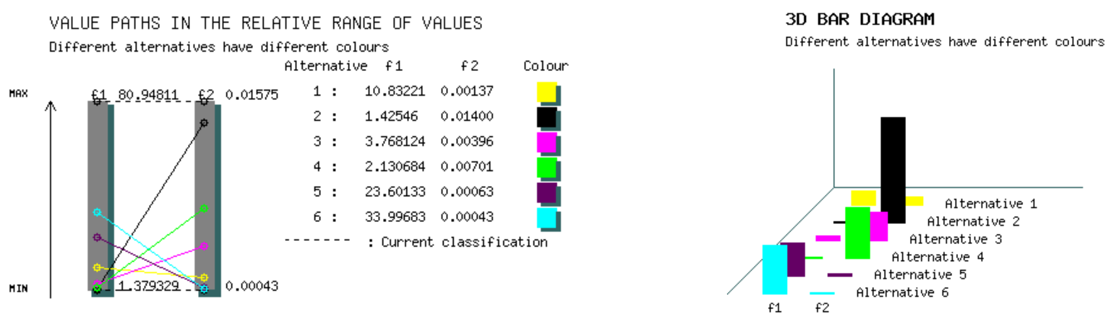


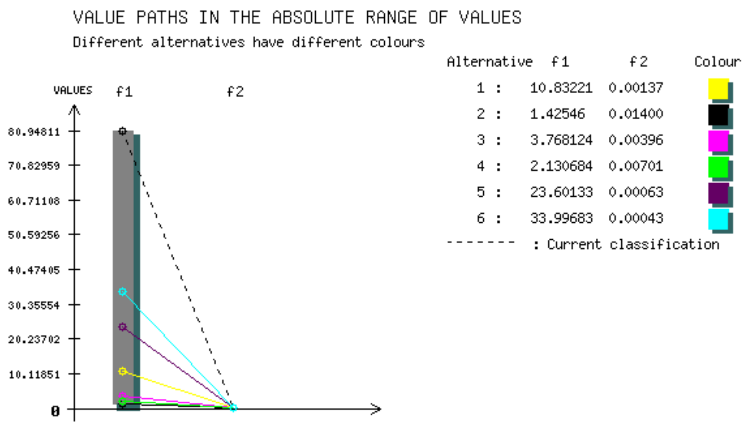
Figure B.5: Petal diagrams of six solutions from Pareto-optimal front.



| Alternative | f1 | f2 | Colour |
|---|---|---|---|
| 1 : | 10.83221 | 0.00137 | 🟨 |
| 2 : | 1.42546 | 0.01400 | ⬛ |
| 3 : | 3.768124 | 0.00396 | 🟪 |
| 4 : | 2.130684 | 0.00701 | 🟩 |
| 5 : | 23.60133 | 0.00063 | 🟪 |
| 6 : | 33.99683 | 0.00043 | 🟦 |

------- : Current classification

Figure B.6: Relative value paths (left) and 3D bar charts (right) of six solutions from Pareto-optimal front.



| Alternative | f1 | f2 | Colour |
|---|---|---|---|
| 1 : | 10.83221 | 0.00137 | 🟨 |
| 2 : | 1.42546 | 0.01400 | ⬛ |
| 3 : | 3.768124 | 0.00396 | 🟪 |
| 4 : | 2.130684 | 0.00701 | 🟩 |
| 5 : | 23.60133 | 0.00063 | 🟪 |
| 6 : | 33.99683 | 0.00043 | 🟦 |

------- : Current classification

Figure B.7: Value paths of six solutions from Pareto-optimal front.

where the deflection term $\delta(\vec{x})$ is given by

$$\delta(\vec{x}) = \frac{2.1952}{t^3 b} \ . \tag{B.7}$$

The first constraint $g_1$ deals with shear stress at the support location and the second constrain $g_2$ with normal stress at the same place. The third constraint $g_3$ ensures that the thickness of the beam is not smaller than the weld and the last constraint $g_4$ controls the allowable buckling load along the $t$ direction. The corresponding stress and buckling terms are defined as

$$\begin{aligned}
\tau(\vec{x}) &= \sqrt{(\tau')^2 + (\tau'')^2 + (l\tau'\tau'')/\sqrt{0.25(l^2+(h+t)^2)}} \ , && \text{(B.8)}\\
\tau' &= \frac{6,000}{\sqrt{2}hl} \ , && \text{(B.9)}\\
\tau'' &= \frac{6,000(14+0.5l)\sqrt{0.25(l^2+(h+t)^2)}}{2\{0.707hl(l^2/12+0.25(h+t)^2)\}} \ , && \text{(B.10)}\\
\sigma(\vec{x}) &= \frac{504,000}{t^2 b} \ , && \text{(B.11)}\\
P_c(\vec{x}) &= 64,746.022(1-0.0282346t)tb^3 \ . && \text{(B.12)}
\end{aligned}$$

