FACULTY OF MECHANICAL ENGINEERING
DEPARTMENT OF COMPUTATIONAL
MECHANICS AND ENGINEERING

M. Sc. Eng. Tomasz Schlieter

OPTIMAL DESIGN OF MECHANICAL SYSTEMS FOR MULTIPLE CRITERIA BY MEANS OF SOFT COMPUTING METHODS

Supervisor:

Adam Długosz, BEng, PhD, DSc

Gliwice, 2021

WYDZIAŁ MECHANICZNY TECHNOLOGICZNY

KATEDRA MECHANIKI

I INŻYNIERII OBLICZENIOWEJ

Mgr inż. Tomasz Schlieter

OPTYMALNE PROJEKTOWANIE UKŁADÓW MECHANICZNYCH DLA WIELU KRYTERIÓW Z WYKORZYSTANIEM METOD OBLICZEŃ MIĘKKICH

Promotor:

Dr hab. inż. Adam Długosz, prof. PŚ

Gliwice, 2021

CONTENTS

BASIC NOTATION

Vectors and matrices are denoted by boldface letters, e.g., **K**

Latin symbols:

b_i – volume force,

c_{ij} – elastic constants,

$\boldsymbol{c_{ijkl}}$ – tensor of elastic constants,

CR – crossover rate,

DE_{iter} – number of generations in differential evolution,

DE_{pops} – size of population in differential evolution,

E – Young's modulus,

f_i – objective function,

f_{buck} – buckling factor,

F_c – contact force,

f – vector of objective functions,

F – scaling factor,

g_i – equality constraint,

h_i – inequality constraint,

hv – hypervolume,

I – electrical current,

I_z – moment of inertia about axis z,

I – global electric current vector,

k – thermal conductivity,

$\mathbf{K_E}$ – global electrical conductivity matrix,

$\mathbf{K_M}$ – global stiffness matrix,

$\mathbf{K_T}$ – global thermal conductivity matrix,

p – vector of parameters,

P – mechanical load vector,

p – mechanical load,

q – heat flux,

Q – internal heat source,

$\mathbf{Q}$ – global heat flux vector,

R – electrical resistivity,

t – time,

$\mathbf{t}$ – transition vector,

T – temperature,

$\mathbf{T}$ – global temperature vector,

u_i – displacement components,

$\mathbf{u}$ – displacement field,

$\mathbf{U}$ – global displacement vector,

$\mathbf{V}$ – global voltage vector,

v – velocity,

$\mathbf{x}$ – vector of design variables.

Greek symbols:

α – angle of attack,

α_c – heat convection coefficient,

α_t – thermal expansion coefficient,

Γ – part of the boundary,

ε_0 – vacuum permittivity,

ε_{ij} – strain tensor components,

θ – subspace of feasible solution space,

λ – Lamé's first parameter,

λ_i – Lagrange multipliers,

μ – Lamé's second parameter,

ν – Poisson's ratio,

ρ – charge flux density,

σ_{ij} – stress tensor components,

σ_{eq} – equivalent stress (Huber-Mises-Hencky hypothesis)

ϕ – electric potential,

φ – pre-bending angle,

Ω – feasible solution space.

1. INTRODUCTION

The design process of mechanical systems leads to creation of new technical objects which satisfy needs identified beforehand. One of the steps of this multiphase process is optimization during which the design is improved in the context of defined objectives. Objectives in the optimization process are related to previously defined needs. For optimization purposes, requirements must be expressed in mathematical form in a way that allows solutions to be compared with each other. It is possible to perform optimization considering either a single objective or a set of objectives at once. In many cases the requirements asked of mechanical systems are contradictory with each other and their simultaneous improvement is difficult or impossible. Results of multiobjective optimization are in the form of set of solutions rather than a single design and can provide additional information on the trade-offs between considered objectives. State and form of an optimized object is controlled by a set of design variables. The direction of search is driven by algorithms in a way to obtain improved solutions over time. There are numerous algorithms aiding the optimization process based on either deterministic or non-deterministic approach. Deterministic methods, known as hard computing, are well-studied and can be used to solve optimization problems of mechanical systems although they lack versatility provided by soft computing. Soft computing methods have been proven to be a successful tool used to enhance the process of optimal design of mechanical systems. Many multiobjective optimization problems have been solved using a wide range of soft computing methods, including very popular evolutionary algorithms which mimic mechanisms of biological evolution such as mutation, reproduction, and selection. There are however certain difficulties algorithms face when dealing with problems concerning many objectives, especially in case of three or more objectives. It is crucial for the algorithms to be able to effectively compare solutions in order to perform reasoning on the direction of the search. Due to the way solutions are compared with each other when multiple objectives are considered simultaneously it is sometimes challenging to draw conclusions on superiority of a certain solution among others. In a situation when too many solutions are incomparable, algorithms fail to efficiently drive the search towards improvement. Considering these difficulties, there is a need to address them and explore other ideas allowing to improve the optimization process. Real mechanical problems, except for particularly simple cases, cannot be solved in an analytical manner and so numerical methods are used for the purpose of simulation of their behaviour. These methods are often computationally expensive and thus time-consuming. Optimization process in general requires multiple numerical analyses to be performed during the search of a new, improved design. Due to the computational effort required to obtain optimization results, especially in case of more demanding multiobjective optimization problems, there is a clear need to establish efficient ways of optimization to find improved designs in limited time. Moreover, algorithms must be able to exchange information with software performing numerical analyses and at the end of the optimization process, results need to be presented in an informative way utilising suitable visualisation techniques to help decision making process based on established preferences.

1.1. Aims, assumptions and thesis

The dissertation aims to propose and develop a multiobjective optimization methodology capable of dealing with optimal design of mechanical systems concerning multiple criteria. The developed algorithm will belong to a group of soft computing methods and optimization will be based on a differential evolution and elements of game theory.

Performance of proposed algorithm will be evaluated by solving mathematical benchmark problems intended to cover a range of features that pose problems in the optimization of mechanical systems. Performance metrics will be compared with existing optimization algorithms. Furthermore, proposed method will be used to solve a set of both analytical and numerical mechanical problems. Efficiency of the algorithm will be understood as an ability to find satisfying results within limited time. Results of optimization tasks will be presented using multiple visualization techniques in order to improve the process of decision making.

In conclusion, the following thesis was formulated:

An algorithm based on differential evolution and elements of game theory can be used as an efficient tool in the optimal design of mechanical systems concerning multiple criteria.

1.2. **Review of content**

The dissertation consists of 6 chapters. The second chapter contains literature review related to problems of optimal design of mechanical systems. Brief information on the design of mechanical systems is introduced. Multiobjective optimization methods are discussed with particular attention given to soft computing methods. Chapter provides information on numerical methods in optimization. Decision making process aided by visualisation techniques is discussed. Moreover, in the chapter test problems and performance metrics related to assessing quality of optimization algorithms are reviewed.

In the third chapter developed algorithm based on differential evolution and elements of game theory is introduced. General idea of the algorithm is described and an example of game theoretic approach to optimization is shown. Chapter describes implementation of the algorithm and the way of communication with FEM systems.

In the fourth chapter methodology of comparative tests of the developed algorithm is proposed. Algorithm is examined based on metrics of performance using mathematical test functions. A set of six test functions is chosen to represent the real difficulties posed to optimization algorithms by mechanical systems. Proposed algorithm is compared with other well-known and broadly used multiobjective optimization algorithms: NSGA-II and NSGA-III. Conclusions on the results of comparison are drawn.

In the fourth chapter proposed algorithm is used to perform optimization tasks for a set of six mechanical problems. The optimized problems concern three cases in which the values of the objective function are obtained by analytical formulas. Moreover, three numerical problems are investigated, in which values of objective functions are simulated numerically by means of FEM. These problems include optimization of an airfoil, electrothermal microactuator and multiscale porous material. Results in this chapter are presented utilising selected techniques of visualisation of multivariate datasets. Chapter contains example of post-optimization decision making process. Each example is concluded with final remarks.

In the last chapter summary of the dissertation is presented along with conclusions and ideas on further research.

2. LITERATURE REVIEW

2.1. Design of mechanical systems

The purpose of designing is creating new technical objects motivated by specific needs and limited by means available to achieve them [1]. The word *design* stems from a Latin verb *designare* meaning *to designate* or *appoint.* The process of design starts with a need, a requirement or an idea and ends with information on how to manufacture and use a product, for example in the form of a set of drawings or a computer representation [2]. Design is an innovative, highly iterative process with multiple interactive phases [3]. Decision making process is an important phase of design. Steps associated with the design process are often illustrated as shown in the Fig. 1.

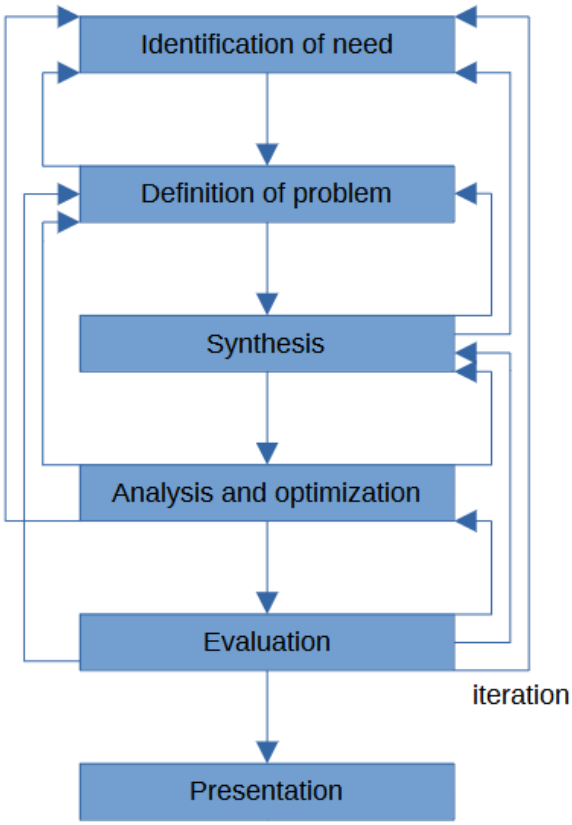


Fig. 1: Outline of the design process steps

Arrows in the Fig. 1 denote iteration and therefore it can be seen that the process involves many back-and-forth reasoning, and several phases can be repeated.

Identification of need begins the process, at this step, the need might be vague and must be recognised and phrased to start a creative act of design. Recognition of need is often triggered by adverse circumstances.

Following the identification comes the definition of problem, when specific goals are stated, including the input and output quantities, characteristics, and limitations of a desired product.

Synthesis, sometimes also referred to as *invention of the concept,* or *concept design* is the formative and creative stage of the design in which some solutions to a said problem are proposed, investigated, assessed, and improved.

In the analysis and optimization phase it is necessary to construct abstract models of a system to which mathematical tools could be applied. The mathematical model is supposed to simulate the mechanical system in a satisfying way, which is often a challenging task. The model is used to quantify whether the demands identified earlier are fulfilled and to further improve it in an optimization process. In general, mathematical model is a set of variables describing an object and its state and a set of mathematical relations between them. In the optimization model this set of variables is composed of a set of constant parameters and a set of design variables. Parameters are constant during the optimization process and can include for example physical properties (e.g., Young's modulus or density) of a system and design variables (e.g., width of a beam, hole diameter) are changed during the optimization process in order to improve design in the context of a specific need. In general, constraints can be imposed on the design variables to reflect real limits of a system. To assess the quality of solution, an objective function (also called fitness function) must be established as a function of design variables. Objective function is supposed to represent specific real needs asked of product, such as for example low cost, low mass, or high stiffness.

Evaluation focuses on providing final proofs of successful design and often incorporates creation and assessment of a prototype in a laboratory environment. The evaluation process should answer questions such as: Are the needs satisfied? Is the product reliable? Can it compete with similar products? Is it possible and economically viable to manufacture and use the product? Can the product generate profit by sale or use?

The final step of the process is presentation, where a new solution is described to others. The new design is an accomplishment which should be explained in an appropriate way, often from a marketing perspective, taking into consideration varying levels of knowledge on the topic among the audience.

Engineers solving design problems nowadays are assisted with a great variety of tools and resources. Computer software packages provide tools to enhance design, analysis, and simulation of mechanical systems. Computer-aided design (CAD) software allows to develop 3D models of mechanical components and to prepare documentation, often in the form of 2D orthographic views supplemented with dimensions. CAD models can also be used to determine geometrical properties of objects quickly and accurately, such as: volume, center of gravity or moments of inertia. Another application of CAD models is prevalent after supplementing them to software dedicated to analysis and simulation, including finite element method (FEM) software for analysis of stress and deflection, heat transfer, vibration (e.g. MSC Patran/Nastran, Ansys, Abaqus), computation fluid dynamics (CFD) software for analysis and simulation of fluid flows (e.g. Ansys Fluent), dynamic systems analysis (e.g. ADAMS) or interdisciplinary coupled fields analysis, often referred to as multiphysics which deals with coupled systems involving more than one physical fields simultaneously occurring and interacting with each other (e.g. COMSOL, Ansys Multiphysics). Multiphysics cover a range of scientific and engineering disciplines and involves problems such as thermo-mechanical, electro-mechanical, electro-thermo-mechanical. Besides electric and thermal, other fields often accompanying mechanical problems are acoustic, magnetic, and fluid-dynamic. Software dedicated to analysis

is often referred to as computer-aided engineering (CAE) software. Present CAE software often includes optimization modules, however in many cases it is limited to single-objective optimization and if multiobjective optimization is possible, it is often according to NSGA-II algorithm, which has significant limitations. There is other non-engineering specific software, which is often utilised by engineers during the design process, such as word processing and spreadsheet software (e.g., Microsoft Office, LibreOffice), graphic software (e.g., Corel, Photoshop) or mathematical solvers (e.g., MATLAB, Scilab, Octave, MathCAD).

2.2. **Optimization of mechanical systems**

Among many phases of design of mechanical systems, the greatest emphasis in this dissertation is devoted to optimization. Whenever a product is created or designed some needs are supposed to be satisfied in the best possible way and therefore optimization is performed. The optimization process is often manual and involves a search of a solution and associated parameters in a step-by-step approach. A manual optimization process often does not provide extensive exploration and exploitation of the solution space and thus produces sub-optimal solutions. Exploration mechanism is understood as ability of optimization algorithm to explore new regions in a feasible solution space, while exploitation is a local search focused on a promising region. With the advance of computational methods, algorithms, software engineering and computational capabilities of hardware and on the other hand global competition it became possible and essential to design products to satisfy consumers needs in the most effective way. Replacing manual approach to optimization with an automatic process and application of intelligent computing helps achieve this goal [4]. In many real problems it is not possible to find a strictly optimal solution due to the lack of knowledge or size of the problem, in such case an optimization process essentially aims to find a design improvement [5].

To perform an efficient optimization of a mechanical system, it is necessary to operate with a model. There are many ways of understanding the meaning of a model, for example [6] defines model of a real object as *an imaginable or materially realisable system which, by reflecting or reproducing an object, is capable of replacing it so that its study provides new, verifiable information about the object*.

Model can also be understood as a good enough simplification of an object. There are many simple yet popular and successful models in constant use in problems concerning mechanical systems, for example point mass, rigid body, or deformable body. In the problems of strength of materials models of rods, beams or trusses for example are widely used. In fluid dynamics Newtonian fluid is an important model. These models are abstract, expressed in form of certain concepts. There is also a group of material models, which should be understood in a different way – they imitate object of interest in a certain way for the purpose of satisfying a specific goal, analysis, or synthesis. An example of application of a material model is experimental stress distribution analysis utilising elasto-optic effect.

In case of optimization problems, mechanical systems need to be expressed in the form of mathematical tools and language, allowing reduction of a system to a formal, numerical expressions – a mathematical model. There are several methods of examining mathematical models, particularly: analytical methods, which are looking for solutions either qualitative or quantitative, exact, or approximate in the form of analytical expression. Results of these methods are convenient to analyse, although it is often very challenging or even impossible to

apply analytical methods to complex systems, especially in case of non-linear or multidimensional systems. Numerical methods utilise computers and iterative computations to produce usually approximate results, however the approximation can in many cases be reliably precise. These methods found broad application in optimization of mechanical systems thanks to FEM software, which can be used and automatically compute values of optimization objective functions. Other methods of dealing with mathematical models include graphic methods and experimental methods, neither of which found a significant application in optimization problems.

Performing design optimization can utilise or require knowledge on some aspects of a problem such as: current design, design variables and their limits, constraints, parameters, and objective functions. Variables can take either quantitative or qualitative form, however quantitative form is more desired as it can be easily expressed in the mathematical form, e.g., length or temperature of a system can take an explicit numerical value, while qualitative expressions such as aesthetics or ease of manufacturing require additional transformation to be expressed in the mathematical form. In most cases bounds are imposed on the variable, limiting the search space. Besides simple limits, constraints can be applied on the system, which can take form of equalities or inequalities. Equality constraints are particularly hard to satisfy for any optimization task and if it is possible, should be transformed to inequality constraints [7]. Optimization functions can, like variables, take qualitive or quantitative form, among which the latter is desired for similar reasons. Qualitative optimization functions are generally knowledge-based and quantitative optimization functions are derived from analytical or numerical models. Taking the aforementioned aspects into consideration, the optimization problem can be described mathematically as a search of a vector of design variables:

$$\mathbf{x} = [x_1, x_2, \dots, x_n]^T \tag{1}$$

under m equality constraints

$$g_i(\mathbf{x}) = 0, i = 1,2, \dots, m \tag{2}$$

and p inequality constraints

$$h_j(\mathbf{x}) \geq 0, j = 1,2, \dots, p \tag{3}$$

to optimize (maximise or minimise) k objective functions

$$\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_k(\mathbf{x})]^T \tag{4}$$

It is worth noting that maximization of a function $f(\mathbf{x})$ is equivalent to a problem of minimization of a function $-f(\mathbf{x})$ and therefore maximization and minimization functions can be easily transformed. It is a common practice to transform and present all the optimization functions in the form of minimization. Multiple objective functions are used to represent the quality of solution in terms of different measures of performance, e.g., weight, mass, cost, and deflection. These objective functions usually have contradictory nature.

Classification of design optimization problems is an essential task necessary to select an appropriate approach to examined problem. The classification schemes and categories were used to construct a Tab. 1, presenting classification of engineering design problems [5]. A choice of a method of optimization to solve a specific problem should follow the analysis of the model, particularly desired precision of results, analysis of objective functions and computational means available.

Tab. 1: Classification of engineering design optimization problems [5]

Classification schemes based on		Categories
Design variables	Number of variables	• Single-dimensional • Multi-dimensional
	Nature of design variables	• Static • Dynamic
	Permissible values of design variables	• Integer-values • Real-valued • Hybrid
	Dependence among design variables	• Independent-variable • Dependent-variable

Constraints	Existence of constraints	• Constrained
		  ○ Inequality
		  ○ Equality
		  ○ Linear
		  ○ Non-linear
		  ○ Separable
		  ○ Inseparable
		• Unconstrained

Objective functions	Number of objective functions	• Single-objective
		• Multiobjective
		  ○ <10 objectives
		  ○ Large scale multiobjective (>10 objectives)

	Nature of objective functions	• Quantitative
		  ○ Simulation based
		  ○ Analytical (linear and non-linear)
		  ○ Empirical
		• Qualitative
		• Hybrid

		• Inexpensive • Computationally expensive
		• Uni-modal • Multimodal
		• Linear • Non-linear
		• Continuous • Discontinuous • Not defined outside the feasible space
	Separability of objective functions	• Separable • Non-separable
Problem domain	Physics of problem	• Mechanics • Thermal • Electric • Multi-physics
Environment	Uncertain	• Without uncertainty • Uncertain • Robust • Reliability based
	Existing knowledge about the problem	• Known search-space • Unknown search-space
	Designer confidence required	• Interactive • Qualitative
	Nature of the environment	• Static • Dynamic

Among many methods of optimization to choose from, certain attention needs to be paid to classical deterministic methods. These methods utilise analytical properties of optimization problems to converge to optimal solution or find its approximation. Important drawbacks of deterministic methods of optimization include low flexibility, low efficiency dealing with certain problems [8] and limited scope of applicability of some of these methods. Selected deterministic methods of optimization will be discussed further in the following paragraphs.

Analytical method – using this method an optimum of function is sought as function extremum obtained by differentiation. This method is an application of extreme value theorem in mathematical optimization. Extreme value theorem also called Weierstrass extreme value theorem says that if a real-valued function f is continuous on a closed interval $[a, b]$, then f must have at least one maximum and minimum. Therefore, numbers c and d in $[a, b]$ exist such that:

$$f(c) \geq f(x) \geq f(d) \ \forall_x \in [a, b] \tag{5}$$

Analytical method aims to find a global extreme and all local extrema of function. If the global extremum of function does not belong to the feasible solution space, then extremum of function should be sought on the boundary area of the solution space. Similar approach should be used if analysed function does not include any extremes. Using this method does not provide any further information on the nature of optimization problem besides the location of extrema and can only be used to solve problems with continuous design variable space. Application of this method is limited to simple problems with low dimensionality of design variable space uncommon in engineering practice. Example of a problem which could be solved using analytical method is optimization of a cross section of a simple cantilever beam to minimize mass of a beam considering allowable stresses in the system.

Method of Lagrange multipliers – is used to solve constrained optimization problems with constraints under the assumption that objective function $f(\mathbf{x}, \mathbf{p})$ and all the constraints $g_j(\mathbf{x}, \mathbf{p}) = 0, j = 1, \dots, s$ are differentiable with respect to all arguments. In this method a function L is build that:

$$L(\mathbf{x}, \lambda, \mathbf{p}) = f(\mathbf{x}, \mathbf{p}) + \sum_{j=1}^{s} \lambda_j g_j(\mathbf{x}, \mathbf{p}) \tag{6}$$

where $\lambda_j = \lambda_1, \dots, \lambda_s$ is set of Lagrange multipliers.

It can be proven that $\mathbf{x}_{opt}$ representing global or local extremum of function $f(\mathbf{x}, \mathbf{p})$ satisfies the system of equations:

$$\begin{aligned} \frac{\partial L(\mathbf{x}, \lambda, \mathbf{p})}{\partial x_i} &= 0 \qquad i = 1, \dots, n \\ g_j(\mathbf{x}, \mathbf{p}) &= 0 \qquad j = 1, \dots, s \end{aligned} \tag{7}$$

provided that rank r of a matrix $\mathbf{D}$:

$$\mathbf{D} = \begin{bmatrix} \frac{\partial g_[(\mathbf{x}_{opt}, \mathbf{p})}{\partial x_i} & \cdots & \frac{\partial g_](\mathbf{x}_{opt}, \mathbf{p})}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial g_s(\mathbf{x}_{opt}, \mathbf{p})}{\partial x_i} & \cdots & \frac{\partial g_s(\mathbf{x}_{opt}, \mathbf{p})}{\partial x_n} \end{bmatrix} \tag{8}$$

equals $r = \mathrm{rank}(\mathbf{D}) = s$.