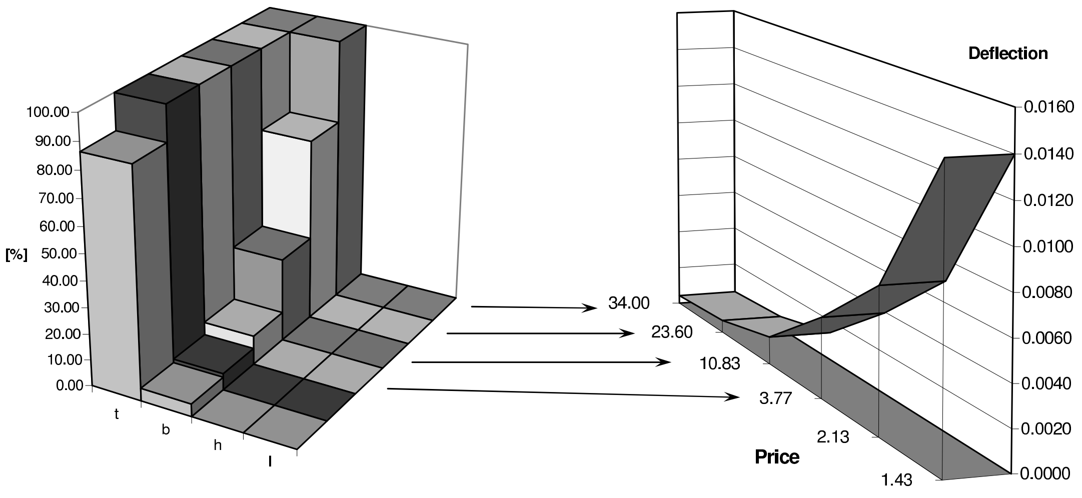


Figure B.8: Pareto-optimal set (left) and Pareto-optimal front (right) of the example studied.

For the solution, the Internet-based interactive multi-objective optimization system NIMBUS [Miettinen and Mäkelä, 2004] was used. Because the interactive mode does not enable the effective search for the whole Pareto-optimal set, only six solutions found are depicted. The comparisons of these solutions in the objective space are shown in different graphs in Figures B.5–B.7. Unfortunately, the NIMBUS system has no visualization tool for the decision space and offers only text view of the found Pareto-optimal set. Therefore, it is very difficult to discover the most important feature of the solutions for this engineering problem such that the solutions in the Pareto-optimal front are almost created only by changes in the variable $b$ and other variables are fixed in constant values [Deb, 1999]. If the proposed visualization is used, this fact is clearly visible, see Fig. B.8. The Pareto-optimal set is depicted on the left in relative measures. The six solutions are characterized by the minimal values of variables $l$ and $h$, almost maximal allowable values of the variable $t$ and, which is the most important information, by the increasing values of the variable $b$ in the connection with the decreasing deflection in the objective space, what is drawn on the right in Fig. B.8.

To show a possibility of proposed discrete visualization, imagine the discrete definition of the weld problem example. Let us assume the discreteness of the variables as discrete sets, particularly $l = \{1.0, 2.0, \ldots, 10.0\}$, $b = \{0.5, 1.0, \ldots, 5.0\}$, $t = \{1.0, 2.0, \ldots, 10.0\}$, and $h = \{0.5, 1.0, \ldots, 5.0\}$. Therefore, the whole domain is a $10 \times 10 \times 10 \times 10$ hypercube. Following description given in section B.2.2, this hypercube can be unfolded e.g. into the $|l||t| \times |b||h|$ hypergraph containing $100 \times 100$ cells. To illustrate the possibilities of this visualization, the feasible set is depicted by the gray color, see Fig. B.9. Note, that in this case the $\{i, j\}$ position of the vector $\vec{x} = \{l, b, t, h\}$ is given by the $i$-th row, $i = b|D_h| + h$, and the $j$-th column, $j = l|D_t| + t$, respectively. It can be easily gathered from this graph, that the feasible set is bounded by the $(1.0, 10.0) \times (2.0, 5.0) \times (2.0, 3.0) \times (0.5, 5.0)$ prism. This can be very useful in the design process to designer to track new feasible solutions.