Systematic search method – involves choosing a set of points in the feasible solution space, where values of objective functions are determined. Based on this information, assuming adequate density of points an approximate extremum can be found with a given accuracy, which depends on the selected set of points. Selection of points is a vital part of the method so as to provide desired accuracy within a limited computational effort. If the points are chosen before the computation, then the approach is called simultaneous and if they are chosen during the computation considering previously obtained results, then the approach is called sequential and the latter one is typically more effective. This method requires discretization of the feasible solution space Ω or a subspace θ, assuming $\mathbf{x}_{opt} \in \theta \cap \Omega$ (Fig. 2). The density of the grid imposed on the search space reflects on the accuracy of results obtained by the method, however opting into too fine grid results in extensive computational effort required by the method.

Among methods of sequential search of multidimensional spaces of design variables, one of the most popular and influential methods is the gradient descent method attributed to a

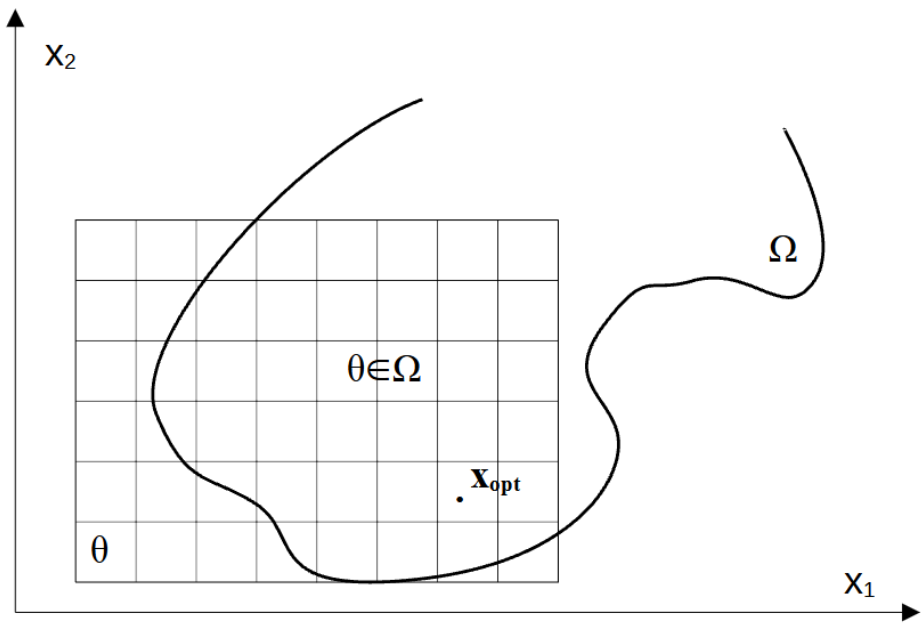


Fig. 2: Systematic search method

French mathematician Augustin Louis Cauchy who introduced it in his 1847 article *Methode generale pour la resolution des systemes d'equations simultanees* [9], [10]. Similar method was independently proposed in 1908 by a French mathematician Jacques Hadamard [11]. American mathematician Haskell Curry studied the convergence of the method for non-linear optimization problems and called it a method of steepest descent [12]. The method proceeds with the search following the direction of the steepest descent (assuming minimization problem)

of the objective function. This direction is obtained by calculation of objective function and its first derivative.

In the gradient descent method, a start point $\mathbf{x}_0 \in \Omega$ is chosen and a gradient of objective function in this point is calculated: $\mathrm{grad} f(\mathbf{x}_0, \mathbf{p})$. This gradient is used to determine the direction of the steepest descent of the optimization function. Moving a step further in said direction, an improved value of objective function is obtained, new gradient is calculated in the point $\mathbf{x}_1$ and the algorithm then proceeds iteratively as illustrated in Fig. 3.

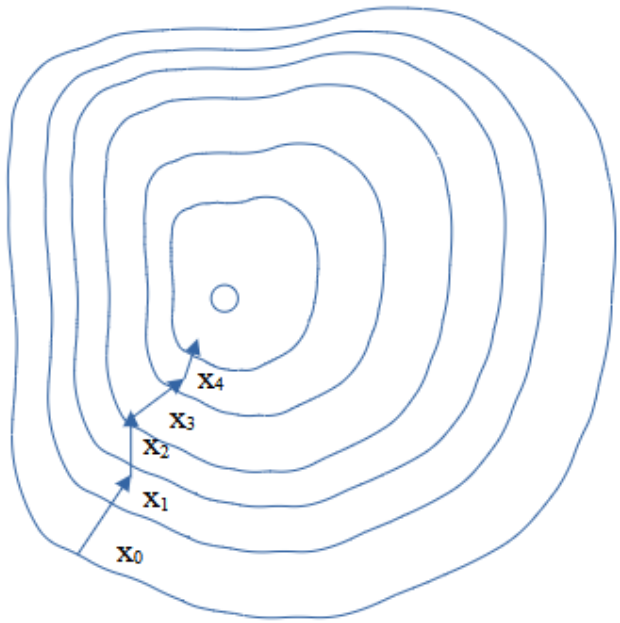


Fig. 3: Gradient descent method

Termination of the process is triggered either by going beyond the scope of admissible solutions or a decreased value of objective function, which indicates it has reached a nearby local extremum. Whether a local extremum is indeed the sought global extremum is a matter of further investigation. If objective function is unimodal or monotonous and decision space is convex then any found local extremum is provided to be global extremum as well. Otherwise, global optimum can perhaps be found by restarting the algorithm a couple of times changing the start point. Step size can be decreased in order to examine the area of the decision with increased density. Apart from the choice of location of a start point X_0 and step size, the third factor influencing the quality of results is the method of determining the gradient of objective function.

The steepest descent method has found many modifications, one of the most important was introduction of penalty function (which should be differentiable) to handle constraints violations. Optimal gradient method is another modification, in which the direction of search is changed only if the value of optimization function wasn't improved, otherwise steps are taken in a constant direction. In many cases this method provides faster convergence to an optimal solution, which is achieved thanks to a reduction in the number of times gradient of function needs to be computed.

Gauss-Siedel method was first mentioned by a German mathematician Carl Friedrich Gauss in 1823 in a letter to one of his students and later published by another German

mathematician Philipp Ludwig von Seidel in 1847 in his book *Abhandlungen der Mathematisch-Physikalischen Klasse der Königlich Bayerischen Akademie der Wissenschaften*. It is similar to previously mentioned optimal gradient method, although it does not involve computation of gradients. Starting point $X_0 \in \Omega$ is chosen and then steps are taken in the direction of the axis of the coordinate system unless the objective function stops improving. In this case search is continued from a point yielding the best value of objective function in the direction of next axis of coordinate system. The process is then continued iteratively.

A very simple method of optimization is a random search method, also called Monte Carlo method. In this method, a set of randomly selected points from a feasible solution space is chosen, values of objective functions are calculated, and the best solution is chosen. The method is not very effective and requires many objective function computations, but it has some advantages: thanks to its simplicity Monte Carlo method is very easy to implement and doesn't impose any assumptions on the optimization problem. There are some modifications of random method, including narrowing down the search area in which the points are randomly selected as the process proceeds utilising previously obtained solutions. Monte Carlo method can also be coupled with a gradient method to form a method in which start points are drawn at random and then optimization is continued according to the gradient descent method. This produces an efficient alternative to a classic gradient method as thanks to having multiple start points, in this approach the algorithm is less likely to get stuck in a local optimum.

Newton Method is another optimization method in which at first a start point $\mathbf{x}_0 \in \Omega$ is selected and then an improved solution is sought in the direction $\mathbf{d} \in \Omega$ in an iterative process. Direction $\mathbf{d}$ is obtained by using second-order Taylor expansion of objective function *f* around $\mathbf{x_0}$ and requires calculation of gradient and Hessian of function *f* [13]. Direction calculated this way uses additional information on curvature of optimization function, represented by Hessian and thanks to this feature can take a more direct route towards optima compared to steepest descent method which uses only information provided by gradient of function. For one dimensional problem this is reduced to calculation of first and second derivative of function *f*. Consequently, to use Newton method optimization function must be twice differentiable, which limits the scope of its applicability, although there are some approaches to overcome this, most notably quasi-Newton methods, which attempt to use information on second derivative of objective function without calculation of Hessian matrix.

There are also more recent deterministic methods, such as Limited-Memory Broyden-Fletcher-Goldfarb-Shanno Algorithm (LM-BFGS) [14], which belongs to a family of quasi-Newton methods for unconstrained optimization. LM-BFGS is acclaimed for its efficiency in dealing with large-scale problems and is often used for parameter estimation in machine learning. The algorithm was also modified to handle constrained problems [15].

Apart from deterministic methods, which could be understood as hard computing methods, second large group of optimization methods are heuristic methods which belong to the group of soft computing techniques [16]. The word *heuristic* comes from Greek *heuriskein* and means *to find, find out, discover,* or *invent*. This group of methods aims to find a solution which might not be strictly optimal, but can be found when other, direct methods fail to deliver for various reasons. Problems solved by heuristic methods often cannot be solved using deterministic approaches at all or solving them would be too demanding. For problems when

methods of finding solutions are unknown, vague, or timely, heuristic methods offer a convenient alternative, trading off qualities such as accuracy, precision, completeness in exchange for speed, versatility, and applicability. It should be noted that it is still possible and likely to produce satisfying results in terms of aforementioned sacrificed qualities when using heuristic methods. The term metaheuristics is often used to refer to algorithms solving sophisticated optimization problems [17], [18].

Many heuristic algorithms include some aspects of stochastic optimization in a way that solutions obtained by them are dependent on some form of random numbers. Another popular feature of many heuristic algorithms is their population-based structure so that they operate on a set of solutions at the time rather than on a single solution. A large part of metaheuristics is inspired by nature. Classification of metaheuristics and some examples of algorithms is presented in the Fig. 4. Nevertheless, it must be understood that the number of metaheuristic algorithms available nowadays is much larger than presented and new algorithms are proposed every day.

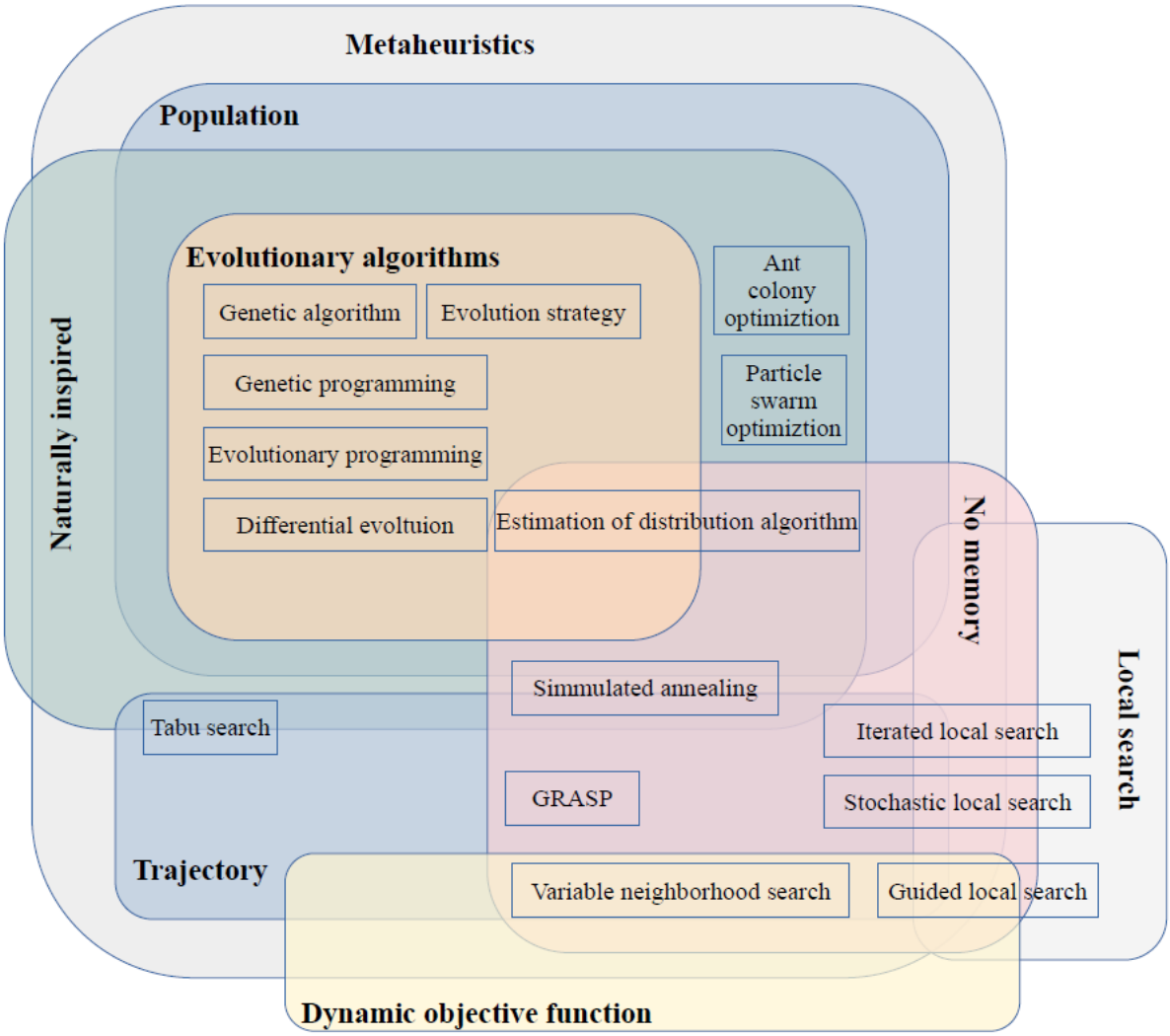


Fig. 4: Metaheuristics classification on an Euler diagram

In order for any algorithm attempting to solve a problem by search it is necessary for it to maintain an ability to explore and exploit the search space well. “Exploration and exploitation are the two cornerstones of problem solving by search” is an observation made by authors in their paper from 1998 concerning said mechanisms in evolutionary algorithms [19]. Their

insights are still valid and influence researchers dealing with heuristic approaches to optimization to this day. The process of exploration provides an ability to examine new areas of the search space whilst the process of exploitation provides extensive local search in vicinity of particularly desirable solutions. It is vital for a good algorithm to balance out the impact of both these features to produce satisfying results [20]. Genetic algorithms are understood to work remarkably well as general purpose (domain independent) optimizers thanks to a good ratio between exploitation and exploration mechanisms [21].

Evolutionary algorithms (EAs) are bio-inspired, population-based metaheuristics with stochastic features which tend to mimic biological evolution mechanisms such as selection, mutation, reproduction, and recombination. These features in EAs are referred to as operators. EAs can be understood as an algorithmic interpretation of Darwinian concept of survival of the fittest. Solutions of the optimization problems are individuals in a population whose quality is assessed by the value of optimization function. Applying evolutionary operators leads to evolution of the population and therefore improved quality of solutions. Thanks to their robust behaviour and flexible nature EAs have been successfully applied to solve a wide range of complex problems in many fields [22] including engineering [23], computer science [24], natural science, mathematics, earth science, finance and economics, social sciences, industry, management and biological science. The general idea of EAs is shown in the Fig. 5.

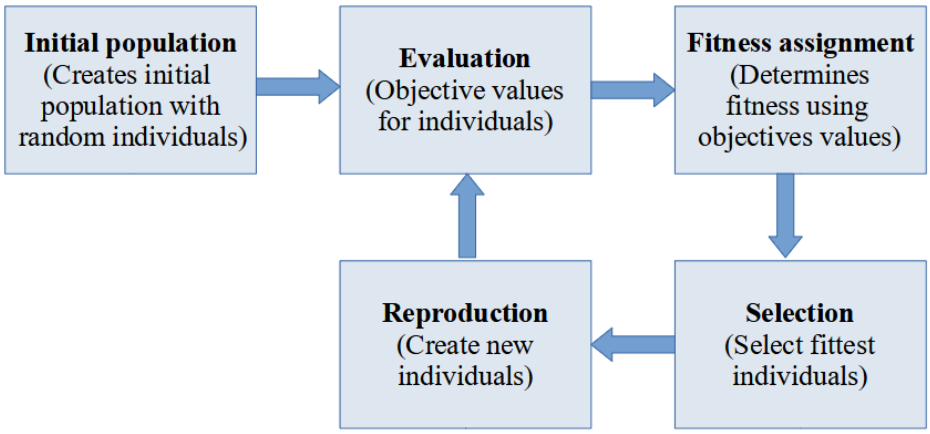


Fig. 5: General idea of EAs

There are several types of EAs different in the way they represent solutions in genetic analogies, varying in implementation and specific purpose. The most important are:

- genetic algorithm,
- genetic programming,
- evolution strategies,
- evolutionary programming,
- differential evolution.

Genetic algorithm (GA) first emerged in the 60s and 70s in the works of Holland [25]–[27] and is inspired by natural processes of evolution happening over time. Their ultimate goal is maximizing individuals' fitness to survive in their natural environment and objective function (fitness function) is used to measure their ability to fit in the environment. Despite utilising stochastic elements, GA does not work at random but effectively use past experience in the form of previous solutions. GA traditionally adopted binary encoding of solutions, although

later works introduced other types of encoding including real numbers [21]. GA uses recombination (crossover) as a major operator whilst mutation operator plays a minor role. Selection operator is used in a probabilistic form. There are multiple schemes of crossover, mutation and selection, and the basic pseudo code of GA is shown in the Fig. 6.

```
i ← 0

Generate initial population P0;

Evaluate P0;

while termination condition

        i ← i+1

        Select Gi from Gi-1        // selection operators

        Change Gi                  // recombination and mutation

        Evaluate Gi

end-while
```

Fig. 6: Pseudo code of GA

Genetic Programming (GP) was introduced in 1992 [28] and is understood either as a separate paradigm of EAs or a type of a genetic algorithm. The main difference between GP and GA is the way attributes are represented. In GP attributes represent instruction sets or programs and algorithm generates computer programs whose ability to solve computational problems is measured as objective function. Characteristic feature of GP is tree-based encoding, with trees consisting of functions and terminals. Functions available in most general-purpose programming languages such as arithmetic and Boolean operations, mathematical and recursive functions, loops, and conditionals are usually available in GP and supplemented with domain-specific functions. Terminals can be interpreted as functions without arguments and usually are variables or parameters.

Evolution strategies were developed in 1973 [29] to answer difficulties with dealing with hydrodynamical optimization problems, which involved particularly hard objective functions: complex, multimodal and non-differentiable. The new algorithm was very simple, and the main idea was to apply random changes (mutation) to a selected solution (a single parent) to generate a single offspring. Offspring would then be compared with parent and superior of the two would become parent in the next generation. Later the mutation process was improved to adjust standard deviation of solutions in a deterministic manner so that the strategy could converge to a global optimum. Other modifications of the method were proposed over the years, including generation of more than a single offspring, using more than a single parent, more complex selection mechanisms to determine the new parents and self-adaptive mutation parameters.

In the 60s Evolutionary Programming was proposed as a way to achieve artificial intelligence by adaptive behaviour [30]. Unlike other EAs which mostly try to emulate genetic mechanisms of transmission of information, this one focuses on behavioural relationships between parents and offspring. In Evolutionary Programming, several types of mutation

operators are used but no recombination is allowed. Each parent in the population generates only one offspring and selection mechanisms are probabilistic, rather than deterministic or random.

As shown, there is a variety of EAs to solve a range of optimization problems. Even though many of these algorithms were developed over 50 years ago, their variants and modifications along with new emerging EAs are still used to this day.

Apart from evolutionary approaches, one of the most popular and well-established optimization methods among heuristic optimization techniques is Particle Swarm Optimization (PSO). The idea was introduced in 1995 by Kennedy and Eberhart [31] and has since then brought significant attention of researchers generating over two dozens of variants of the original algorithm and numerous hybrid algorithms combining PSO with other methods [32]. The technique attempts to mimic the behaviour of swarms of animals such as flocking birds or schooling fish, who are organized in terms of their direction, speed and spacing between each other. Unlike many EAs, the algorithm does not use mutation or recombination mechanisms and does not require encoding/decoding of information but relies real-number randomness and global communication between particles representing solutions. Swarming particles in search of an improvement of objective functions move in a direction described by two major components each: one deterministic and one stochastic. Every particle is attracted by position of a global best solution and a best solution found previously by itself. Movement of particles is also influenced by a random factor. PSO is acclaimed for its simplicity and flexibility and has found application in many branches of science and engineering, aside from optimization it was successfully used for training of artificial neural networks and a broader term of Swarm Intelligence is often used for these non-optimization related purposes.

Another biologically inspired although not evolutionary group of optimization algorithms are Artificial Immune Systems (AIS) [33]–[35]. These algorithms try to mimic processes of natural immune systems, including immunological learning and memory, immune response, antigen-antibody interactions, cell division and somatic hypermutation. It has found a wide range of application in fields such as image processing, pattern recognition, classification, and clustering.

Other popular nature-inspired algorithms which found its application to optimization problems include Simulated Annealing, Firefly Algorithms, Cuckoo Search, Flower Pollination Algorithms, Ant Algorithms, Bee Algorithms, and Harmony Search among others. Overall, both deterministic (hard computing) and heuristic (soft computing) methods can be used for optimization of mechanical systems although due to certain limitations of analytical methods and flexibility and versatility of soft computing methods on the other hands, the latter seem to have a drawn a particular attention of researchers dealing with optimization of mechanical systems.

Due to the lack of a universal optimization algorithm, a vast number of new algorithms inspired by natural, social, cultural, or physical processes are created every day. These algorithms often exhibit similar structure with only their tiny elements modified. In many cases new algorithms are effective to solve a specific group of problems. According to many researchers such an approach is unnecessary and new algorithms do not provide much of a novelty, instead they copy well established approaches and frame them differently, focusing on new metaphors rather than increased quality of algorithms [36]–[39].

2.3. **Multiobjective optimization**

All the optimization methods discussed so far were in essence devoted to single objective problems. Often in real optimization problems many objectives need to be improved simultaneously, for example we want to increase acceleration of a car, reduce fuel consumption and cost – these are three conflicting objectives. Such problems require a different approach and therefore other optimization methods. Results of a multiobjective optimization task consist of a set of solutions, rather than a single best solution and in many cases optimization techniques to find these solutions are more complicated and require more computational effort. In fact, it is a common practice during the design process to reduce complex problems with conflicting goals to scenarios, in which systems are geared only towards a single one of the objectives. This can be done in many ways, for example excessive objective functions can be introduced in the optimization problem in the form of constraints. Alternatively, multiple objectives can be aggregated into a single synthetic objective using mathematical operations, which may require applying *a priori* arbitrary weights to the objectives. These methods certainly can simplify a complex multiobjective problem and provide decision maker with a single solution using relatively simple techniques nonetheless this comes at a cost. Simplifying a problem results in the reduction of information obtained after the optimization, losing on the knowledge of trade-offs between objectives and on a set of alternative solutions. Moreover, having a set of optimal solutions allows decision maker do draw conclusions regarding interdependencies between objectives, design variables, and constraints. Not having to choose preferences concerning objectives *a priori* presents an opportunity to articulate them with a more complete information *a posteriori* based on a set of Pareto optimal solutions (Fig. 7).

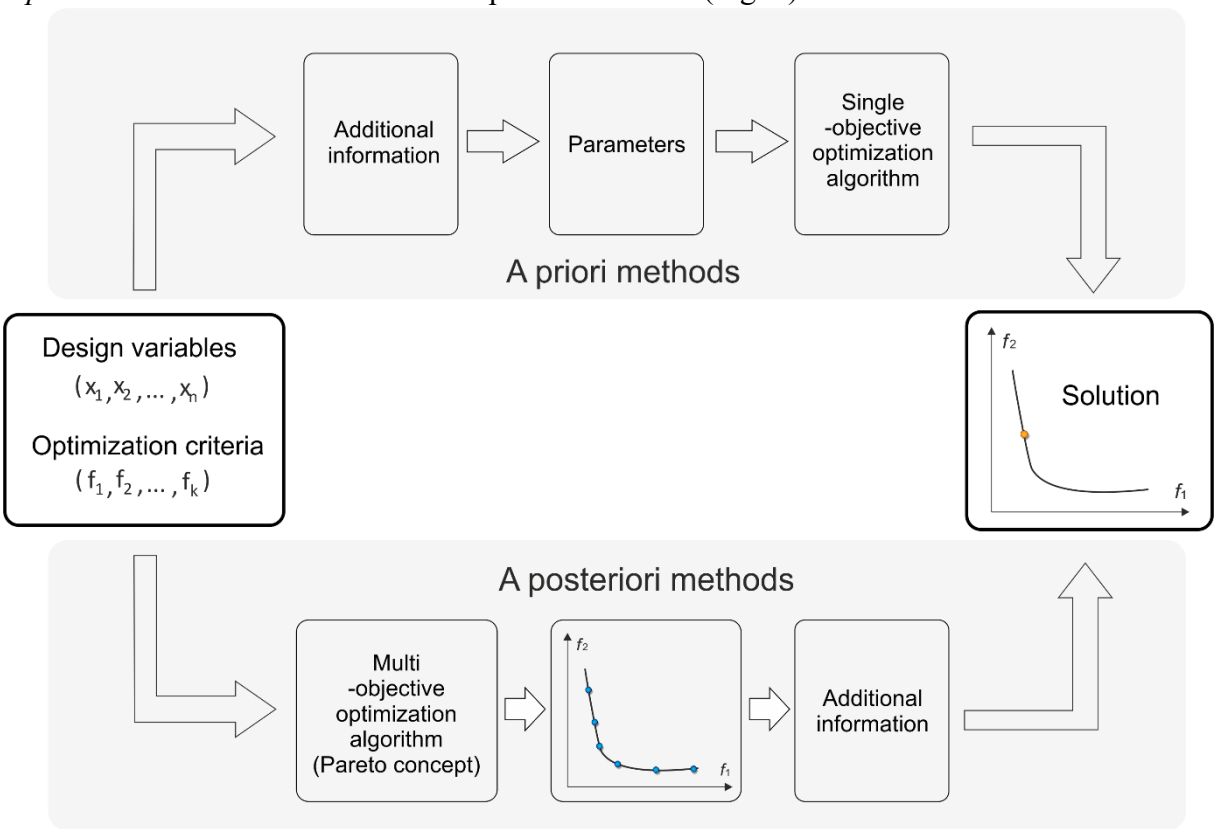


Fig. 7: *A priori* and *a posteriori* approaches to multiobjective optimization

The concept of Pareto optimality uses terms such as: dominated solutions, non-dominated solutions (also known as: Pareto-optimal, efficient, or non-inferior) and incomparable (neutral) solutions. Taking into consideration two vectors **x** and **y** in the admissible solution space and assuming minimization problem:

• **x** strongly dominates **y**, if:

$$\forall_{i\in\{1,2,\ldots,k\}}: f_i(\mathbf{x}) < f_i(\mathbf{y}) \tag{9}$$

• **x** dominates **y**, if:

$$\forall_{i\in\{1,2,\ldots,k\}}: f_i(\mathbf{x}) \leq f_i(\mathbf{y}) \ \wedge \exists_{j\in\{1,2,\ldots,k\}}: f_j(\mathbf{x}) < f_j(\mathbf{y}) \tag{10}$$

• **x** is neutral (incomparable) relative to **y**, if:

$$\exists_{i,j\in\{1,2,\ldots,k\}}: f_i(\mathbf{x}) < f_i(\mathbf{y}) \wedge f_j(\mathbf{x}) > f_j(\mathbf{y}) \tag{11}$$

In other words, **x** dominates **y** if **y** is not better in any objectives and **x** is better in at least one objective. The relation can mathematically be written as $\mathbf{x} \preccurlyeq \mathbf{y}$. The domination relation is shown graphically in the Fig. 8.

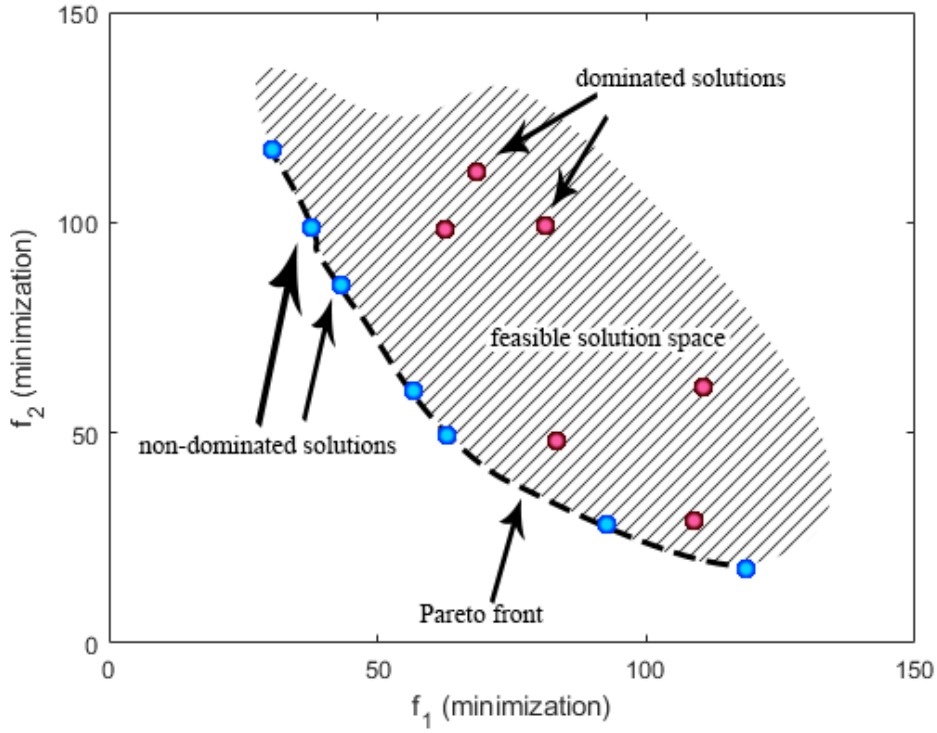


Fig. 8: Domination relationship between solutions

Fig. 9 shows the relationship between the vectors of solutions. Solution a is dominated by x (in other words solution x dominates a), solution x weakly dominates solution d, whereas solutions b and c are neutral (incomparable) relative to x.

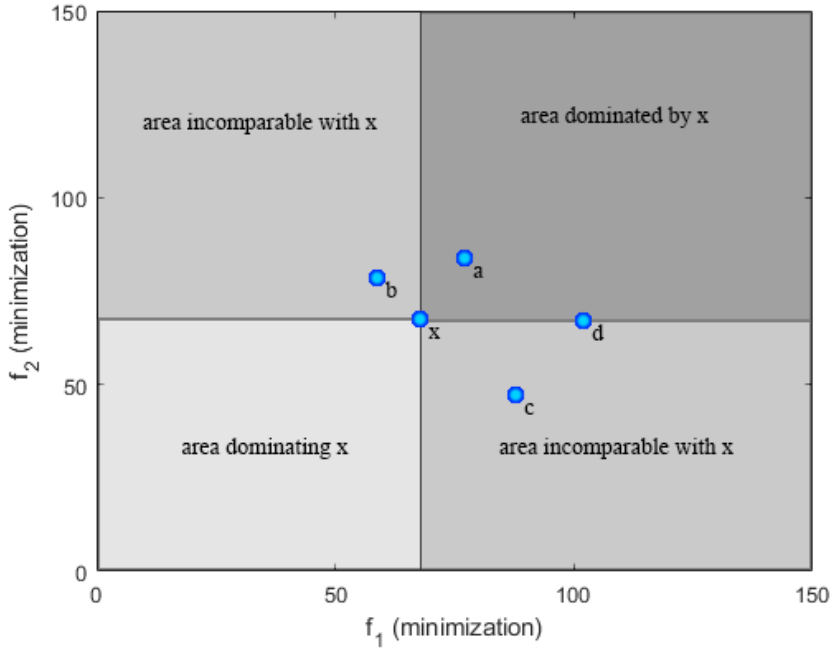


Fig. 9: Domination relationship between solutions

The set of all Pareto-optimal solutions creates a so-called Pareto front (Fig. 8). If three criteria are considered, solutions belonging to the Pareto front define a surface. Optimization methods based on Pareto approach, belong to a group of *a posteriori* methods. On the basis of a set of solutions it is possible to select a solution according to established preferences.

Methods of multiobjective optimization are intensively developed since the end of last century. Classical methods such as Weighted Sum Method, ε-Constraint Method, Weighted Metric Methods, Benson's Method, Goal Programming Method and Interactive Method all aim to transform multiobjective problem into a single-objective problem and have significant limitations, such as [40]:

- they only find a single solution during a run of an algorithm,
- they face difficulties dealing with nonconvex problems and
- they require prior assumptions on the problems in the form such as weights or target values.

Newly designed algorithms often have biological inspiration, or such elements are integrated into deterministic algorithms to eliminate some of their drawbacks.

Significance of the population-based methods can be seen especially for the multi-criteria optimization problems, because such an approach naturally corresponds to the challenges of multi-criteria optimization, in which a set (population) of solutions is sought. Most multi-criteria evolutionary algorithms, using Pareto approach, utilise the methods of comparing feasible solutions within the population for effective optimization, for at most three criteria. These include the following groups:

- methods, which refer to the number of individuals by which particular solution is dominated (dominance rank),
- methods, which refer to the number of individuals which particular solution dominates (dominance count),
- methods, where each of the individual is categorized on the basis of the membership to the separate fronts (dominance depth).

A significant drawback of these approaches is their low efficiency for problems with more than 3 objective functions. In high-dimensional solution spaces, newly found solutions are often incomparable and therefore the algorithms have trouble applying comparison mechanisms such as selection, which are essential in most of the optimization problems, both single- and multiobjective. The reason for increased number of incomparable solutions in a set is the increase of ratio between non-dominated and dominated solutions along with the increase of number of objectives. Practically, for problems with 4 or more objectives, the entire set of solutions is often non-dominated and therefore solutions are incomparable with each other, which poses a significant challenge for multiobjective optimization algorithms. This limitation can be confronted by a unique approach to soft computing optimization involving elements of game theory as stated in the thesis of this dissertation.

Overall, multiobjective optimization algorithms aim to achieve two main goals: place found set of solutions close to the true Pareto front and at the same time achieve a diverse and widely spread set of solutions. It was proven by Purshouse and Fleming [41] that these two objectives are in fact contradictory and usual genetic operators fail to achieve aforementioned two features simultaneously.

Elitist Non-dominated Sorting Genetic Algorithm or NSGA-II, introduced in 2002 by Deb [42], is still one of the most widely used algorithms to solve multiobjective optimization problems. Its popularity can be demonstrated by the fact that in computational software MATLAB it serves as a basic tool for multiobjective optimization in Global Optimization Tool (function *gamultiobj*). MATLAB, according to company brochure [43] is used by over 4 million people, primarily engineers and scientists worldwide.

The three main principles of the algorithm are:

- elitism mechanism preserving the best solutions,
- explicit diversity preserving mechanism, and
- focus on non-dominated solutions.

During the course of the algorithm, at any generation i, a new offspring population O_i is created on the basis of parent population P_i using common genetic operators. Parent and offspring populations have the same size and constitute a temporary population T_i with double size. Population size may not be changed, so a selection of solutions must be performed. New population N_i is filled with non-dominated solutions from population T_i. These solutions are removed from T_i and therefore new previously dominated solutions become non-dominated. Another set of non-dominated solutions is then transferred from T_i to N_i and so on until they fill the number of slots dictated by population size. When the number of slots available in the N_i is lower than the current non-dominated set size in T_i then instead of discarding excessive solutions, only solutions which improve diversity in N_i the most are transferred to fill the remaining slots. The rest of T_i is removed and N_i becomes parent population in the next step.

The metric which manages measuring influence of solutions on diversity of population used in the algorithm is called crowding distance and is expressed as an area between a solution and the closest other solutions in the set. The idea of NSGA-II is illustrated in the Fig. 10.

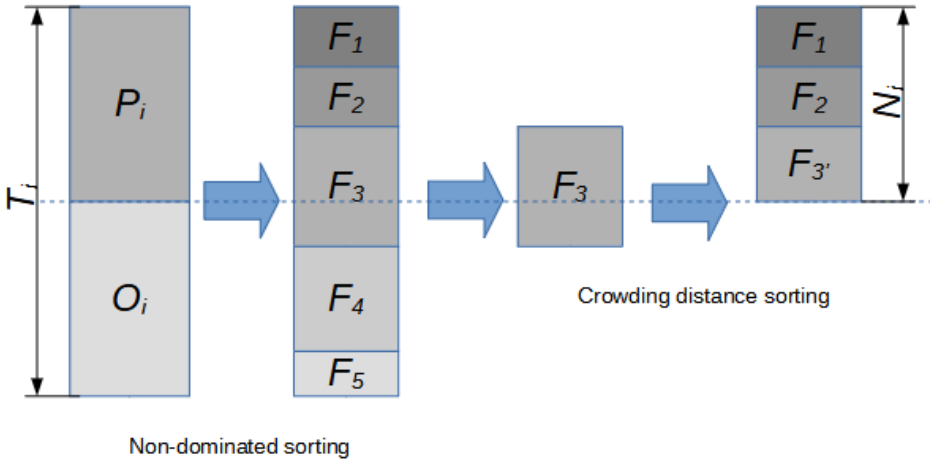


Fig. 10: The idea of NSGA-II

The algorithm have found a numerous application such as machining processes [44], planetary gearbox optimization [45], workshop scheduling [46], filament winding machine optimization [47], electrical distribution network optimization [48], optimization of parameters in control systems [49].

NSGA-II has found multiple modifications and variants until its author in 2014 proposed a new Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach called Many-Objective NSGA-II or NSGA-III to specifically address difficulties of original NSGA-II in handling problems with a large number of objectives [50]. The algorithm has been used to solve constrained and unconstrained optimization problems within a range of 2 to 15 objectives, including test problems with convex, concave, disjointed, normalized, scaled, and degenerated Pareto fronts, considered difficult for many optimization algorithms. NSGA-III was compared with other multiobjective optimization algorithm MOEA/D on a set of challenging benchmark functions, including the impact of population size on the quality of results [51].

The idea of NSGA-III is in many ways similar to its predecessor, although it is significantly different in the way solutions responsible for maintaining diversity are selected. Offspring is generated using genetic operators and consists of equal numbers of elements as parent population. Both sets are combined and constitute a population of double size compared to the parent populations, which ensures preservation of elite solutions, but requires selection and discarding of half of the solution. Non-dominated sorting is used to transfer solutions to an offspring population and instead of crowding distance sorting, a new, more complex approach is proposed. A set of well-spread reference points is used to select solutions based on the diversity they offer to a set of solutions obtained. Reference points can either be preferentially supplied by user or if there is no additional information, they can be automatically generated.

NSGA-III was proven superior to MOEA/D in many, but not all cases of analysed test functions and number of objective functions. The implementation of the algorithm was not

shared by original authors, however unofficial implementations, including a C++ implementation [52] are available.

2.4. **Finite element method in optimization**

Values of objective functions in optimization of mechanical systems often cannot be expressed in the form of analytical functions and have to be computed using numerical methods. There are multiple numerical methods suitable for determining quantities of mechanical systems: finite element method (FEM), boundary element method (BEM) and finite difference method (FDM) are the most popular, but the first one is particularly important due to its versatility and availability in CAE software. FEM can be used to solve one-, two- and three-dimensional problems in the fields such as structural analysis, heat transfer, fluid flow, magnetic flux and others [53]. Typically, in optimization problems objectives related to features of mechanical systems such as deformations, stresses, temperatures, dynamic response, modal frequencies can be obtained automatically using FEM-based CAE software.

From a mathematical perspective, FEM can be seen as a numerical tool for solving partial differential equations governing problems which describe behaviour of mechanical systems. The method will be briefly described on the basis of a static structural stress and deformation analysis. Other types of problems to which FEM can be applied should be understand relatively easy afterwards. Problems concerning other physical fields will be discussed in further chapters.

A body occupying a volume V and bounded by surface S in 3-dimensional x, y, z space is described by boundary conditions (including loads) and material distribution. The simplest, yet important material model used in FEM analyses concerns linear-elastic, isotropic material. Assuming static problem, small displacements and strains, behaviour of the deformable body under mechanical loads is governed by following equations:

- Geometric relations:

$$\varepsilon_{ij} = \frac{1}{2}(u_{i,j} + u_{j,i}) \tag{12}$$

- Constitutive law (Hooke's law):

$$\sigma_{ij} = \lambda\delta_{ij}\varepsilon_{kk} + 2\mu\varepsilon_{ij} \text{ or } \sigma_{ij} = C_{ijkl}\varepsilon_{kl} \tag{13}$$

- Equation of equilibrium:

$$\sigma_{ij,j} + b_i = 0 \tag{14}$$

where ε_{ij} are strain tensor components, u_i are displacement components, σ_{ij} are stress tensor components, λ and μ are Lamé's parameters, C_{ijkl} is stiffness tensor and b_i are volume forces.

The basic goal of the FEM analysis is to determine displacement field $\mathbf{u}(x, y, z)$ representing displacement at any point of the body. Further information on strains and stresses can be obtained based on the displacements. Overall, body can be modelled in FEM as: a 3D body, 2D plane strain, 2D plane stress, shell, frame, or a truss. Vector of design variables $\mathbf{x} = [x_1, x_2, \dots, x_n]$ describes the shape or topology of body, for example its thickness, cross sectional area, moment of inertia, characteristic dimensions etc. Optimization task involving FEM analysis can be expressed as:

optimize (maximise or minimise)

$$f(\mathbf{x}, \mathbf{U}) \tag{15}$$

under m equality constraints

$$g_i(\mathbf{x}, \mathbf{U}) = 0, i = 1,2, \dots, m \tag{16}$$

and p inequality constraints

$$h_j(\mathbf{x}, \mathbf{U}) \geq 0, j = 1,2, \dots, p. \tag{17}$$

where $\mathbf{U}$ is a nodal displacement vector of size *ndof* used to determine displacement field $\mathbf{u}(x, y, z)$. *ndof* expresses the number of degrees of freedom in the system. In this example a single objective optimization problem is displayed, but a multiobjective problem consisting of a vector of objective functions in place of (15) can be considered accordingly. Partial differential equations of equilibrium describe the relation between $\mathbf{u}$ and $\mathbf{x}$. Employing FEM these differential equations may be transformed into a system of linear equations for $\mathbf{U}$:

$$\mathbf{K_M}(\mathbf{x})\mathbf{U} = \mathbf{P}(\mathbf{x}) \tag{18}$$

where $\mathbf{K_M}$ is a square stiffness matrix of size *ndof* x *ndof* and $\mathbf{P}$ is a load vector of size *ndof*. Functions f, g_i and h_j depend on vector of design variables $\mathbf{x}$ both explicitly and implicitly through $\mathbf{U}$ as shown in equation (18).

Overall, FEM requires a geometrical model on which a mesh consisting of elements, nodes and shape functions is imposed. The quality of the mesh affects the accuracy of the solutions in the method. The geometrical model needs to be supplemented with a set of boundary conditions describing loads and displacements on specified part of the boundary. Boundary conditions are related to supports, fixed ends, given displacements, tractions, body loads and point loads. Strain-displacement and stress-strain relations must be known. The latter is expressed by generalized Hooke's law for linear elastic materials and for isotropic materials the two material properties governing the relation are Young's modulus E and Poisson's ratio ν. On the basis of these information local stiffness matrices for elements are calculated and assembled into a global stiffness matrix $\mathbf{K_M}$, which is symmetric, sparse, and banded matrix. Having determined stiffness matrix $\mathbf{K_M}$ and global load vector $\mathbf{P}$, a global displacement vector $\mathbf{U}$ can be calculated from equation (18) using matrix algebra and equation solving methods [54].

2.5. **Visualisation and decision making**

Results of multiobjective optimization are in the form of set of solutions, each solution being a vector of length corresponding to the number of objective functions. In general, visualisation of any data is possible only in two dimensions, therefore displaying multivariate data sets in a graphical form requires generation of a two-dimensional representation of a multi-dimensional data. Effective visualization procedures are vital as the graphical interpretation of data acquired from the optimization task is a link supplying the decision maker with the knowledge from the analyst. Some techniques such as scatter plots can be easily employed to present three-dimensional data sets simply by adding a third axis. It might be possible to represent more dimensions in the form of colour or shape of markers, but with the increase of dimensionality, the visual representation becomes vague. Overall, visualisation of a multivariate data sets can be achieved in two ways, either by reduction of the dimensionality of a problem or by symbolic representation as an object (an icon) [55].

Selected visualisation techniques will be presented using a set of non-dominated solutions obtained during the course of optimization of a microactuator with respect to 6 objectives. The set consists of 2891 Pareto-optimal solutions and therefore might be challenging to present in a way which provides complete, clear, and precise information on the values of optimization functions. The problem from a mechanical perspective is further discussed in section 5.3 meanwhile at this point only the visualisation of results is considered.