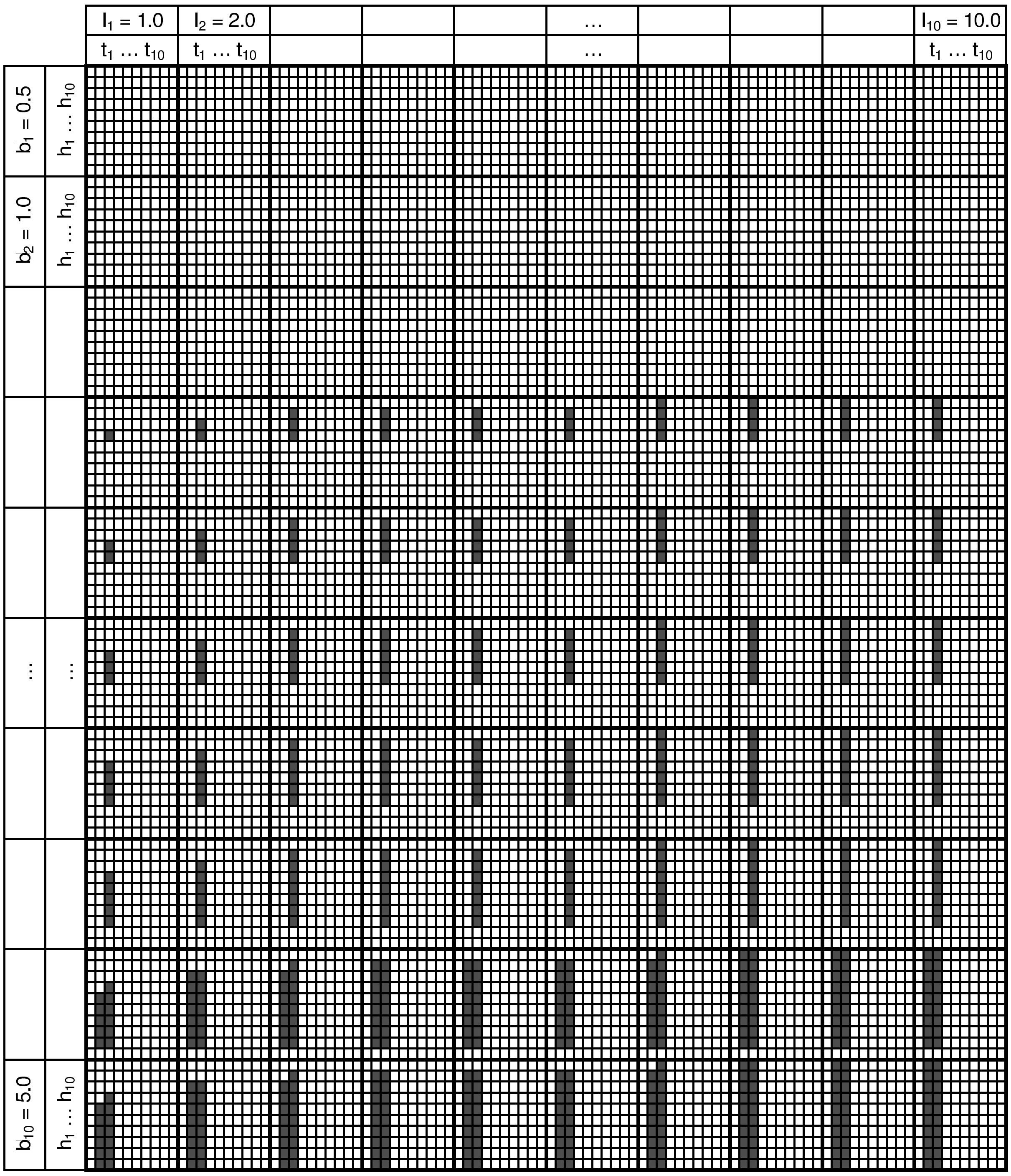


Figure B.9: The hypergraph of the feasible space in the discrete formulation of the studied example.