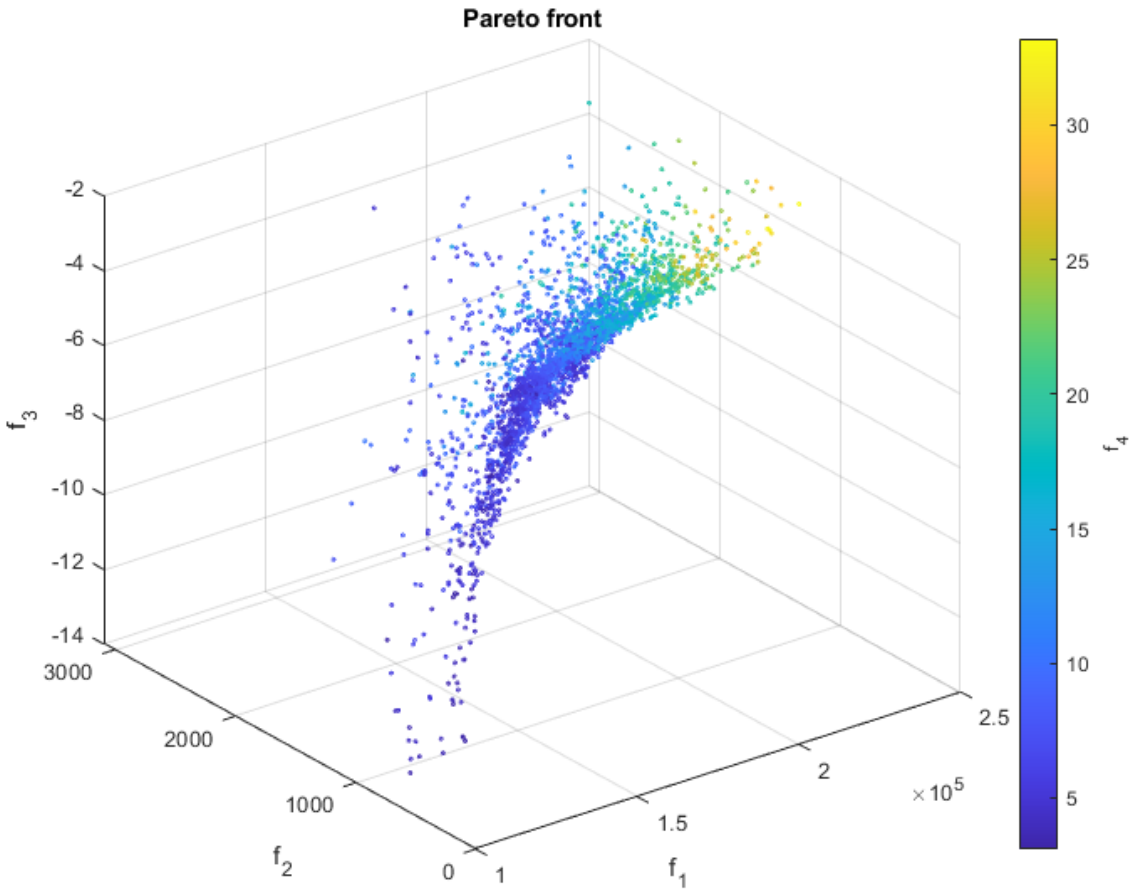


Fig. 11: 3D scatter plot with f_4 mapped in colour

In the Fig. 11 the information is presented in the form of a 3D scatter plot with first three objectives mapped onto the axes of a plot and location of points corresponding to the values of optimization function with fourth objective function represented by a colour of point. This method does not provide information on the value of objective f_5 and f_6 and therefore is incomplete although it might be helpful to provide insight on trade-offs between selected objectives and range of values of obtained solutions. To supplement the information with an addition objective, the size of the markers might be decided based on the value of objective f_5.

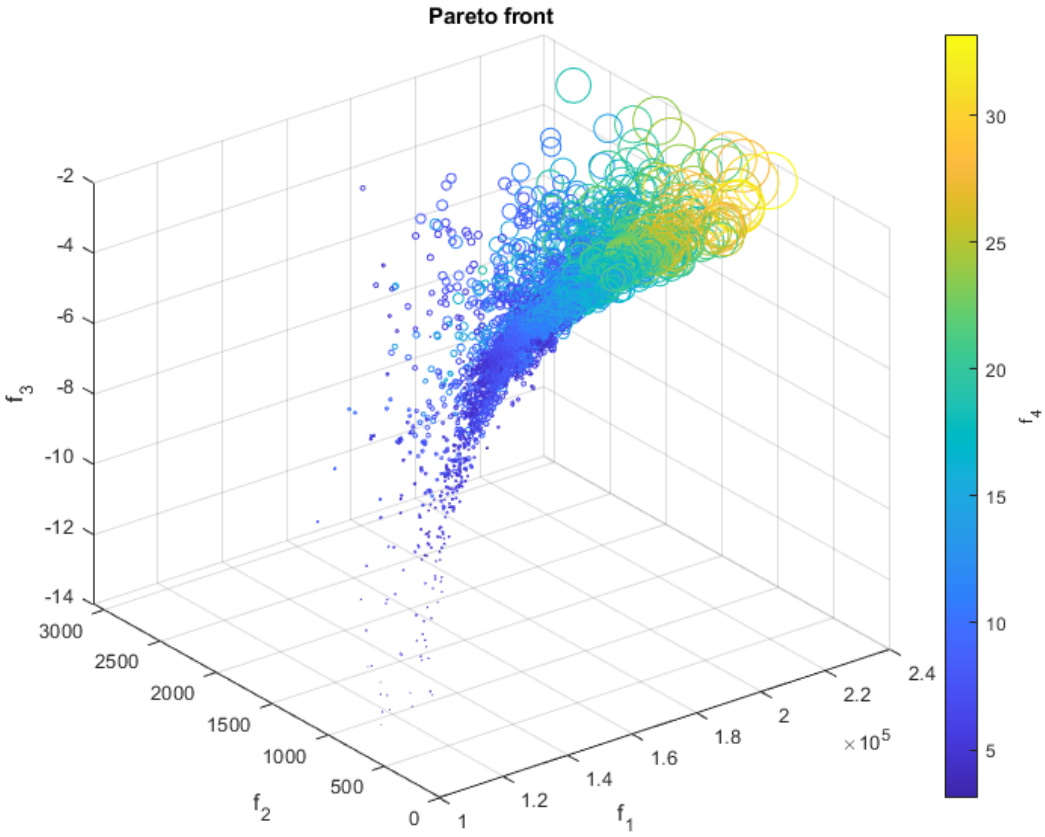


Fig. 12: 3D scatter plot with f_4 mapped in colour and f_5 mapped as size of the points

Another information is included in the Fig. 12, but the data is still neither complete nor precise as it does not include information on the value of objective f_6 and the size of the elements cannot be precisely measured and linked with a value. Partial information on the value of objective f_6 can be introduced to the plot by assigning ranges of values of f_6 to specific shapes of the marker and displaying them on the graph instead of circles. Data presented in this way is hard to handle when precise values are sought. It can be used only for the rough information on the trade-offs and ranges of objectives, but it would require additional, likely tabular information to read specific information on objective values of selected solutions.

Alternative approach to visualisation is utilising parallel coordinates. In this way, information on a high number of objectives can be presented in a relatively easy way providing precise and complete information, although it is limited to sets with a relatively small number of solutions, as the visual representation becomes cluttered with the increase of data points representing solutions. In case of objectives of a different magnitude of values, data requires normalization to a uniform range (e.g. [0,1]) to be presented effectively.

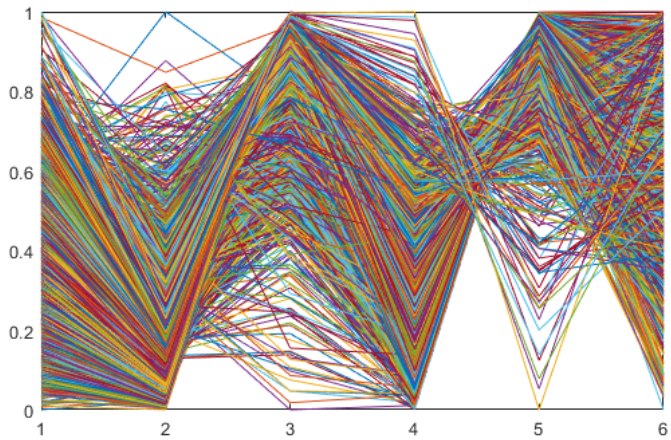


Fig. 13: Plotting in parallel coordinates.

In the Fig. 13 values of all 6 objectives and all 2891 solutions are presented after normalization of objective function values to a range of [0,1]. Due to a large number of solutions, the visualisation is cluttered and therefore vague. This can be overcome by selecting a lower number of solutions to be displayed, however at the cost of reducing completeness of information (Fig. 14).

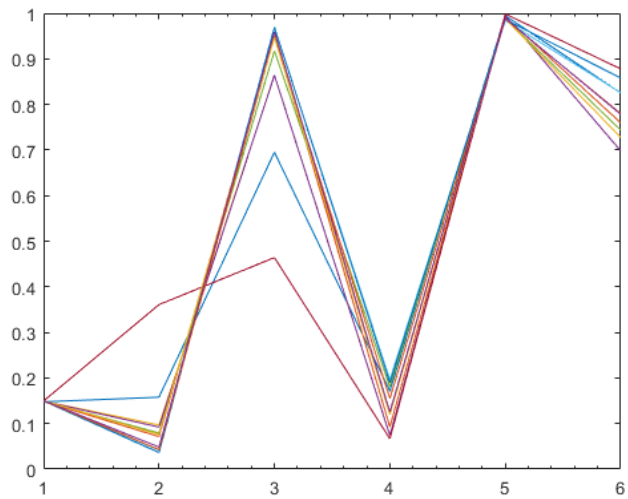


Fig. 14: Plotting in parallel coordinates (selected 10 solutions).

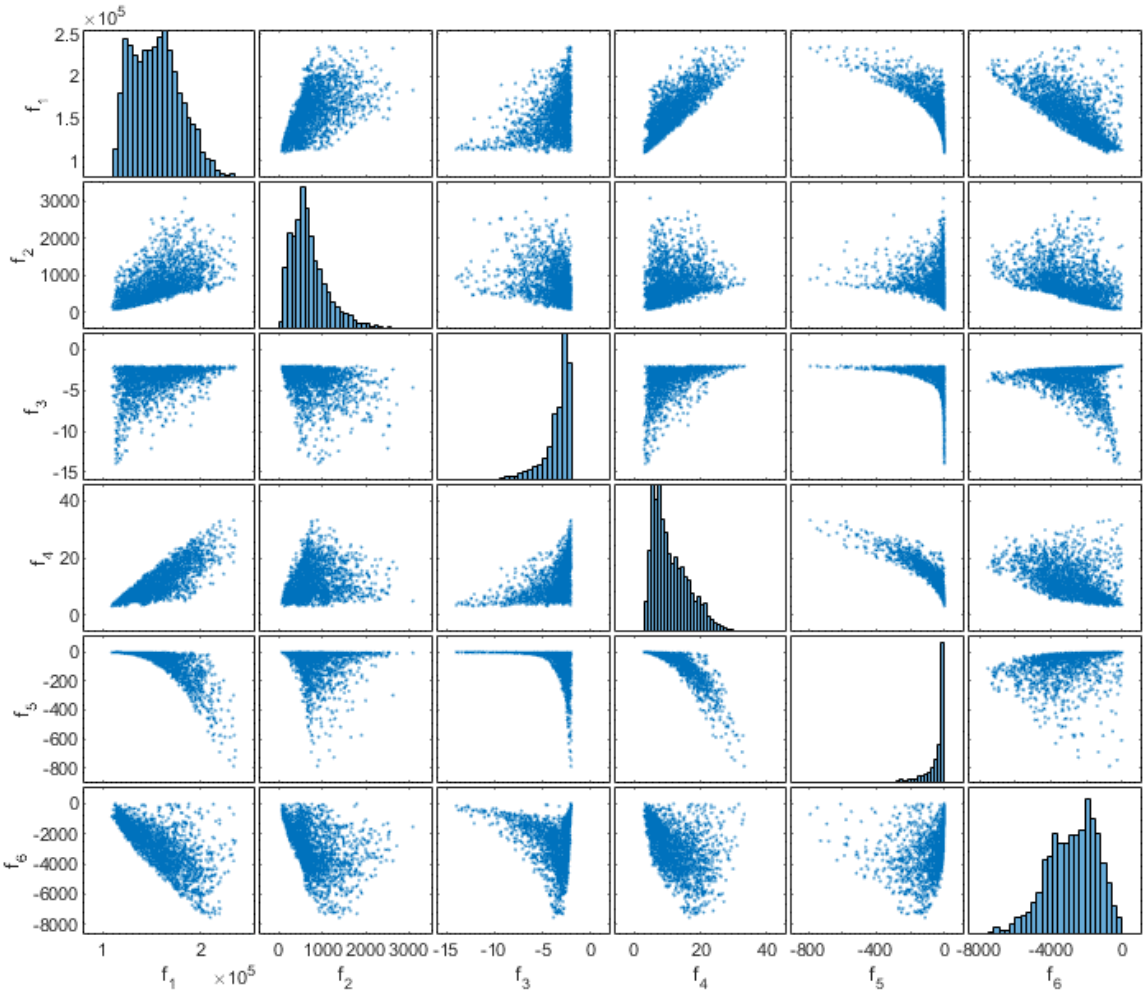


Fig. 15: Scatter plot matrix

Another approach to visualisation of multivariate datasets is using scatter plot matrix consisting of pairs of objectives, often supplemented with the histograms of objective function values (Fig. 15). This approach provides complete information on the values of objective and is an easy way to illustrate trade-offs between pairs of objectives as well as the number of solutions in selected range of values.

An interesting approach to visualisation of multidimensional data sets is utilising Kohonen's Self-Organizing Maps (SOMs) [56]. SOMs are used to produce a similarity graph of input data. High-dimensional sets of non-dominated solutions are translated into geometric relationships of their image points on a regular, usually hexagonal, 2D grids of nodes. It has many applications including visualization, clustering, and data mining.

Every SOM node (unit) is defined by its codebook vector, consisting of map weights. Size of codebook vectors is equal to a size of a single input data sample. Codebook vectors' values are initialized in the initial phase of establishing a network. Random or linear (Fig. 16) initialization are applied, the latter is proven to be more effective. In the training part of the algorithm input data is presented to the network and the best-matching unit (BMU) is chosen

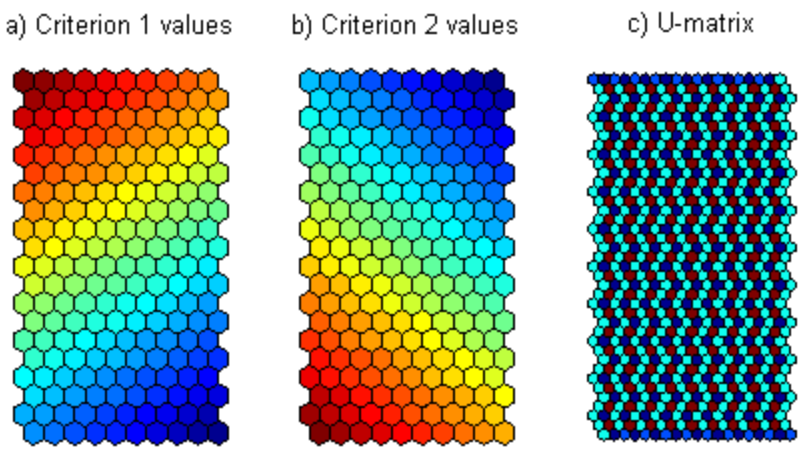


Fig. 16: Presentation of a 20×10 units SOM for the 2 criterion optimization before the training, linear initialization only, no input data presented to the network.

amongst all map units utilising a Euclidean distance as a measure to be minimized. BMU and its neighbours are modified towards an input pattern, therefore the topological order of data is maintained in SOM – the data located close to each other in the input space are close to each other on the map as well. SOM training is an iterative job, frequently involving rough training and fine tuning. Trained SOM can be displayed as a number of grids of coloured nodes. In case of optimization results visualization, each grid corresponding to a single objective function and nodes corresponding to particular solutions, with the value of criterion shown as unit's colour (Fig. 16 a) and b)). Often a measure of similarity between data in the neighbouring units is desired to be depicted, thereof another grid called U-matrix, showing the unified Euclidean distance between codebook vectors of the neighbouring units (Fig. 16 c)).

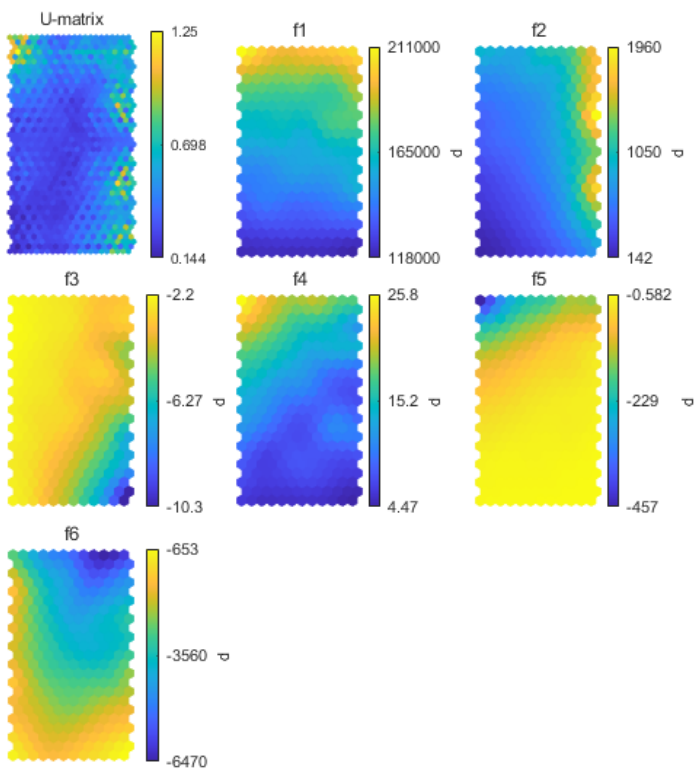


Fig. 17: Self-Organizing Map

In the Fig. 17 SOM visualisation of 6-objective Pareto front is shown. Such a representation of solutions helps with decision making in a way it enhances the process of exclusion and inclusion of solutions based on the range of values of objectives. Decision maker can decide to exclude all solutions above or below a certain value of optimization function. This can be conveniently done on SOM by marking area related to selected threshold on one of the maps and then disregarding the same area on the remaining maps. This process can be repeated for many objectives and values to narrow down obtained set of solutions to a smaller set with regards to additional established preferences. The process of decision making enhanced by SOM on the example of a search of compromise solution is further discussed in section 5.3.

In conclusion, decision making, as a step preceding optimization process is an essential part of the design process. Decision maker needs to be provided with information acquired during the optimization process presented in a way which enables him to each a decision regarding a final design of the system. This task is complicated and often requires employing visualization techniques to enhance making a decision. Many visualization methods are available, but it is often necessary to choose the most appropriate one depending on the information to be presented. In this dissertation, solved analytical and numerical mechanical optimization problems will be presented utilising scatter plot matrices, 3D scatter plots with colour mapping and SOM, based on self-developed and implemented procedures of generating maps and plots.

2.6. **Test problems for optimization**

The primary reason of developing new optimization algorithms is to solve optimization problems in a way which provides us with good quality of solutions and doesn't require an excessive computational effort at the same time. In case of real optimization problems, the solution is naturally unknown beforehand, which makes it hard to assess the performance of algorithm at the end of the process of optimization. In order to assess the performance of optimization algorithms test problems or benchmarks functions, preferably with known optimal solutions are used [40]. The main purpose of these functions is to test the performance of new algorithms, compare new algorithms with others and to understand the principles, strengths, and flaws of new algorithms. There are many requirements asked of these functions so that they could reflect real systems subjected to optimization. For this reason, some controlled difficulties in converging to the Pareto front and maintaining a satisfactory spread of solutions are introduced. In case of mechanical systems, the solution space is often multimodal and discontinuous. The latter feature of solution space is especially prevalent as far as optimization functionals are computed using FEM-based CAE software. These programs happen to face difficulties when trying to mesh complex geometries of certain variants of parametric models, resulting in a discontinuous solution space, as candidate solutions which failed to be meshed correctly should not be included in the solution space. It is therefore a vital feature to be included in test functions designed to assess performance of algorithms which are supposed to work with mechanical systems.

Furthermore, it is convenient for the test functions for multiobjective optimization problems to be scalable in terms of both the number of variables and the number of objectives. There are numerous test functions developed for problems with 2 or 3 objectives [57], but to perform a tests concerning higher solution space dimensions, scalable problems are required.

Test functions are supposed to be quickly evaluated (unlike most real mechanical optimization problems) and thus make it possible for a statistical, parametric, or non-parametric method of comparison of obtained sets of non-dominated solutions. Performing statistical tests requires multiple runs of an algorithm, with each run requiring a set number of test function calls.

Some of the test functions, which due to their aforementioned qualities were chosen to assess performance of a novel algorithm described in this thesis are described in detail in the following pages. In total 6 test functions were used for this purpose.

A representative set of 7 test functions, called *DTLZ Test Problem Suite* was introduced in [58] and was later expanded with 2 additional functions in [57]. DLTZ problems are fully scalable in terms of both number of objectives and design variables, which drew attention of many researchers investigating multiobjective optimization algorithms. Problems developed by Deb et. al are simple to implement, have known true Pareto-fronts and feature characteristics such as multimodality, convex shapes of fronts, bias and discontinuity of fronts [59]. For each DTLZ problem, the design variable domain is of $x_i \in [0,1]$.

Only functions which will further be used to assess performance of a multiobjective optimization algorithm developed in this thesis are described in this chapter:

- DTLZ1

DTLZ1 is a simple M-objective test problem with a linear Pareto-optimal front, all functions to be minimized.

$$\begin{cases} f_1(\mathbf{x}) = \frac{1}{2} x_1 x_2 \dots x_{M-1} \left(1 + g(\mathbf{x_M})\right), \\ f_2(\mathbf{x}) = \frac{1}{2} x_1 x_2 \dots (1 - x_{M-1}) \left(1 + g(\mathbf{x_M})\right) \\ \vdots \\ f_{M-1}(\mathbf{x}) = \frac{1}{2} x_1 (1 - x_2) \left(1 + g(\mathbf{x_M})\right) \\ f_M(\mathbf{x}) = \frac{1}{2} (1 - x_1) \left(1 + g(\mathbf{x_M})\right) \\ \text{subject to } 0 < x_i < 1, \text{for } i = 1, 2, \dots, n \end{cases} \tag{19}$$

The $g(\mathbf{x_M})$ functional requires $|\mathbf{x_M}| = k$ variables and can take any function as long as $g(\mathbf{x_M}) > 0$. In this thesis $g(\mathbf{x_M})$ is, following the suggestion of authors in [57], assumed as:

$$g(\mathbf{x_M}) = 100[|\mathbf{x_M}| + \sum_{x_i \epsilon\, \mathbf{x_M}} (x_i - 0.5)^2 - \cos\left(20\pi(x_i - 0.5)\right) \tag{20}$$

The total number of variables n depends on the number of objectives and the length of $\mathbf{x_M}$ and is equal to $n = \mathrm{M} + k - 1$. Value of parameter $k = 5$ is suggested. Pareto optimal front is achieved when $\mathbf{x_M} = 0$ and for a 3D case is shown in the Fig. 18.