Appendix C

# PARAMETERS SETTINGS FOR FOUR EVOLUTIONARY ALGORITHMS

## *C.1 Parameter settings for RASA*

| **Parameter** | **Beam** | **Others** |
|---|---|---|
| `pop_size` | $64$ | $32$ |
| `q` | $0.04$ | $0.04$ |
| `p_uni_mut` | $0.525$ | $0.05$ |
| `p_bnd_mut` | $0.125$ | $0.05$ |
| `p_nun_mut` | $0.125$ | $0.05$ |
| `p_mnu_mut` | $0.125$ | $0.05$ |
| `p_smp_crs` | $0.025$ | $0.15$ |
| `p_sar_crs` | $0.025$ | $0.15$ |
| `p_war_crs` | $0.025$ | $0.15$ |
| `p_heu_crs` | $0.025$ | $0.35$ |
| `b` | $0.25$ | $2.0$ |
| `T_frac` | $10^{-2}$ | $10^{-10}$ |
| `T_frac_min` | $10^{-4}$ | $10^{-14}$ |
| `T_mult` | $0.9$ | $0.9$ |
| `num_success_max` | $10\times$`pop_size` | $10\times$`pop_size` |
| `num_counter_max` | $50\times$`pop_size` | $50\times$`pop_size` |
| `num_heu_max` | $20$ | $20$ |
| `precision` (step 4a) | see Section 3.3.3 | $10^{-4}$ |

Table C.1: Parameter settings for RASA

## *C.2 Parameter settings for DE*

| **Parameter** | **Chebychev, Type 0** | **Beam** | **PUC** |
|---|---|---|---|
| `pop_size` | $10 \times$ `dim` | $11 \times$ `dim` | $10 \times$ `dim` |
| $F_1 = F_2$ | $0.85$ | $0.85$ | $0.75$ |
| $CR$ | $1$ | $0.1$ | $1$ |

Table C.2: Parameter settings for DE

## *C.3 Parameter settings for SADE*

| **Parameter** | **Chebychev** | **Type 0** | **Beam** | **PUC** |
|---|---|---|---|---|
| `pop_size` | $10\times$`dim` | $25\times$`dim` | $10\times$`dim` | $10\times$`dim` |
| $CR$ | 0.44 | 0.1 | 0.3 | 0.2 |
| *radioactivity* | 0 | 0.05 | 0.05 | 0.3 |
| $MR$ | 0.5 | 0.5 | 0.5 | 0.5 |

Table C.3: Parameter settings for SADE

## *C.4 Parameter settings for IASA*

| **Parameter** | **Chebychev** | **Type 0** | **Beam** | **PUC** |
|---|---|---|---|---|
| `OldSize` | 80 | 900 | 180 | 200 |
| `NewSize` | 5 | 600 | 250 | 100 |
| `T_max` | $10^{-5}$ | $10^{-5}$ | $10^{-4}$ | $10^{-1}$ |
| `T_min` | $10^{-7}$ | $10^{-10}$ | $10^{-5}$ | $10^{-5}$ |
| `SuccessMax` | 1000 | 1000 | 1000 | 1000 |
| `CounterMax` | 5000 | 5000 | 5000 | 5000 |
| `TminAtCallsRate` | 19% | 100% | 25% | 20% |
| `CrossoverProb` | 97% | 92% | 60% | 90% |
| $CR$ | 0.5 | 0.6 | 1.3 | 1.0 |

Table C.4: Parameter settings for IASA

Appendix D

# COMPARISON OF DIFFERENT CORRELATION COEFFICIENTS

When computing a correlation coefficient, one may be confused, which of the many formulas to select. In the literature, see e.g. [Holmes, 2001], there can be found the three most often used ones:

## D.1 Pearson product moment correlation coefficient

This coefficient is only based on $x$,$y$ positions of individual samples and defined as

$$cor = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y})^2}} \ . \tag{D.1}$$

## D.2 Spearman rank correlation coefficient

This coefficient uses the knowledge of individual ranks and, hence it is given by

$$cor = 1 - \frac{6 \sum d_i^2}{n(n-1)(n+1)} \ , \tag{D.2}$$

where $d_i = r(x_i) - r(y_i)$ and $r(x_i)$ stands for the rank of $x_i$ value and the same relation is valid for $y_i$.