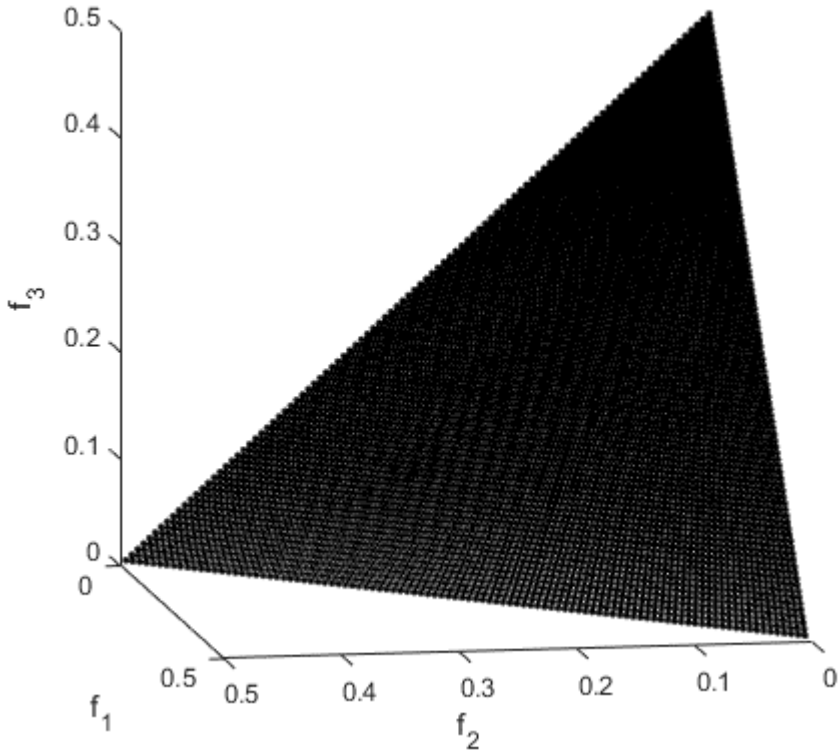


Fig. 18: True Pareto front of DTLZ1 (M=3)

- DTLZ2

Test problem DTLZ2 has a spherical Pareto-optimal front as shown in the Fig. 18

$$\begin{cases} f_1(\boldsymbol{x}) = \left(1 + g(\mathbf{x_M})\right) \cos\left(\frac{x_1 \pi}{2}\right) \dots \cos\left(\frac{x_{M-1}\pi}{2}\right) \\ f_2(\mathbf{x}) = \left(1 + g(\mathbf{x_M})\right) \cos\left(\frac{x_1 \pi}{2}\right) \dots \sin\left(\frac{x_{M-1}\pi}{2}\right) \\ \vdots \\ f_M(\mathbf{x}) = \left(1 + g(\mathbf{x_M})\right) \sin\left(\frac{x_{M-1}\pi}{2}\right) \\ \text{subject to } 0 < x_i < 1, \text{for } i = 1, 2, \dots, n \\ \text{where } g(\mathbf{x_M}) = \sum_{x_i \epsilon\, \mathbf{x_M}} (x_i - 0.5)^2 \end{cases} \tag{21}$$

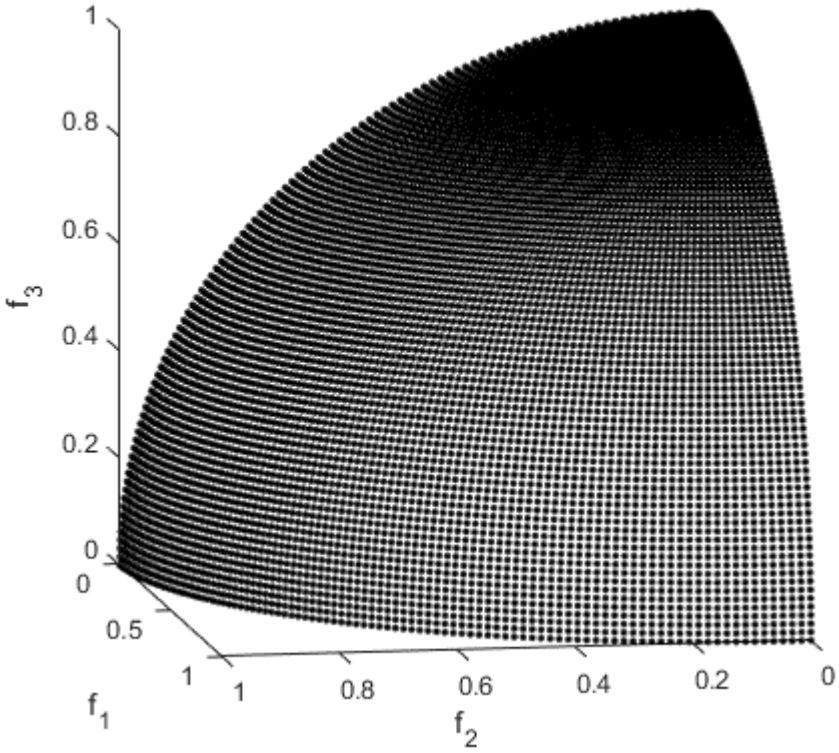


Fig. 19: True Pareto front of DTLZ2 (M=3)

It is suggested to use $k = 10$ in this case.

- DTLZ3

DTLZ3 is a more sophisticated problem, based on f_i functions from DTLZ2, equation (21) and $g(\mathbf{x_M})$ as in DTLZ1, equation (20). It was reported that algorithms found it harder to converge to true Pareto front in this case than in the previous ones due to many local Pareto-fronts which can be extensively explored during the course of optimization.

$$\begin{cases} f_1(\boldsymbol{x}) = \left(1 + g(\mathbf{x_M})\right) \cos\left(\frac{x_1\pi}{2}\right) \dots \cos\left(\frac{x_{M-1}\pi}{2}\right) \\ f_2(\mathbf{x}) = \left(1 + g(\mathbf{x_M})\right) \cos\left(\frac{x_1\pi}{2}\right) \dots \sin\left(\frac{x_{M-1}\pi}{2}\right) \\ \vdots \\ f_M(\mathbf{x}) = \left(1 + g(\mathbf{x_M})\right) \sin\left(\frac{x_{M-1}\pi}{2}\right) \\ \text{subject to } 0 < x_i < 1, \text{for } i = 1, 2, \dots, n \\ \text{where } g(\mathbf{x_M}) = 100[|\mathbf{x_M}| + \sum_{x_i \epsilon\, \mathbf{x_M}} (x_i - 0.5)^2 - \cos\left(20\pi(x_i - 0.5)\right) \end{cases} \tag{22}$$

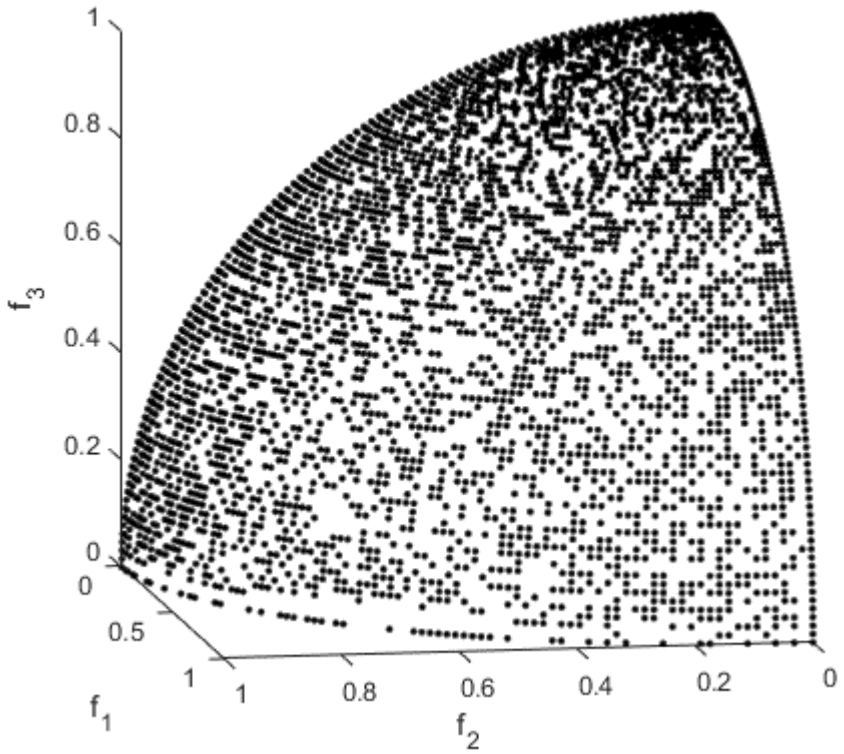


Fig. 20: True Pareto front of DTLZ3 (M=3).

- DTLZ4

DTLZ4 problem is based on DTLZ2 with an introduction of $x_i \rightarrow x_i^{\alpha}$, with the value $\alpha = 100$ suggested. The parameter α introduces an increased dependence on initial population when solving this problem.

$$\begin{cases} f_1(\boldsymbol{x}) = \left(1 + g(\mathbf{x_M})\right) \cos\left(\frac{x_1^{\alpha}\pi}{2}\right) \dots \cos\left(\frac{x_{M-1}^{\alpha}\pi}{2}\right) \\ f_2(\mathbf{x}) = \left(1 + g(\mathbf{x_M})\right) \cos\left(\frac{x_1^{\alpha}\pi}{2}\right) \dots \sin\left(\frac{x_{M-1}^{\alpha}\,\pi}{2}\right) \\ \vdots \\ f_M(\mathbf{x}) = \left(1 + g(\mathbf{x_M})\right) \sin\left(\frac{x_{M-1}^{\alpha}\pi}{2}\right) \\ \text{subject to } 0 < x_i < 1, \text{for } i = 1, 2, \dots, n \\ \text{where } g(\mathbf{x_M}) = \sum_{x_i \epsilon\, \mathbf{x_M}} (x_i - 0.5)^2 \end{cases} \tag{23}$$

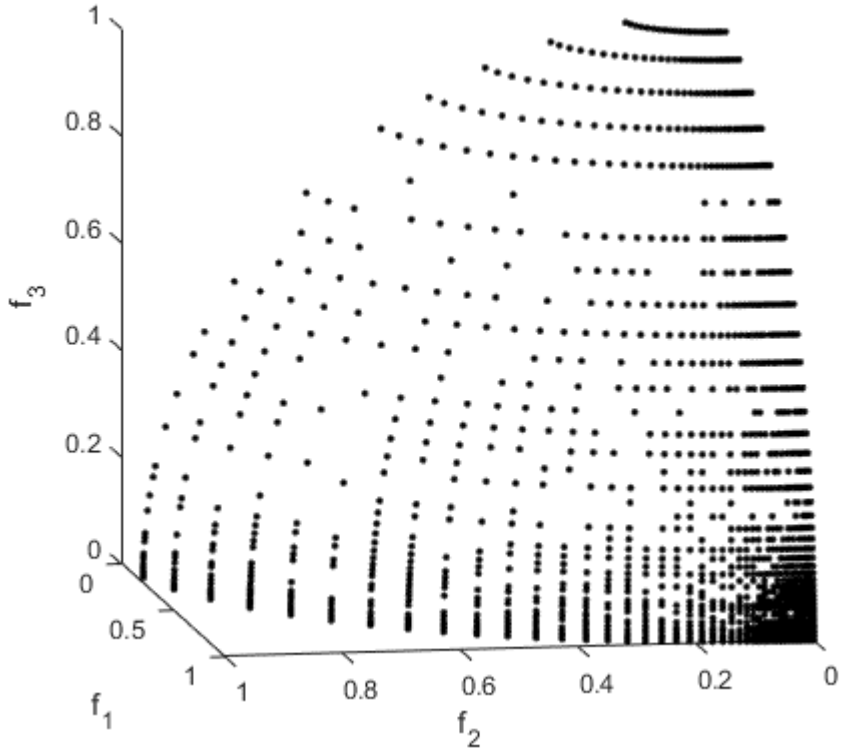


Fig. 21: True Pareto front of DTLZ4 (M=3).

Due to lack of other scalable test problems and several limitations of DTLZ such as none of the problems being deceptive or non-separable, a new set of test problems called *The WFG Toolkit* was proposed in [60]. WFG problems are built in a different manner than previously described problems. Values of WFG functions are obtained via a series of transitions. WFG operates on a vector of parameters **x** which is derived, through transformation functions, from a vector of working parameters (or design variables) **z** which is directly controlled by optimization algorithm. Every transition implements another feature such as multimodality or non-separability to a resulting vector. Except for transformation function a set of shape functions is also used to determine the geometry of fitness space. To create a test problem using the WFG Toolkit it is necessary to establish constraints, a set of transformation functions, including transformation parameters if applicable and a set of shape functions. The resulting problems are fully scalable in terms of number of objectives and design variables.

The general idea of a problem built using WFG Toolkit is as follows:

$$
\begin{aligned}
&\text{Given:}\\
&\qquad \mathbf{z} = \{z_1, \dots, z_k, z_{k+1}, \dots, z_n\}\\
&\text{Minimise:}\\
&\qquad f_{m=1:M}(\mathbf{x}) = x_M + S_m h_m(x_1, \dots, x_{M-1})\\
&\text{where:}\\
&\qquad \mathbf{x} = \{x_1, \dots x_M\}\\
&\qquad = \{\max(t_M^p, A_1)\,(t_1^p - 0.5)\\
&\qquad + 0.5, \dots, \max(t_M^p, A_{M-1})\,(t_{M-1}^p - 0.5)\\
&\qquad + 0.5, t_M^p\}\\
&t^p = \{t_1^p, \dots, t_M^p\} \leftarrow [\ \mathbf{t}^{p-1} \leftarrow [\ \dots \leftarrow [\ \mathbf{t}^1 \leftarrow [\ \mathbf{z}_{[0,1]}\\
&\mathbf{z}_{[0,1]} = \{z_{1,[0,1]}, \dots, z_{n,[0,1]}\} = \left\{\frac{z_1}{z_{1,max}}, \dots, \frac{z_n}{z_{n,max}}\right\}
\end{aligned}
\tag{24}
$$

where M is the number of objectives, $\mathbf{x}$ is a set of M underlying parameters, $\mathbf{z}$ is a set of $n \geq M$ working parameters (design variables in the optimization problem) including k position parameters and l distance parameters, A are degeneracy constants, S are scaling constants, h are shape functions, $\mathbf{t}$ are transition vectors (each one of them is a result of using a transformation function on a vector). The domain of $z_i \in [0, z_{i,max}]$ and the domain of $x_i \in [0,1]$.

In [60] authors suggest a set of transformation functions and shape functions, a selection of which will be described in detail.

Convex shape function:

$$
\begin{aligned}
&convex_1(x_1, \dots, x_{M-1}) = \prod_i^{M-1} \left(1 - \cos\left(\frac{x_i \pi}{2}\right)\right)\\
&convex_{m=2:M-1}(x_1, \dots, x_{M-1})\\
&= \left(\prod_i^{M-1} \left(1 - \cos\left(\frac{x_i \pi}{2}\right)\right)\right)\left(1 - \sin\left(\frac{x_i \pi}{2}\right)\right)\\
&convex_M(x_1, \dots, x_{M-1}) = \left(1 - \sin\left(\frac{x_i \pi}{2}\right)\right)
\end{aligned}
\tag{25}
$$

Concave shape function:

$$
\begin{aligned}
&concave_1(x_1, \dots, x_{M-1}) = \prod_i^{M-1} \sin\left(\frac{x_i \pi}{2}\right)\\
&concave_{m=2:M-1}(x_1, \dots, x_{M-1})\\
&= \left(\prod_i^{M-1} \sin\left(\frac{x_i \pi}{2}\right)\right) \cos\left(\frac{x_{M-m+1} \pi}{2}\right)\\
&concave_M(x_1, \dots, x_{M-1}) = \cos\left(\frac{x_i \pi}{2}\right)
\end{aligned}
\tag{26}
$$

Mixed shape function ($\alpha > 0;\ A \in \{1,2, \dots\}$).:

$$mixed_M(x_1, \dots, x_{M-1}) = \left(1 - x_1 - \frac{\cos\left(2A\pi x_1 + \frac{\pi}{2}\right)}{2A\pi}\right)^{\alpha} \tag{27}$$

In case of all shape functions $x_1, \dots, x_{M-1} \in [0,1]$ and A and α are constants.

b_falt (bias: flat region) transformation function ($A, B, C \in [0,1], B < C, B = 0 \Rightarrow A = 0 \wedge C \neq 1, , C = 1 \Rightarrow A = 1 \wedge B \neq 0$):

$$\begin{aligned} b_{flat(y,A,B,C)} = A + \min(0, \lfloor y - B \rfloor) \frac{A(B - y)}{B} \\ - \min(0, \lfloor C - y \rfloor) \frac{(1 - A)(y - C)}{1 - C} \end{aligned} \tag{28}$$

b_poly (bias: polynomial) transformation function ($\alpha > 0, \alpha \neq 1$):

$$b_{poly(y,\alpha)} = y^{\alpha} \tag{29}$$

s_linear (shift: linear) transformation function ($A \in (0,1)$):

$$s\_linear(y, A) = \frac{|y - A|}{|\lfloor A - y \rfloor + A|} \tag{30}$$

s_multi (shift: multi-modal) transformation function ($A \in \{1,2, \dots\}, B \geq 0, (4A + 2\pi) \geq 4B, C \in (0,1)$):

$$\begin{aligned} & s\_multi(y, A, B, C) = \\ & = \frac{1 + \cos\left[(4A + 2)\pi\left(0.5 - \frac{|y - C|}{2\lfloor C - y \rfloor + C}\right)\right] + 4B\left(\frac{|y - C|}{2(\lfloor C - y \rfloor + C)}\right)^2}{B + 2} \end{aligned} \tag{31}$$

r_sum (reduction: weighted sum) transformation function ($|\mathbf{w}| = |\mathbf{y}|, w, \dots, w_{|\mathbf{y}|} > 0$):

$$r\_sum(\mathbf{y}, \mathbf{w}) = \left(\sum_{i=1}^{|\mathbf{y}|} w_i y_i\right) / \sum_{i=1}^{|\mathbf{y}|} w_i \tag{32}$$

For all WFG functions the constants are set:

$$\begin{gathered} S_{m=1:M} = 2m \\ A_1 = 1 \\ A_{2:M-1} = \begin{cases} 0, \text{for WFG3} \\ 1, otherwise \end{cases} \end{gathered} \tag{33}$$

Working parameter domain is set be of a different magnitude for each parameter:

$$z_{i=1:n,max} = 2i \tag{34}$$

Based on aforementioned shape and transformation function test problems can be build. Authors suggest a series of 9 test functions. Most of these functions have known true Pareto fronts although it was proven that WFG3's true Pareto front is still unknown [61].

- WFG1

WFG1 is a separable, unimodal problem with polynomial and flat bias and both convex and mixed geometry of the Pareto front. It is constructed using the following set of transformations, shape functions and parameters:

Shape	$h_{m=1:M-1} = convex_m$ $h_M = mixed_M$ (with $\alpha = 1$ and $A = 5$)
t¹	$t^1_{i=1:k} = y_i$ $t^1_{i=k+1:n} = s\_linear(y_i, 0.35)$
t²	$t^2_{i=1:k} = y_i$ $t^2_{i=k+1:n} = b\_flat(y_i, 0.8, 0{,}75, 0.85)$
t³	$t^3_{i=1:n} = b\_poly(y_i, 0.02)$
t⁴	$t^4_{i=1:M-1} = r\_sum\left(\left\{y_{\frac{(i-1)k}{M-1}+1}, \dots, y_{\frac{ik}{M-1}}\right\}, \left\{\frac{(i-1)k}{M-1} + 1, \frac{ik}{M-1}\right\}\right)$ $t^2_{i=k+1:n} = b\_flat(y_i, 0.8, 0{,}75, 0.85)$

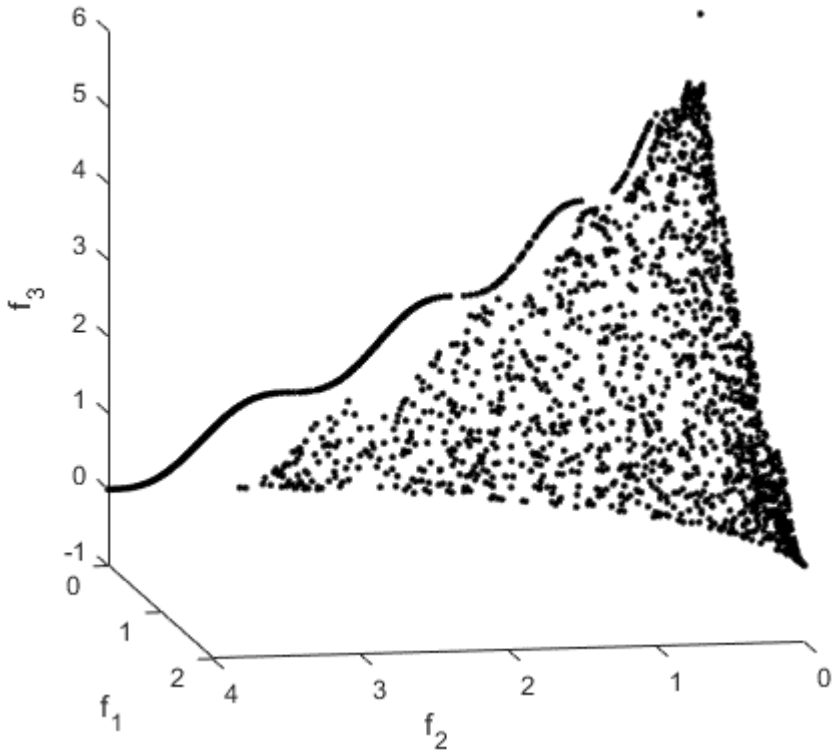


Fig. 22: True Pareto front of WFG1 (M=3)

- WFG4

WFG4 is a separable, multimodal problem with no bias and a concave geometry of Pareto front. It is constructed using the following set of transformations, shape functions and parameters. It is constructed using the following set of transformations, shape functions and parameters:

Shape	$h_{m=1:M} = concave$
t¹	$t^1_{i=1:M} = s\_multi(y_i, 30, 10, 0.35)$
t²	$t^2_{i=1:M-1} = r\_sum\left(\left\{y_{\frac{(i-1)k}{M-1}+1}, \dots, y_{\frac{ik}{M-1}}\right\}, \{1, \dots, 1\}\right)$ $t^2_M = r\_sum(\{y_{k+1}, \dots, y_n\}, \{1, \dots, 1\})$

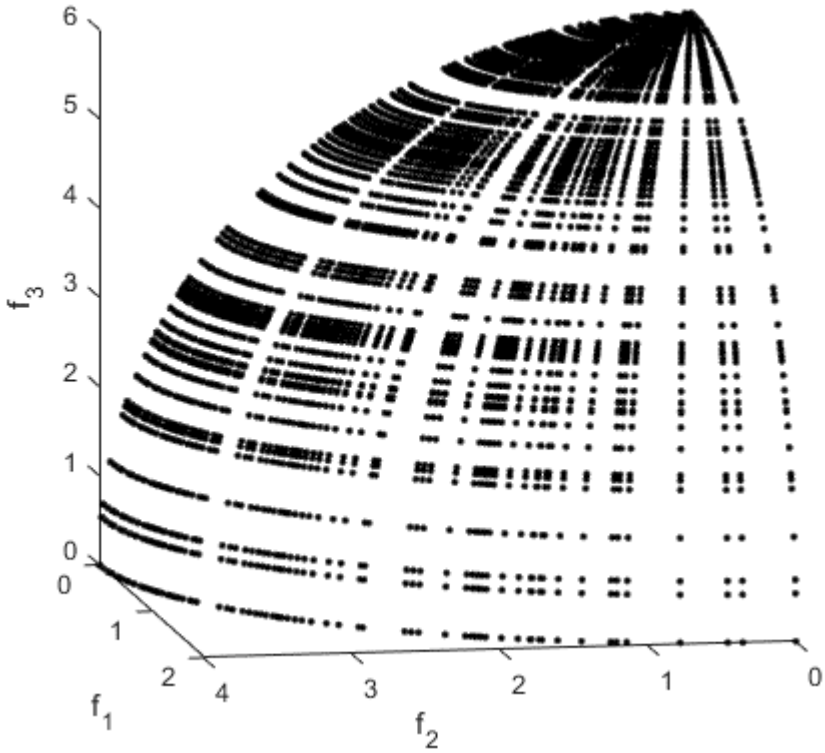


Fig. 23: True Pareto front of WFG4 (M=3).

Test problems belonging to a WFG toolkit exceed functionality of previous problems in a way they allow a combination of features desired of test functions and the choice on which features should be exhibited in a constructed test function entirely depend on problem designer's choice. This toolkit can be understood as a set of features from which certain characteristics can be drawn to define a scalable problem according to specific needs.

2.7. **Performance metrics for multiobjective optimization**

In order to assess performance of optimization algorithms, the resulting Pareto fronts have to be compared. In case of a single-objective optimization problem, this would be a simple task reduced to comparing the values of the considered function achieved after the optimization

process. For the multiobjective problems, the results after the optimization come in form of a set of non-dominated (Pareto-optimal) solutions, which stems the necessity to employ other comparison methods. For multiobjective problem the definition of quality of obtained set of solution is much more complex than in case of a single objective and involves features such as [62]:

- Distance between obtained set of solution and a true Pareto front,
- Good distribution of obtained solutions, usually uniform is desired,
- Wide extend of values should be covered by obtained solutions.

In order to assess said features in obtained solutions multiple approaches are used either as standalone performance metrics or as combination of metrics:

- Generational distance (GD)

Generational distance provides information on the distance between obtained Pareto front and true Pareto front. It can be considered as an error measure [63].

$$\mathrm{GD} = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n} \tag{35}$$

where $d_i = \min \left\| f(x_i) - \mathrm{PF}_{\mathrm{true}}(x_j) \right\|$ is a distance in the solution space between solution x_i and nearest solution on the true Pareto front. n is the number of solutions in the analysed set. Lower value of GD indicates better performance.

- Inverted generational distance (IGD)

Inverted generational distance provides information on both convergence and diversity.

$$\mathrm{IGD} = \frac{\sum_{v \in \mathrm{PF}_{\mathrm{true}}} d(v, X)}{|\mathrm{PF}_{\mathrm{true}}|} \tag{36}$$

where X is a set of non-dominated solutions and $d(v, X)$ is a minimum Euclidean distance between v and points in X. Lower value of IGD indicates better performance.

- Pure diversity (PD)

Pure diversity metric was introduced in [64] and is determined using a clustering algorithm based on distance between solutions. Higher value of PD indicates a better performance.

- Spacing (S)

Spacing metric provides information on whether found solutions are evenly distributed over the approximation front. $\bar{d}$ is mean value of all d_i and n is the number of solutions.

$$\mathrm{S} = \frac{1}{n} \sum_{i=1}^{i} \left(d_i - \bar{d}\right)^2 \tag{37}$$

where d_i is the Euclidean distance in solution space between solution x_i and closest solution on the true Pareto front.

Lower value of spacing indicates better performance.

- Spread (Δ)

Spread is a metric which gives information on how well found solutions are distributed over the solution space and to what extend is the true Pareto front covered [65].

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{N-1} |d_i - \bar{d}|}{d_f + d_l + (N-1)\bar{d}} \tag{38}$$

where N is the number of solutions. d_i is the Euclidean distance between neighbouring solutions in the obtained set of solutions, $\bar{d}$ is mean value of all d_i. d_f and d_l are Euclidean distance

between extreme solutions and the boundary solutions in the obtained set. Lower value of spread indicates better performance.

- Hypervolume (HV)

Hypervolume is a metric which considers all features of a good quality Pareto front: closeness to true Pareto front, diversity of obtained solutions and their extent. This feature and the fact that it does not require any knowledge on true Pareto front makes it a convenient indicator of quality of solutions, which at the same time is considered a comprehensive one. HV requires defining a reference point W for which the value of hypervolume is determined. Hypervolume indicator can be considered as a union of hypercubes constructed using a section between a reference point W and solution X_i as a diagonal for all the solutions.

$$\mathrm{HV} = \bigcup_{i=1}^{|\Omega|} v_i \tag{39}$$

An important downside of HV is its computational complexity, especially in case of high-dimensional solution space, calculating HV can be time-consuming. Computational complexity of HV is exponential with the number of objectives [66]–[69]. Higher value of HV indicates better performance. Graphic interpretation of HV in case of two-dimensional solution space is shown in Fig. 24.

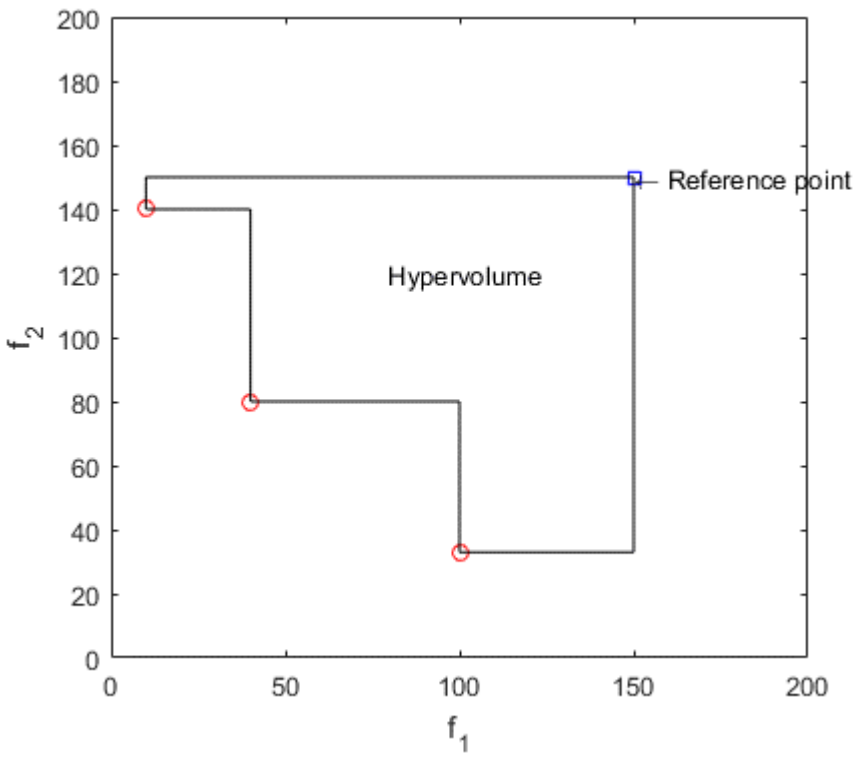


Fig. 24: HV in case of two objectives, red circles are non-dominated solutions.

3. DIFFERENTIAL EVOLUTION – GAME THEORY ALGORITHM (DEGT)

One of the main goals of this dissertation is development of a novel multiobjective algorithm to deal with optimization of mechanical problems concerning a large number of objectives within the frames of soft computing. Key idea of the algorithm is to use differential evolution as a single-objective optimizer combined with elements of game theory to form a new multiobjective algorithm DEGT. Author's idea was implemented in C++ programming language, comprehensively tested using mathematical test functions as well as compared with well-established multiobjective optimization algorithms. The algorithm was then used to optimize mechanical systems both analytical and numerical including multiscale and multiphysics simulations.

Many researchers have been investigating the idea of utilising game theory elements in multiobjective optimization. Ameljanczyk used vector optimization methods to solve decision models including game problems in [70]. In [71] a game theoretic approach to optimization problems using multiple genetic algorithms to solve a multiobjective problems was proven to be computationally beneficial. Multiobjective discrete optimization of laminates using multi-membered evolution strategy and a cooperative game theory approach was discussed in [72]. Ideas of using genetic algorithms enhanced by game theory elements in multiobjective optimization and their various applications in engineering problems were presented by Periaux *et al.* in [73]–[75]. In [76] Nash strategies were used coupled with evolutionary algorithms so solve multiobjective optimization problems and it was proven that looking for a Nash strategy as a preceding step before applying evolutionary optimization algorithm can considerably improve the time needed to find the Pareto optimal front. An approach utilising game theory elements and fuzzy logic to solve multiobjective optimization design problem was shown in [77]. Coevolutionary approach to optimization based on game theory was used to introduce a novel multiobjective algorithm in [78]. In [79] three game models: Nash equilibrium game model, coalition cooperative game model and evolutionary game model are examined and compared with each other proving evolutionary game model is the most effective in terms of both computational efficiency and precision. In [80], [81] game theory was used to design a multiobjective optimization algorithm coupled with artificial immune systems. A review of multiple biologically-inspired methods and game theory in multiobjective optimization was presented in [82]. The example of an algorithm utilising elements of game theory to solve a multiobjective problem with two objectives and four design variables is shown in the Fig. 25.

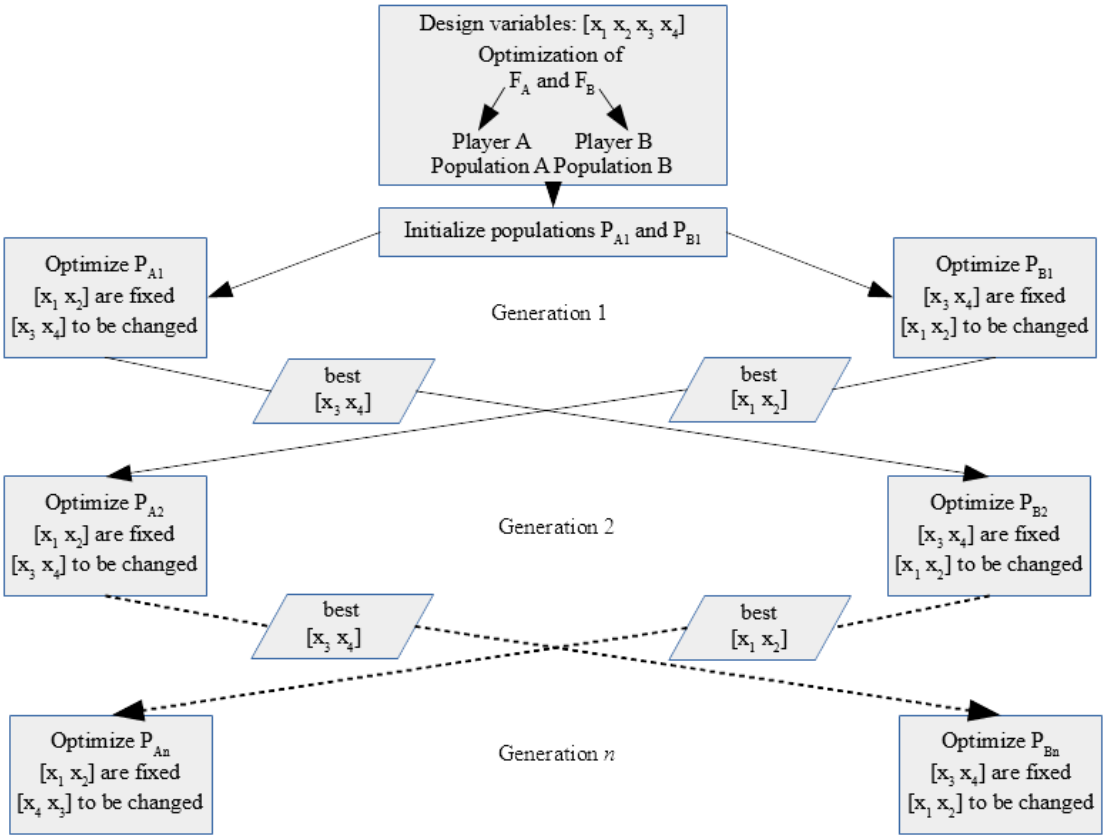


Fig. 25: Example of using game theory in optimization of a two-objective problem

3.1. **Game theory**

It is likely not possible to indicate a place where or a moment when games were first present in the human history. Games have always existed alongside humankind as people made strategic decisions to achieve specific goals. Games have been played throughout history for a variety of reasons starting with petty motivations such as entertainment or even greed, ending on very noble causes such us improvement in medicine, engineering, and other branches of science. The decisions made during the games can take many shapes and forms. Games might be played by a single player or multiple players and if there are multiple players involved in the game, their decisions might either affect and be affected by other players decisions, or on the contrary – decisions might be independent and not influence or be influenced by other players. The player might want to make his decisions based on a single objective or a group of objectives to be fulfilled. Games might be very simple, for example a toss of coin to determine the winner can be considered one of the simplest possible games, but thanks to its simplicity, ease of access and little time required to be played, it was coined as one of people's favourite, universal way to help resolve situations at random. But games can also have very sophisticated rules, moreover situation concerned in a game might have clear, static rules or be dynamic and change over time. It is also possible to imagine a situation in which some rules of the games are unknown to players and even despite this, they might want to play, take decisions, and observe the outcome. Many games are believed to best be taught when you play them.

With the diversity of games scenarios arose the need to capture these ideas in a formal way. Therefore, mathematical, algorithmic, and economic tools were developed to deal with decision making processes – to model, analyse and solve game problems.

Back in 16th century an Italian mathematician, philosopher, and physician Gerolamo Cardano wrote The Book on Games of Chance (orig. *Liber de ludo aleae*). His passion for gambling resulted in a book concerning probability and mathematical laws governing the outcome of seemingly random events. The book was only published posthumously in 1663 and didn't have a significant impact on scholars investigating probability. A French writer and another gambler, Chevalier de Méré, was a friend of a well-known French inventor and physicist Blaise Pascal and presented him with a certain early game theoretic problem. This problem was discussed in a series of letters exchanged between Pascal and another French mathematician Pierre de Fernat. A modern writer, Keith Devlin, describes the impact of these letters in these words: "*The Pascal-Fermat correspondence showed that it is possible to use mathematics to see into the future*" in [83]. Inspired by these letters Pascal wrote Treatise on the Arithmetical Triangle (orig. *Traité du triangle arithmétique*), written in 1654 and published in 1665 in which he describes what we today know as famous Pascal's Triangle. This work is considered to be one of the earliest papers on probability and has introduced a new, later broadly used term: *expected value*. In 1657, a Dutch mathematician Christiaan Huygens, influenced by Pascal and Fermat, published *Exercitationum Mathematicarum* where he investigates problem of points and brings an early foundation of a game-theoretic approach to a problem of probability. Another thinkers, who were successors of Huygen's works on probability were: Pierre Remond de Monmort (1678–1719), Abraham De Moivre (1667–1754) and Jacob Bernoulli (1655-1705) but they disregarded Huygen's and Pascal's concept of a game theoretic approach, focused on reasoning based on the structure of the game and instead proceeded to investigate probability problems based on frequency, which was a well-known and universally accepted approach among mathematicians at the time [84]. In 1713, a British diplomat James Waldegrave in his letters proposed a mixed min-max a solution to a card game *Le Her*, later known as the Waldergrave problem in probability and game theory. In 1838, a French mathematician Antoine Augustin Cournot published Researches into the Mathematical Principles of the Theory of Wealth (orig. *Recherches sur les principes mathématiques de la théorie des richesses*) where he investigates a problem of duopoly and proposes solution which is a Nash equilibrium of the game. Optimal chess strategy was proven to be strictly determined in 1913 by a German mathematician Ernst Zermelo in his book On an Application of Set Theory to the Theory of the Game of Chess (orig. *Über eine Anwendung der Mengenlehre auf die Theorie des Schachspiels*).

These previous works sprang novel concepts and gave a significant background to a new field of science, which was soon to be created, but it was only until 1928 and the paper On the Theory of Games of Strategy [85] (orig. *Sur la théorie des jeux* or *Zur Theorie der Gesellschaftsspiele*) when game theory emerged as a modern, unique field of mathematics thanks to the work of a Hungarian-American mathematician John von Neumann. The main innovation in this paper was the proof of minimax theorem for some variants of two-players zero-sum games [86] which was later generalised and proven in an alternative way by several researchers [87], [88]. In 1944 Neuman, together with a German economist Oskar Morgenstern, wrote Theory of Games and Economic Behavior [89] which was a breakthrough work in understanding game theory as a multidisciplinary research field. In the book, cooperative games of multiple players are considered. A significant observation was made, that in case of games with 3 players, the nature of their conflict is no longer strictly competitive, as players can form coalitions to draw benefits. The book also introduced two terms: *imputation* and *domination*, the latter of which, understood as an abstract relation on a set of points is particularly important

in the field of multiobjective optimization. Later studies on the history of game theory [90] notice that very little attention was given to science concerning conflicts of interest between 1928 and 1944 and attribute this to the fact, that first publication was directed towards mathematicians and the book of 1944 was successful in attracting attention of researchers in both mathematical and non-mathematical journals. Since then, game theory was a field of extensive research of many scientists in various branches of science, including applied mathematics, economics, biology, political science, computer science, military, and philosophy. An important person in the development of game theory was an American mathematician John Forbes Nash Jr. whose 28-page PhD thesis (1950) on non-cooperative games earned him the Nobel Memorial Prize in Economic Sciences in 1994. Nash also authored a series of articles [91]–[94] on the essential concept of non-cooperative equilibrium, later named Nash equilibrium after him. 12 game theorists have been awarded Nobel prize, the latest ones being awarded in the field of economy for Roger B. Myerson, Leonid Hurwicz and Eric S. Maskin for their work on mechanism design theory in 2007.

There are multiple definitions of game theory, although most of them try to encapsulate the process of decision making. In [95] three elements of a game in a strategic form are distinguished: finite set of players, strategy and payoff functions. Each player tries to increase his payoff function, which might involve helping or hurting other players, called opponents, whether or not their best interests are shared. The definition of strategy varies in different fields of applicability, for example in case of economy, strategy might be understood as a set of prices and in political science as votes. In these cases, strategy is a set of action based on deep analysis and thoughtful reasoning, but that's not always a case. Game theory has successfully explained behaviour of animals such as spiders or fish, whose decisions aren't based on thinking at all. The reason for this success is the fact that these animals' genes programmed them to behave rationally, because otherwise they would be extinct, as nature eliminates unfit units in the evolutionary process.

Another definition of game theory presented in [96] emphasizes interaction between decision makers, who are rational and reason strategically. The game theory regards abstract models which can describe real-life situations. The line between applied and theoretical aspects of game theory is vague and if these two sides were to be considered players, they would play a cooperative game, as real problems are solved thanks to theoretical advancements, and these are motivated and pushed forward by real problems which arise over time.

Games models can be established in a variety of ways, one of the most basic divisions is related to the aspect of cooperation between players: non-cooperative games versus cooperative games. In case of non-cooperative games, players are always treated as single entities in a game, meanwhile in case of games with cooperation, a group of players can be regarded as a single entity. Another way of defining whether a game is cooperative or not is the permission to hold communication between players before the game. In cooperative games, players are allowed to discuss tactics, form alliances, and make binding agreements. In non-cooperative games these actions are strictly prohibited. [90]

Zero-sum games are a specific case of games where a benefit gained by a player always equals a loss gained by a player or players. In other words, a zero-sum game is one in which an aggregate gains and losses must be zero. Many well-known games, such as Poker, Chess or Go are zero-sum games. On the contrary, non-zero-sum games allow a possibility of a net gain

or loss different than zero. This situation is also often prevailing in real decision making scenarios, for example regarding stock market, when gain of one investor is not necessary linked to loss of another. A well-known paradox known as prisoner's dilemma (Fig. 26) is an example of a non-zero sum game. It was proven in [89] that any non-zero sum game can be transformed to a zero-sum game by introducing an additional player representing the global gain or loss.

	Prisoner A stays silent	Prisoner A collaborates
Prisoner B stays silent	Both: 6 months sentence	Prisoner A: goes free Prisoner B: 10 years jail
Prisoner B collaborates	Prisoner A: 10 years jail Prisoner B: goes free	Both: 5 years jail

Fig. 26: Prisoner's dilemma is a non-zero-sum game.

Games can also be divided on the basis of order of actions on simultaneous (static) and sequential games. In simultaneous games players decide on their actions without prior knowledge on the choices of their opponents. In contrast, sequential games are played according to a schedule and each player can adjust his moves in relation to opponent's action. An example of a sequential game is chess and rock-paper-scissors is a simultaneous game.

Another significant division can be made on games with perfect and imperfect information about all aspects of the game, including: players, strategies, payoffs and previous decisions. Tic-tac-toe, checkers and go are examples of games with perfect information unlike poker and bridge, which are games with imperfect information.

An important, basic concept in the field of game theory, providing a widely used solution for strategic games is Nash's equilibrium. It is applicable to games provided that the game is strategic (or otherwise in normal form) and therefore:

- consists of a finite set N of players,
- a nonempty set of actions A_i for each of the players $i \in N$ is defined and
- a set of preference relations $\succsim_i$ on $A = x_{j \in N} A_j$ for each of the players $i \in N$ is defined.

If a set of actions is finite, then a game can also be considered finite. The set of outcomes $x_{j \in N} A_j$ is defined over A rather than A_i deliberately because each player takes into account not only his actions but also actions of other players.

These limitations constitute a model of a high level of abstraction which is applicable to a broad range of real problems. Players may be single person, a group of people, an abstract entity, organization, animal etc. and sets of action can be either very narrow and consist of only a few alternatives, or can they be a sophisticated plans that cover a variety of aspects of a specific decision. These limitations are relatively easy to fulfil, the more challenging one is related to the definition of a set of preferences, which might cover for example how an individual feels about a certain outcome caused by his actions or if the entity does not act consciously then for example their rate of survival or reproductive success. An informal definition of Nash's equilibrium is a situation in which no player can further improve his payoff by taking any possible action unless other players change their action. A more formal definition of a Nash's

equilibrium of a strategic game $\langle N, (A_i), \succsim_i \rangle$ is a profile of actions $a^* \in$ A that for every player $i \in$ N we have:

$$(a^*_{-i}, a^*_i) \succsim_i (a^*_{-i}, a^*_i) \text{ for all } a_i \in \text{A} \tag{40}$$

The concept of Nash equilibrium assumes that it is a steady state in which each player acts rationally and his expectations towards opponents' actions are correct. Nash's equilibrium may not exist for all games and if it exists, it might not necessarily be a Pareto-optimal solution. It is worth highlighting that the early concept of Nash equilibrium does not examine the process of reaching the state of equilibrium, which might be achieved in various ways further investigated by many researchers up to this day.