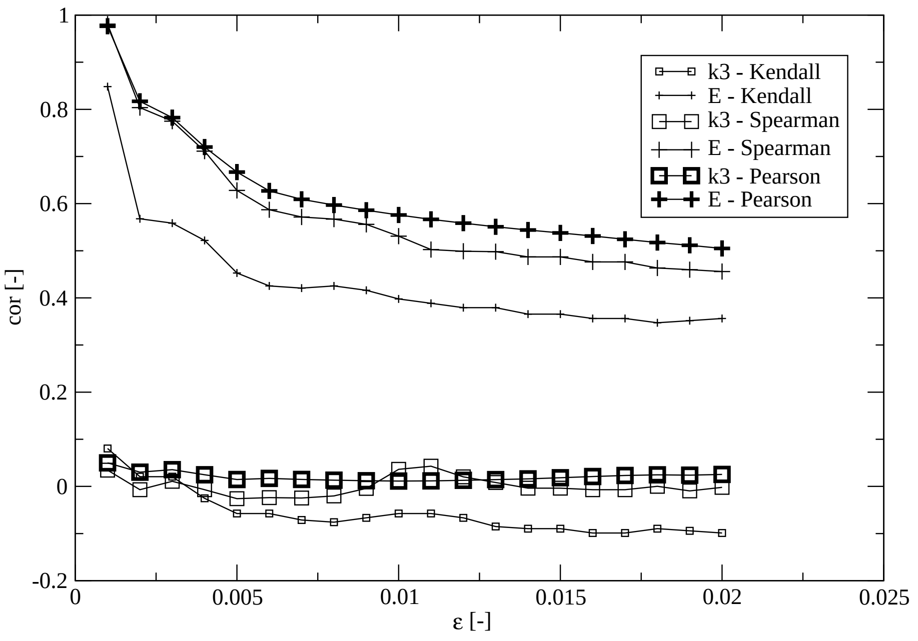
## D.3 Kendall rank correlation coefficient

This rank coefficient is based on the comparison of all possible pairs and can be written as

$$cor = \frac{2(C-D)}{n(n-1)} \ , \tag{D.3}$$

where $C$ and $D$ are numbers of growing and descending pairs, respectively.

How individual coefficients work and why is nicely presented in [Holmes, 2001]. Our aim is to select the most appropriate one from the point of view of stochastic sensitivity analysis of microplane material model parameters. For example, in the work [Teplý and Novák, 1999], the rank coefficient is claimed to be superior to the remaining ones due to its ability to better describe non-linear dependencies. Therefore, we have tried all these three coefficient on the stochastic sensitivity analysis described in Section 5.3.1. The results are shown in Figs. D.1–D.2. It can be clearly visible that, for this particular example, the differences are not significant and any of these coefficients can be used.