3.2. **Differential evolution**

There are numerous approaches to solving optimization problems. Deterministic and non-deterministic (heuristic) methods can be distinguished. Deterministic methods include gradient methods (e.g., steepest descent, conjugate gradient, Quasi-Newton) and non-gradient methods (e.g., Powell, Nelder-Mead). Gradient methods are very effective, but the need to calculate optimization functionals gradients is a significant drawback, limiting the application of this method in many practical mechanical problems. Another deficiency of deterministic methods (including non-gradient methods) is their tendency to getting stuck in local extremes, which is often the case in real engineering optimization problems. Heuristic optimization techniques, including bio-inspired, population-based methods came as an answer to the aforementioned limitations and found a wide range of applications in the problems of optimal design of mechanical systems [97].

One of such methods is Differential Evolution (DE), a single objective optimizer, which was introduced by Price and Storn in 1997 [98] and is acclaimed for its simple structure, ease of use, speed and robustness. Simple structure, which is a significant advantage for many researches as it considerably eases implementation, does not come at the expense of performance, which was proven to be superior to many much more complex algorithms [99], [100]. Low number of parameters, whose impact on the performance of the algorithm is well-studied is a significant advantage of the algorithm. DE has attracted attention of researchers improving and modifying the algorithm, some of the variants will be discussed in the proceeding parts of this chapter. A considerable study effort has been devoted to application of DE to a range of engineering and scientific problems. DE algorithm modified to automatically performed local restart of the population was used to solve space trajectory optimization problem [101], [102]. A parallel, surrogate based DE algorithm was used to solve coupled economic and emission hydrothermal optimization problem in [103]. Voltage stability driven load shedding optimization was investigated in [104]. Rural area micro grid energy optimization using DE was presented in [105]. Design optimization of microelectromechanical (MEMS) systems on the example of a comb-driven micro resonator was examined in [106]. Vehicle routing problem was solved using DE in [107]. Apart from applications of DE, some attention has been given to the theoretical foundations and analysis of the structure of the algorithm [108], [109].

In the original article [98] authors propose a series of variants of an algorithm, denoted as DE/X/Y/Z where X denotes vector to be mutated, it can be either random or best (highest fitness individual from population), Y is the number of difference vectors used, typically 1, but can be more and Z is a crossover scheme, such as binominal crossover. The most popular, basic variant of the DE is DE/rand/1/bin and unless it is explicitly stated, this variant is considered. Another variant which was early discovered to be especially successful was DE/best/2/bin.

The DE algorithm utilises selection and mutation as exploration and exploitation mechanisms. Initial population Q_0 is a set of parameter vectors x_j usually chosen randomly from the search space. During the course of optimization, a weighted difference between two randomly chosen population members x_r and x_s is calculated and then added, considering crossover rate, to a third one x_t to create a new design variables vector. If the resulting vector yields an improved objective value, it replaces the former one. DE is a self-adaptive method with low number of parameters, which is a significant advantage in comparison with many bio-inspired methods. The algorithm features an inherent elitism mechanism on the individual level, as new offspring solutions can never replace superior parent solutions. Only three parameters govern the performance of DE algorithm: crossover rate $CR \in [0,1]$, scale factor $F \in [0,2]$ and population size $DE_{pops}>3$. Low crossover rates result in preservation of a bigger part of the parent solution. Scale factor is used to calculate a weighted difference when generating individuals and can be considered a mutation parameter. Another parameter might be added as a termination condition, such as maximum number of iterations DE_{iter}. In the original paper [98] authors suggest that for most cases of optimization the value of F should be set between 0.4 and 1 and the value of CR between 0.1 and 0.9. Parameters used for the purpose of this research are population size $DE_{pops} = 6$, crossover rate $CR = 0.5$ and scale factor F = 0.7. Brick wall penalty mechanism is used to handle boundary constraint violation. DE algorithm pseudocode is shown in Fig. 27 and flowchart is shown in Fig. 28.

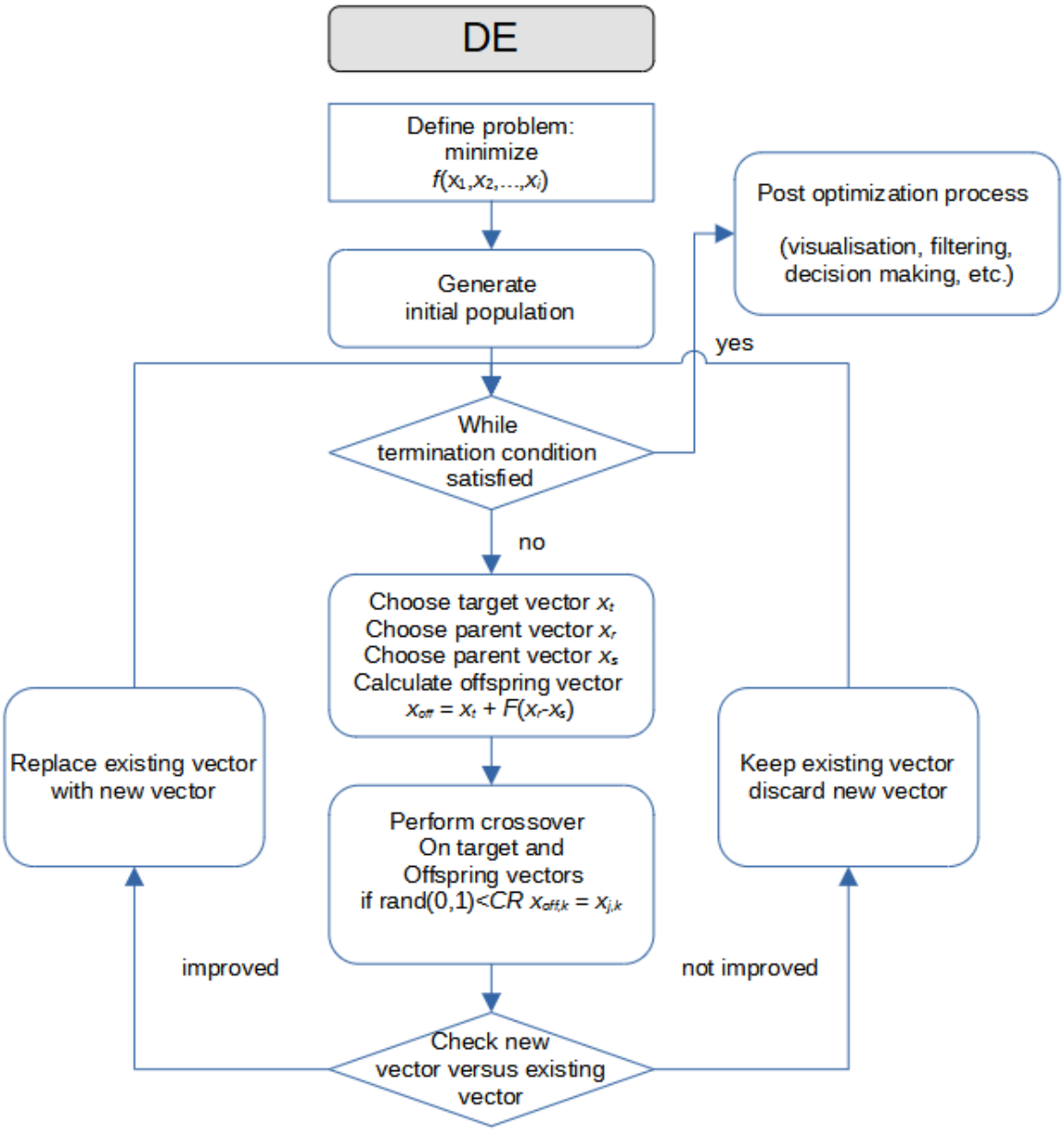


Fig. 27: Flowchart of DE algorithm in a basic variant

```
i ← 0
Pseudo-randomly generate initial population Qi of size NP=DEpops;
for j = 1:NP
        evaluate objective functions f(xj) for Qi;
end-for
while termination condition
        for j = 1:NP
                //mutation
                select three individuals xr, xs and xt from Qi;
                generate individual xoff = xt + F(xr-xs);
                //crossover
                for k=1:n
                        if rand(0,1)<CR
                                xoff,k = xj,k
                        end-if
                end-for
                //selection
                if f(xoff)≤ f(xj) //minimization problem
                        add individual xoff to population Qi+1
                else
                        add individual xj to population Qi+1
                end-if
        end-for
        i ← i+1
end-while
```

Fig. 28: Differential evolution pseudo-code.

Primal proposal of DE has been modified by many researchers and efficient variants are still often used as a single-objective optimizer in different engineering disciplines. Modifications of DE algorithm, such as MLSHADE-SPA [17] and SHADEILS [18] were proven to be the best performing optimizers in a WCCI'2018 Large-Scale Global Optimization Competition and are both considered state-of-the-art algorithms in this field. In [110] authors notice that constraint handling mechanism has an essential impact on the performance of DE and propose a strategy called *Inverse Parabolic Spread* (*IPS*) which is capable of dealing with non-linear constraints and by simple variable bounds. Implementing a self-adapting control parameters in differential evolution to form a new variant of DE algorithm was proven to display improved performance over benchmark problems in [111]. Another approach to provide

a way for self-adapting DE parameters was shown in self-adapting DE (SADE) algorithm in which an idea of self-adaptation involves learning from experience of generating promising solutions. SADE was comprehensively tested using 26 constrained test functions and compared to conventional DE variants and state-of-the-art adaptive DE variants [112]. Differential evolution has been used to form multiobjective evolution algorithms (MOEA/D, MOEA/D-DE), these algorithms were proven to perform similarly or outperform NSGA-II on chosen continuous optimization problems and knapsack problem [113], [114].

3.3. **General idea of a game theoretic approach to multiobjective optimization**

There are multiple analogies in the fields of game theory and multiobjective optimization, some of them are shown in the Tab. 2.

Tab. 2: Analogies between optimization and game theory

Game theory	Optimization
cooperative game	multiobjective optimization
players	objectives
resources	design variables
looking for a Nash equilibrium	looking for a set of Pareto optimal solutions

As a consequence of analogies and similarities between these fields numerous algorithms of single-objective optimization can be used with game theory elements to form multiobjective optimization tools in a various ways as presented in the previous section. In this dissertation the game theory elements were complimented with Differential Evolution algorithm to form a novel multiobjective optimization algorithm called DEGT (Differential Evolution – Game Theory). The choice of Differential Evolution as a single objective optimizer is motivated by its simple structure, ease of use, low number of working parameters and self-adaptive nature. This selection of desired features is not present in a wide range of other single-objective optimizers.

The idea behind coupling differential evolution and elements of game theory comes down to treating objectives as players, playing a cooperative game, trying to improve their respective objectives with the resources given and sharing the information with each other, iteratively looking for a Nash equilibrium.

Each player is given a part of design variable vector at random as their resources, while the rest of the vector is fixed and determined by other players' choices.

To assure diversification of solutions the assignment of resources is changed in a way each design variable is modified by one and only one player. Assignment of resources is changed after each of the players made his move.

The player's move is to run the single-objective optimization process, using differential evolution algorithm to improve one objective at the time using the resources allocated.

Father of game theory, John Neumann, in his first book, considered to be the beginning of modern game theory [85], said:

"*The fate of each player depends not only on his actions but also on those of the others, and their behavior is motivated by the same selfish interests as the behavior of the first player. We feel that the situation is inherently circular.*"

This very cycle mentioned in this quote prevails in the idea of multiobjective optimization using game theory elements, as every player proceeds to improve his objective function, but has to respect decisions of other players, which are also selfish and motivated by other objectives.

The flowchart of a general idea of DEGT is shown in Fig. 29. In the first step a problem needs to be defined including expression of objective functions and design parameters. Players in the game are linked to certain objectives at this point. First solution is sought concerning first objective and then the process is followed by playing consecutive cooperative games using DE

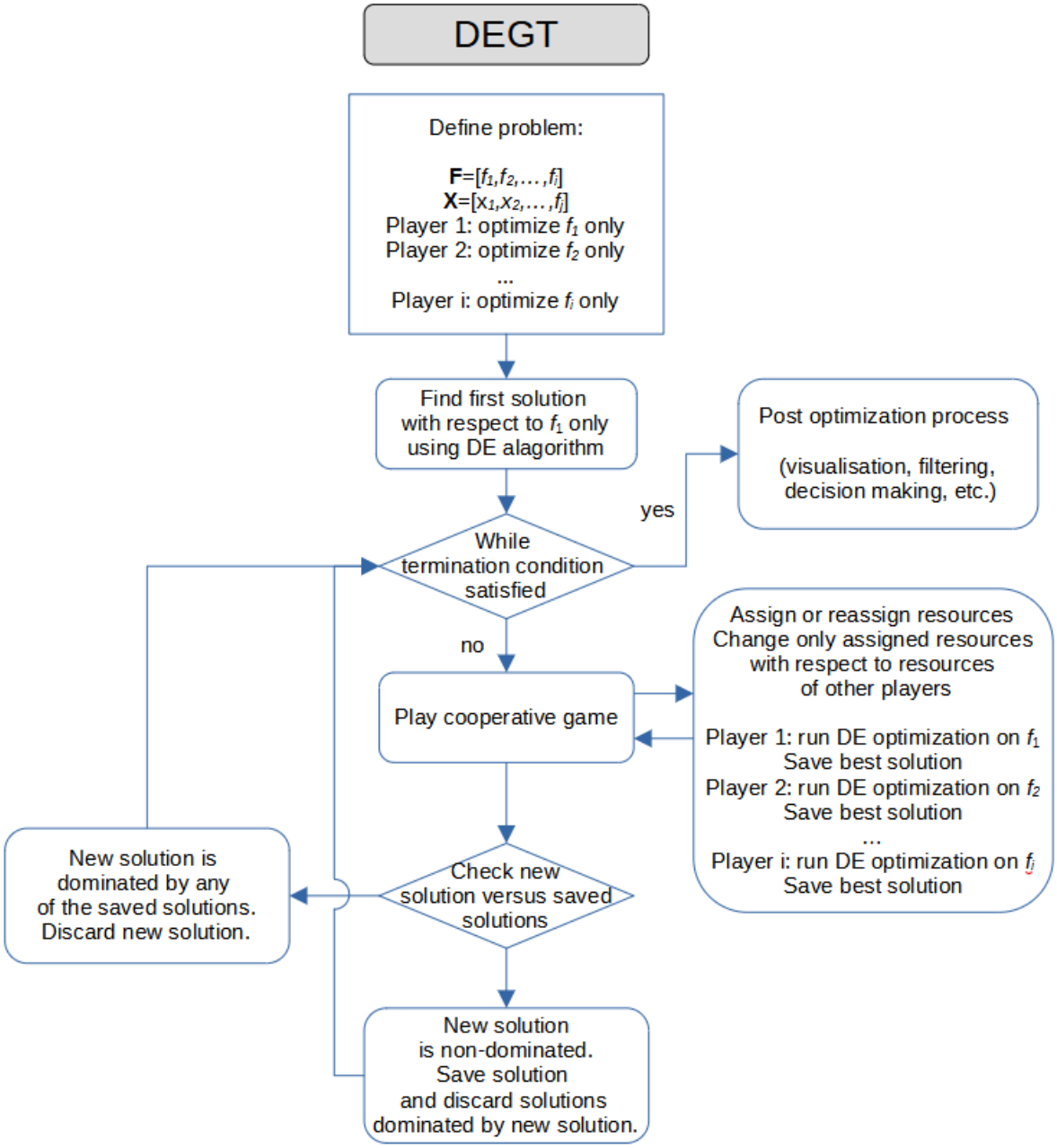


Fig. 29: Flowchart of DEGT

optimizer by each of the players. Optimization proceeds iteratively, new solutions are saved if they are non-dominated and discarded otherwise. After finishing satisfying termination condition the process is concluded with post-optimization tasks.

To further explain the ideas behind the DEGT algorithm a simple pseudocode is shown in the Fig 30.

```
i ← 1
Perform single objective DE optimization on objective 1;
Save design variable vector of best solution Si;
Calculate and save values of remaining objectives;
while termination condition
        if i% nobj=1
                assign design variables to objectives;
        end-if
        Set values of fixed design variables according to solution Si-1;
        Perform single objective DE optimization on objective 1+i% nobj;
        Save design variable vector of best solution Si;
        Calculate values of remaining objectives;
        if solution Si is non-dominated
                save solution Si;
                remove saved solutions dominated by Si;
        end-if
        i ← i+1
end-while
```

Fig. 30: DEGT pseudocode

3.4. **Simple example of a game theoretic approach**

In order to present the principles of game theoretic approach to multiobjective optimization a simple example can be considered (Fig. 31).

In case of mechanical problems optimization functions could be related to quantities such as: mass, volume, stresses, displacements, modal frequencies etc. and design variables could represent for example geometrical structure of an object: length, width, height, inner diameter, outer diameter; material properties: Young's modulus, Poisson's ratio or boundary conditions imposed on an analysed system: fixed or supported. In general design variables

describe the design of the system and objectives the describe system's answers related to particular needs identified beforehand.

For the purpose of an example it is assumed that the optimization problem was identified to consider three objectives represented by a vector of optimization functions $\mathbf{f} = [f_1, f_2, f_3]$ and the design of the problem is described by a vector of design variables $\mathbf{x} = [x_1, x_2, x_3, x_4, x_5]$. The optimization functions might be either minimized or maximized and it will not be distinguished in the example, the positive change in objective function will be referred to as "improvement".

In the first step three players playing a cooperative game are created, so that each one of them can manage exactly one objective. Before players make their decisions, they are assigned resources in the form of a part of design variable vector available for them to be updated. The remaining part of the vector of design variables is fixed and determined by other, players choices, based on their previous moves. If a design variable was not yet determined by any player before then it can take any feasible initial value. This situation happens only before the first move of each player, except for the player who makes his move last, as all other players have already made their decisions and thus determined the entire fixed part of the design variable vector.

In the next step, after resources are assigned, players, one by one, proceed to improve their respective objectives. In general, the assignment of objectives is performed randomly in such a way that each design variable is modified by exactly one player at the time. For the purpose of this example let's assume first assignment of objectives: player 1, who tries to improve objective f_1 can modify design variables x_2 and x_5; player 2, who tries to improve objective f_2 can modify design variable x_1 and player 3, who tries to improve objective f_3 can modify remaining design variables: x_3 and x_4. Players make their decisions in order, first player 1 solves a single objective optimization problem with regards to objective f_1. The value of design variables which are not included in the resources of player 1 take a fixed value during the course of optimization. Let's assume the initial feasible value for each objective can be 0. After having finished optimization, a new design with improved value of objective f_1 is obtained. Vector of design variables before player 1 makes his move: $\mathbf{x} = [0, x_2, 0, 0, x_5]$. After the single objective optimization process the design variables assigned to player 1 take a new value $\mathbf{x} = [0, x_{2N}, 0, 0, x_{5N}]$ this information is saved and shared with proceeding players. Values of remaining objectives are calculated and solution with a set of design variables is saved. Player 2 tries to find a design with improved value of objective f_2 in a similar manner, able to change only the design variable x_1 which was assigned to him. Vector of design variables before player 2 makes his move: $\mathbf{x} = [x_1, x_{2N}, 0, 0, x_{5N}]$. Notice that player 2 already uses the information shared with him by player 1 but still has no information on proposed values of x_3 and x_4. After the single objective optimization process the design variables assigned to player 1 take a new value $\mathbf{x} = [x_{1N}, x_{2N}, 0, 0, x_{5N}]$ this information is saved and shared with proceeding players. Values of remaining objectives are calculated and solution with a set of design variables is saved. Last player performs optimization process with respect to objective f_3 and is able to update values of design variables x_3 and x_4 having information on the remaining design variables from previous players. Vector of design variables before player 2 makes his move is $\mathbf{x} = [x_{1N}, x_{2N}, x_3, x_4, x_{5N}]$. After all, three players have made their moves a new design is obtained: $\mathbf{x} = [x_{1N}, x_{2N}, x_{3N}, x_{4N}, x_{5N}]$ and saved along with remaining values of objectives. All the design variables were able to be updated and all the objectives were able to

be improved. After all the players have made their moves, the assignment of resources (design variables) is rearranged at random with respect to the principle that each of the design variables has to be assigned to exactly one player.

In the next steps, after the assignment of resources was performed anew, the process is repeated in an iterative way until the stop condition is met and reassignment of resources is applied every time after the last player makes his move. After each move one of the objectives is improved or in a very specific scenario can remain unchanged but can never be worsened. Other objectives, not managed by a player making his move at the time, might take worsened values due to contradictory nature of objectives in multiobjective optimization problems. Whenever a new solution is to be saved, it is convening to compare it with solutions found beforehand. If the new solution is dominated by any of the solutions existing in the database, then it doesn't need to be saved and if the solution is non-dominated then any existing solutions dominated by a new solution can be removed from the database of solutions. This process can be performed either during the run of algorithm, or the set of solutions can be filtered off dominated solutions after the optimization process is finished as it is a relatively computationally effortless task.

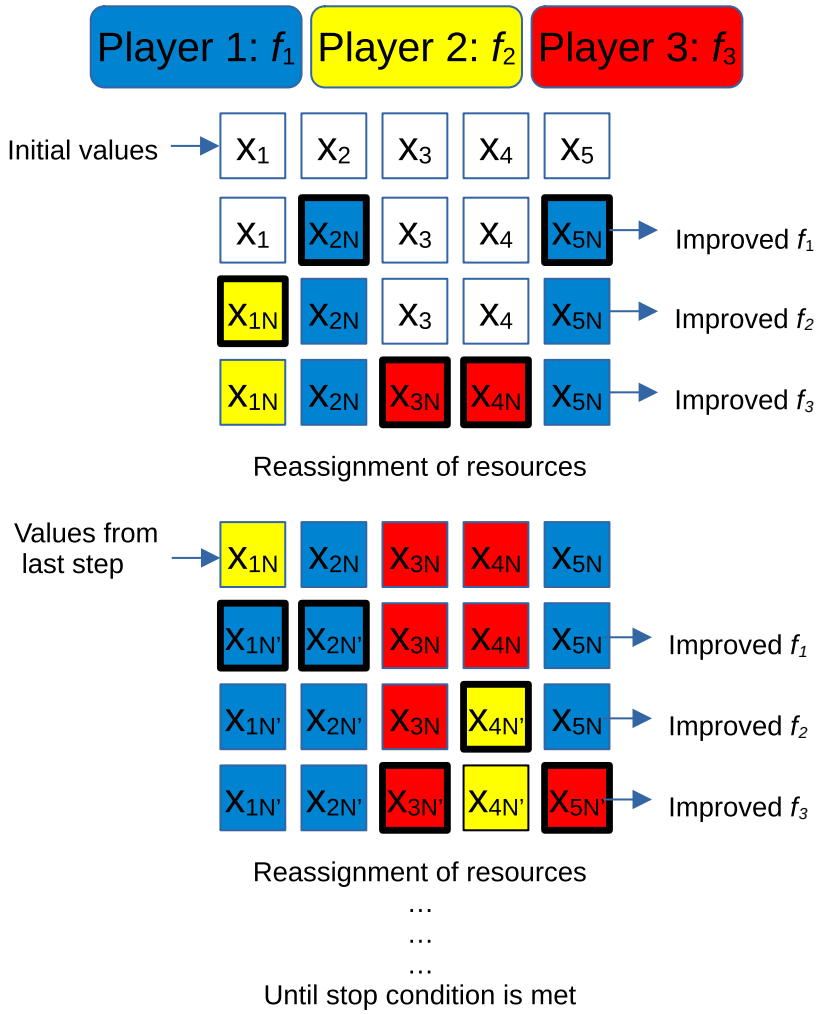


Fig. 31: Simple example of a game theoretic approach. Colours indicate players and bolded frame indicates design variable is changed at the time.