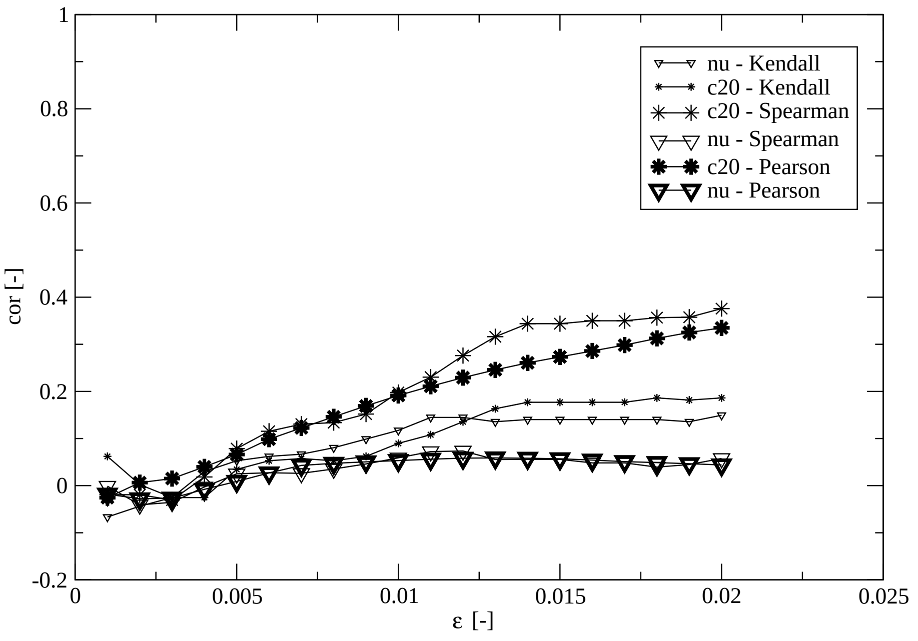
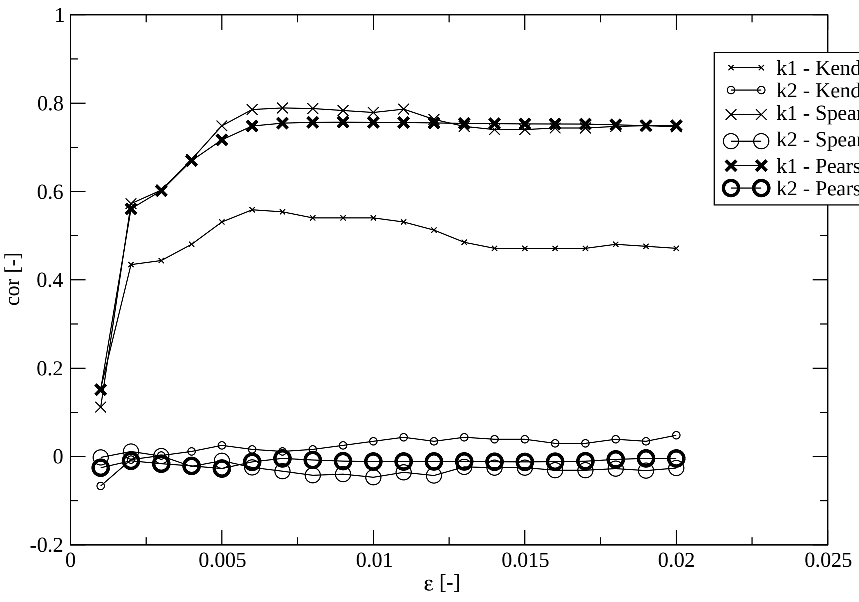


Figure D.1: Three coefficients as a sensitivity analysis of material model parameters on the shape of a stress-strain diagram (Part I).



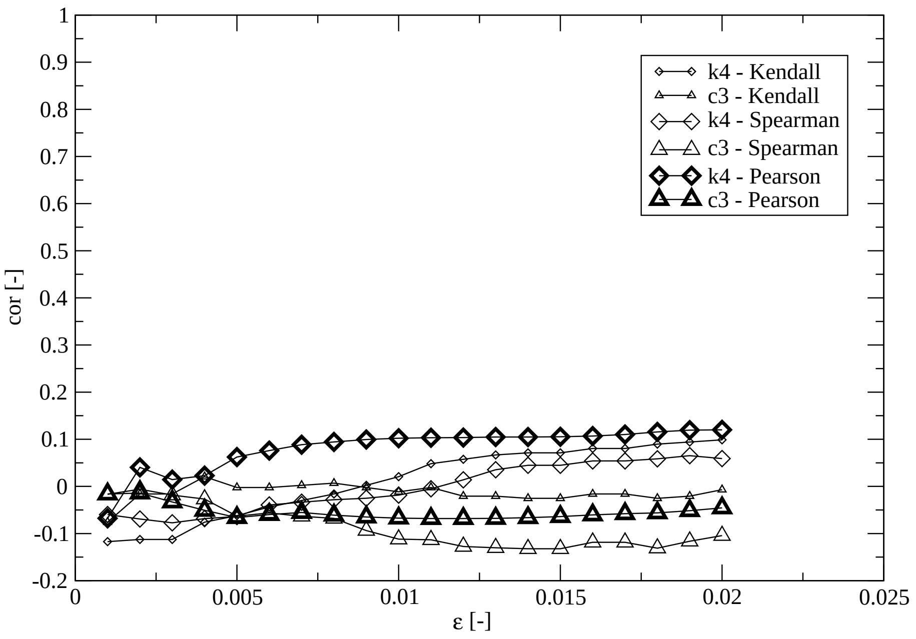


Figure D.2: Three coefficients as a sensitivity analysis of material model parameters on the shape of a stress-strain diagram (Part II).