3.5. **Implementation**

DEGT algorithm was implemented in general-purpose programming language C++ using Visual Studio environment under Windows operating system. The algorithm consists of multiple functions related to tasks such as: setting and changing the parameters of the DEGT algorithm, invoking DE algorithm to solve single objective optimization by players, setting design variables as *fixed* or *free to change* depending on resources assigned to a player, displaying the information obtained in the current step on the console screen and saving the information to log files and checking the termination condition.

Parameters of the main algorithm which need to be set before starting the optimization are:

- number of objective functions n_{obj},
- number of design variables n_{dv},
- termination condition (number of iterations *iter* or number of objective function calls *calls*),
- vector of limits imposed on design variables *limits*.

Apart from these algorithms, a set of parameters related to DE algorithm needs to be set, which includes:

- number of iterations of DE algorithm DE_{iter},
- population size DE_{pops},
- crossover rate CR,
- scaling factor F.

All these parameters can be either included in the main code of the program or read by the algorithm from text file, for the purpose of their easy modification before each run of the algorithm without the need of changing the code.

To start the algorithm, it is necessary to use an admissible solution. It is possible to use any solution within the limits imposed on design variables. A first solution can be explicitly defined by user. In this case, this solution can be for example an existing design of a mechanical system which is about to be optimized. Another application of defining a first solution by user is shown in case when the algorithm needs to be restarted after a shutdown. It is then possible to include the last found solution in the text file and the algorithm will continue the course of optimization starting from this point. This approach is especially useful in case of time-consuming optimization processes as it helps to prevent complications caused by energy shutdowns, hardware of software failures and other situations considering either unexpected or scheduled termination of the algorithm. It is worth noting that the algorithm requires only a single last solution to restart, which is an important advantage compared to many algorithms which require a full history of obtained solutions to continue the course of the algorithm,

If user decides to save the logs, then the algorithm can save information on vectors of design variables and vector of objective function values to save file. By default, the program saves information on each solution analysed, including dominated solutions. It is possible to filter out the dominated solutions to obtain the set of non-dominated solution also called Pareto-optimal set or Pareto front.

Additional functions included in the code to supplement the algorithm are related to the calculation of the hypervolume metric for the purpose of testing the algorithm and comparing the value of HV for the solutions obtained by DEGT with solutions obtained with other algorithms. It is possible to calculate hypervolume for any set of solutions for given reference point. For the purpose of a statistical approach to comparison of metrics it is also possible to calculate mean and standard deviation of hypervolumes of multiple sets of solutions.

3.6. **Communication with FEM software**

In case of optimization of mechanical systems, a vital aspect of the DEGT algorithm is its way of communication with FEM software which is responsible for obtaining values of optimization functions. During the course of optimization values of functionals need to be calculated multiple times based on a set of design variables specified by an algorithm. All the

popular FEM systems provide support for parametric models which are extremely useful in case of optimization tasks. In order to obtain value of an optimization functional by FEM software a parametric model including: geometry, material properties, mesh and boundary conditions and initial conditions for dynamic models should be set up beforehand. After the parametric model is designed it should be solved using the set of parameters, which are design variables in the optimization process. FEM system then generates geometrical model based on supplied parameters and after solving it provides the information on sought quality (e.g., deflection at given point, value of maximum equivalent stress) and exports it to a file. DEGT algorithm should be able to generate the model based on parametric model for a chosen set of parameters, run the solver and read the log file to obtain the value of optimization functional. There are many different approaches to setting up parametric models depending on the software of choice. In case of popular MSC software (Patran/Nastran) parametric models can be build and solved utilising the script language called *Patran Command Language* (PCL). In case of another popular FEM software Ansys parametric models can be build using the graphic user interface in the software or using *Ansys Parametric Design Language* (APDL).

It is possible to obtain values of optimization functionals using a function built in in the program code. The function takes a set of design variables as an input data and returns either a set of objective function values, or a single objective value, depending on what is necessary. All the commands related to setting up parametric model based on a set of design variable, solving it, and reading generated text files to obtain values of optimization functionals should be included in the script. This approach might utilise batch files to design and solve problems, depending on CAE system used.

This approach for communication is used in case of a problem of optimization of an airfoil system described in section 5.2, where Patran/Nastran software is responsible for determining values of optimization functionals. In this problem there are 4 optimization functionals, related to: total mass, equivalent stress, displacement, and modal frequency of the airfoil system under working conditions. An in-built function *airfoil* is responsible for obtaining the values of optimization functionals. Function *airfoil* takes two input arguments: a vector of design variables *vars* and an unsigned integer value obj = {1,2,3,4} determining which of four objective functions is calculated at the time and returns the value of the target function as indicated by *obj*. To determine the value of the function, a Patran/Nastran software is utilised. A parametric model is built based on previously prepared session file and presented vector of design variables. Session file is in fact a script in PCL, which is responsible for setting up a model, including: creation of a geometric model based on presented parameters (design variables), meshing the model, setting up material sections and material properties, applying loads and boundary conditions. The aforementioned parts of the process are related to pre-processing and are realised using Patran software, which is typically used with a graphic user interface although in this case, as it's an automatic process, these are all invoked without the need of using the GUI, only based on session file and a batch file to execute session file in background mode using Patran/Nastran environment. After the tasks related to pre-processing are finished, a solver *Nastran* is invoked to solve a boundary-value problem and save report files for post-processing. Solving the boundary-value problem is typically the most time-consuming part of the process. After the problem has been solved, function *airfoil* accesses report files and extract information on the value of target objective function. The value of the objective *output* function is then returned by the *airfoil* function. Fig. 32 shows which part of the process are responsible for selected tasks in case of optimization of an airfoil system.

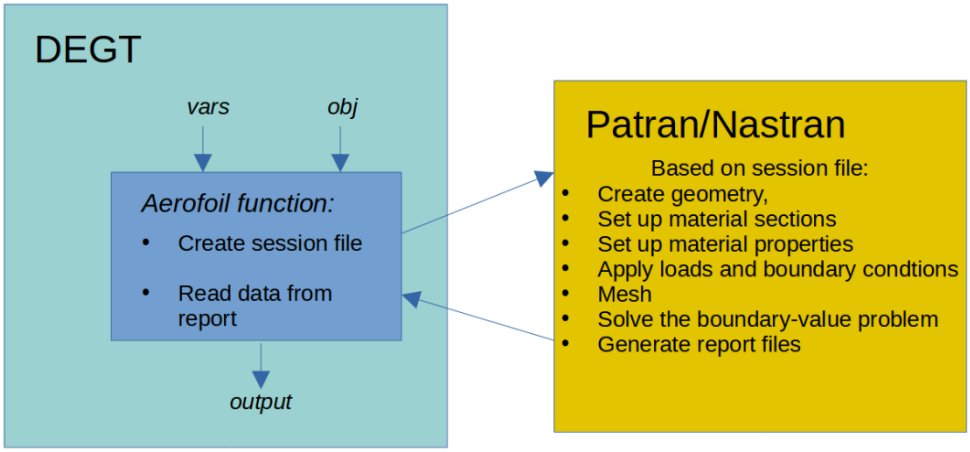


Fig. 32: Communication through internal script in case of optimization of an airfoil system

It is worth noting that some of the objective functions might not need to perform all of the steps described in Patran/Nastran block. For example, calculation of mass requires only creation of geometry, setting up material sections and properties and generation of report file and the intermediate stages should be omitted. If the objective function does not require solving a boundary-value problem, the time to determine its value is substantially shorter.

Another approach to communication doesn't require inference with the code of the algorithm and uses external executable file *solver.exe* instead. If files named *solver.exe, datain.dat* and *dataout.dat* are located in the location where the algorithm is running then these files will be used to obtain values of optimization functionals. *solver.exe* is responsible for solving the problem based on the values of a vector of design variables saved in *datain.dat* file by the algorithm. After solving the problem, the information on optimization functionals should be written in *dataout.dat* file, from which the algorithm obtains sought values. In this approach it is not needed to modify the source code of the algorithm, but it is necessary to design file *solver.exe* able to calculate values of optimization functionals and write them to *dataout.dat* based on a values of design variables in *datain.dat*. Executable file *solver.exe* can be designed, built, and compiled in any way provided it works in a way described beforehand.

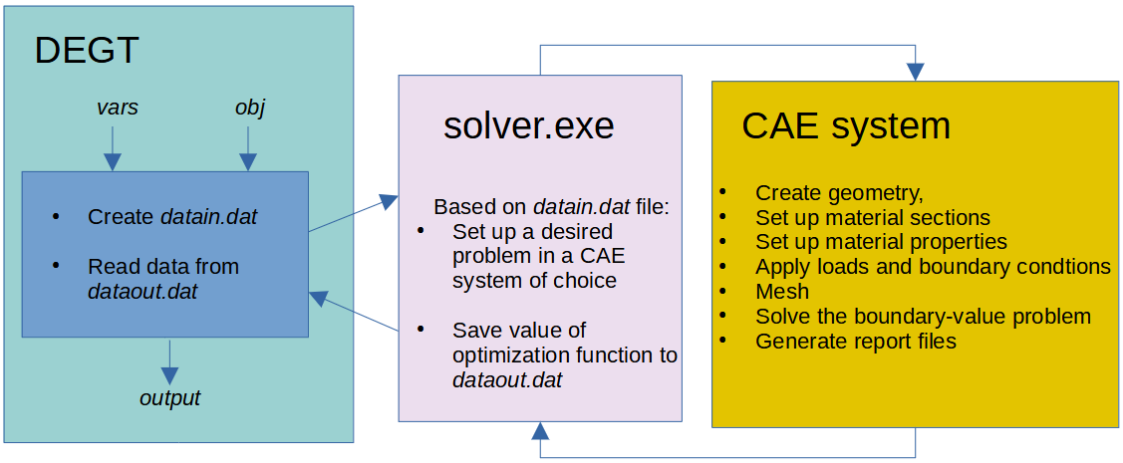


Fig. 33: Communication through *solver.exe* with any CAE system

In the Fig. 33 a process of communication with a CAE system of choice, controlled by an additional executable *solver.exe* and documents *datain.dat* and *dataout.dat* is shown. This approach was used in case of optimization problems related to microactuators and porous materials optimization, described in chapter 5, where communication with two CAE systems: Patran/Nastran and Marc/Mentat was controlled by *solver.exe*.

4. COMPARATIVE TESTS OF THE DEGT ALGORITHM

In order to assess performance of the algorithm it was compared with other well-known algorithms. The algorithms were presented with a set of test functions (benchmark problems) designed in a way to reflect real problems faced by optimization algorithms working with mechanical systems. After a given number of function calls the sets of non-dominated solutions were assessed using comprehensive hypervolume metric. To observe the performance of the algorithm for problems of various dimensionality of solution and design space scalable test functions were used. Test were performed for problems with a range of 3 to 6 objectives and 10 to 20 design variables. For one of the test functions, extra tests were performed for cases with up to 10 objectives. For each instance the algorithm was run 30 times to calculate mean and standard deviation of results. The number of 30 runs is understood as a compromise between reliability and accuracy of obtained statistical results and low computational time required to obtain them. Results analysed were: number of solutions in the set of non-dominated solutions and hypervolume metric. To calculate hypervolume metric it is necessary to use a reference point. Reference point was chosen in a way to be dominated by all the solutions considered in each case and may vary between variants. Termination condition for all the cases was 25000 objective function calls. In case of test functions used, the execution time of such a task is negligible, although in case of real mechanical problems, where FEM-based software is used to obtain value of optimization functions, assuming solving a boundary value problem takes 30 seconds, 25000 calls translate to over a week of computational time. Depending on the details of the problem and especially on the number of nodes in the FEM model used to obtain values of optimization functionals, this time can be considerably increased. For this reason, it is an essential feature of optimization algorithms to be able to obtain a good quality of solutions under a limited number of function calls.

Differential evolution parameters used inside DEGT algorithm were:

- generations $DE_{iter} = 5$,
- population size $DE_{pops} = 6$,
- crossover rate $CR = 0,5$,
- scaling factor $F = 0,7$.

Population size and number of generations are uncharacteristically low for a regular single-objective optimization problem. When optimization is subject to only one fitness function, the objective space should be explored in one direction only, trying to focus on improving a single quantity described by fitness function. This approach is especially prevalent in gradient methods. In this case, however, the idea is to perform single-objective optimization multiple times, such an approach can be understood as taking many small steps in various directions in the objective space as opposed to taking few large steps. Taking the small steps results in more diversified set of solutions and the algorithm is less likely to stuck in local optima, as the assignment of resources is often changing. It is worth noting that the contradictory nature of objectives results in the possibility to worsen other objectives when a contradictory objective is being improved. This serves as a protection mechanism, discouraging the algorithm to perform extensive exploitation of objective space around local minima.

4.1. **DAGT vs NSGA-II vs NSGA-III**

The algorithms chosen to be compared with DEGT are NSGA-II published in 2002 [42] which is still among the most popular multiobjective algorithms. MATLAB, a popular multi-purpose engineering and scientific computing software utilises a controlled, elitist genetic algorithm (a variant of NSGA-II) in Global Optimization Toolbox. Function *gamultiobj* is responsible for genetic multiobjective optimization. All the results related to NSGA-II were calculated using MATLAB R2021a.

Second algorithm, NSGA-III were published in 2013 [50] and is an improved version of NSGA-II. Authors did not share the implementation of the algorithm so an unofficial C++ implementation [52] was used for the purpose of comparison.

For the purpose of comparison, the default values of parameters, proposed by authors of implementations are used in case of both NSGA-II and NSGA-III.

The default values of parameters of NSGA-II are available in MATLAB documentation [115] and default values of parameters of NSGA-III are included in the documentation available on the implementation author's webpage [52].

In case of both NSGA-III and DEGT the algorithms were implemented in C++ so code of test functions was interchangeable. In case of NSGA-II test functions were rewritten from C++ to MATLAB language.

4.2. **Results of the comparison**

Results of the comparison are presented in the form of tables and graphs. Each of the following sections is devoted to a specific test function described in detail in section 2.6. For each scalable test function a set of variants scaled in terms of number of objectives was analysed In case of real test problems, they were transformed to a multiobjective problems by treating inequality constraints as minimization fitness functions, where left side of the inequality is considered to be a minimization function. Data presented in the table includes:

- number of objective functions n_{obj},
- number of design variables n_{dv},
- reference point $r_p = [h, h, \dots, h]$ of dimensionality equal to the number of objective functions,
- mean hypervolume metric *hv* calculated over 30 runs of the algorithm,
- standard deviation *sd* of said hypervolume metric set,
- Pareto size *ps*.

Desired results are: high mean value of hypervolume, low value of standard deviation and high mean value of Pareto size. High hypervolume indicates results are close to true Pareto front, well spread and distributed. Low standard deviation indicates good reproducibility of results. Large Pareto size gives analysts a bigger set of solutions to choose from in the final decision making process.

4.2.1. **DTLZ1**

Results of comparison between 3 algorithms for DTLZ1 are shown in Tab. 3 and Fig 34. Best values for each respective test were highlighted.

Tab. 3: Test function DTLZ1

DTLZ1		*rp*	DEGT			NSGA-II			NSGA-III		
n_{obj}	n_{dv}	*h*	*hv*	*sd*	*ps*	*hv*	*sd*	*ps*	*hv*	*sd*	*ps*
3	10	100	9,11E+05	2,24E+04	45,5333	8,91E+05	1,23E+05	**69,133**	**1,00E+06**	**2,40E-01**	25,00
4	10	100	9,81E+07	5,12E+05	65,6	4,95E+07	2,65E+07	**69,6**	**1,00E+08**	**1,74E+01**	57,00
5	10	100	9,92E+09	3,70E+07	68,83	6,29E+09	2,46E+09	69,8	**1,00E+10**	**8,73E+03**	**129,00**
6	20	500	1,53E+16	1,33E+14	109,6	1,26E+16	1,20E+15	70	**1,56E+16**	**2,22E+11**	**253,00**

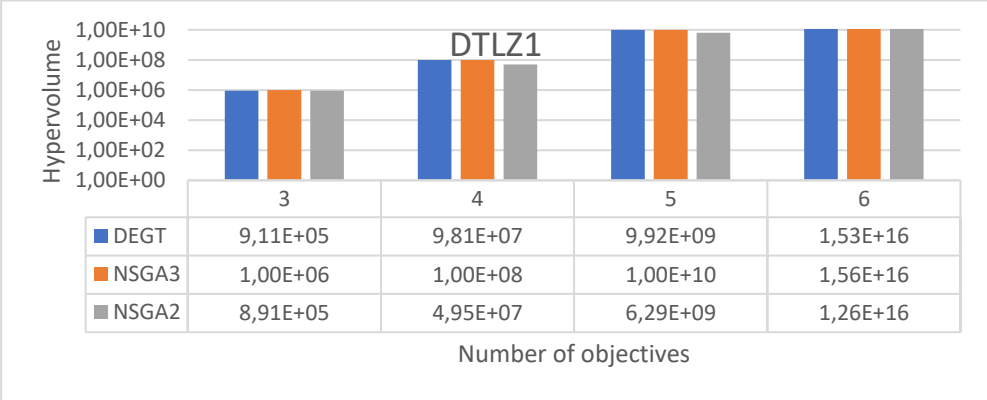


Fig. 34: Test function DTLZ1

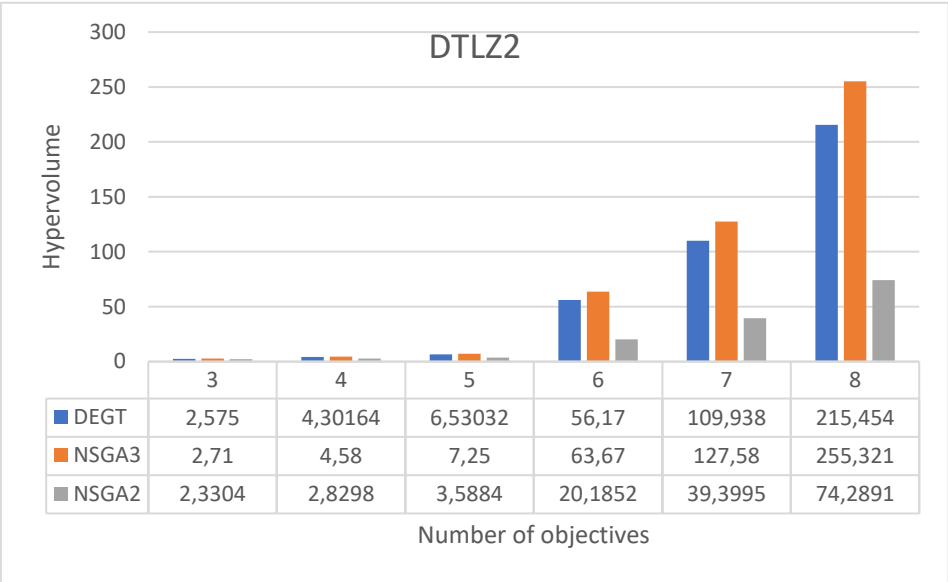
For the DTLZ1 function the superior results in terms of mean hypervolume and standard deviation were obtained by NSGA-III. DEGT was proven superior to NSGA-II. The differences between results in hypervolume obtained by all three algorithms weren't significant, especially for higher dimensional problems. In case of Pareto size, NSGA-III performed significantly better for 6 objectives and significantly worse for 3.

4.2.2. DTLZ2

Results of comparison between 3 algorithms for DTLZ2 are shown in Table 4 and Fig 35. Comparison with NSGA-II was extended to problems with up to 10 objective functions. Comparison with NSGA-III was performed only for problem with up to 8 objective functions, as further calculations of HV for a high-dimensional Pareto fronts with a large Pareto size were extremely time-consuming, but it should be expected for NSGA-III to outperform both algorithms for further increased number of objectives. Best values for each respective test were highlighted.

Tab. 4: Test function DTLZ2

DTLZ2		*rp*	DEGT			NSGA-II			NSGA-III		
n_{obj}	n_{dv}	*h*	*hv*	*sd*	*ps*	*hv*	*sd*	*ps*	*hv*	*sd*	*ps*
3	10	1,5	2,575	0,021	**77,9**	2,3304	0,39917	70	**2,71**	**0,001**	25
4	10	1,5	4,30164	0,0339291	**115,367**	2,8298	0,65657	70	**4,58**	**0,0009**	57
5	10	1,5	6,53032	0,122122	123,233	3,5884	1,0186	70	**7,25**	**0,001**	**129**
6	20	2	56,17	1,7387	91,5667	20,1852	8,9495	70	**63,67**	**0,01**	**253**
7	20	2	109,938	4,49336	83,7333	39,3995	20,9655	72,1667	**127,58**	**0,02**	**465**
8	20	2	215,454	8,94555	73,6667	74,2891	47,2734	73,9	**255,321**	**0,05**	**793**
9	20	2	**431,511**	**16,804**	65,433	165,5719	84,9747	**77,3**	-	-	-
10	20	2	**867,913**	**31,52**	67	345,6197	114,9292	**78,73**	-	-	-



	3	4	5	6	7	8
DEGT	2,575	4,30164	6,53032	56,17	109,938	215,454
NSGA3	2,71	4,58	7,25	63,67	127,58	255,321
NSGA2	2,3304	2,8298	3,5884	20,1852	39,3995	74,2891

Fig. 35: Test function DTLZ2

For 3 to 8 objectives NSGA-III outperformed both algorithms, except for Pareto size in case of 3 and 4 objectives, where DEGT obtained larger sets of non-dominated solutions. NSGA-II was significantly worse than other two algorithms.

Comparison with NSGA-II for a problem scaled to 9 and 10 objectives was for clarity not included in the figure although it should be noted that superiority of DEGT does further increase with the increase of the number of objectives.

4.2.3. DTLZ3

Results of comparison between 3 algorithms for DTLZ3 are shown in Tab. 5 and Fig 36. Best values for each respective test were highlighted.

Tab. 5: Test function DTLZ3

DTLZ3		*rp*	DEGT			NSGA-II			NSGA-III		
n_{obj}	n_{dv}	*h*	*hv*	*sd*	*ps*	*hv*	*sd*	*ps*	*hv*	*sd*	*ps*
3	10	200	6,58E+06	4,43E+05	37,433	7,55E+06	4,50E+05	**68,2667**	**8,00E+06**	**3,49**	25
4	10	200	1,53E+09	2,80E+07	50,9333	6,76E+08	4,25E+08	**70**	**1,60E+09**	**40,82**	57
5	10	200	3,15E+11	2,82E+09	62,9667	1,80E+11	6,76E+10	70	**3,20E+11**	**4,30E+04**	**129**
6	20	1000	9,58E+17	1,29E+16	86,0667	5,68E+17	1,66E+17	70	**9,99E+17**	**4,23E+12**	**253**

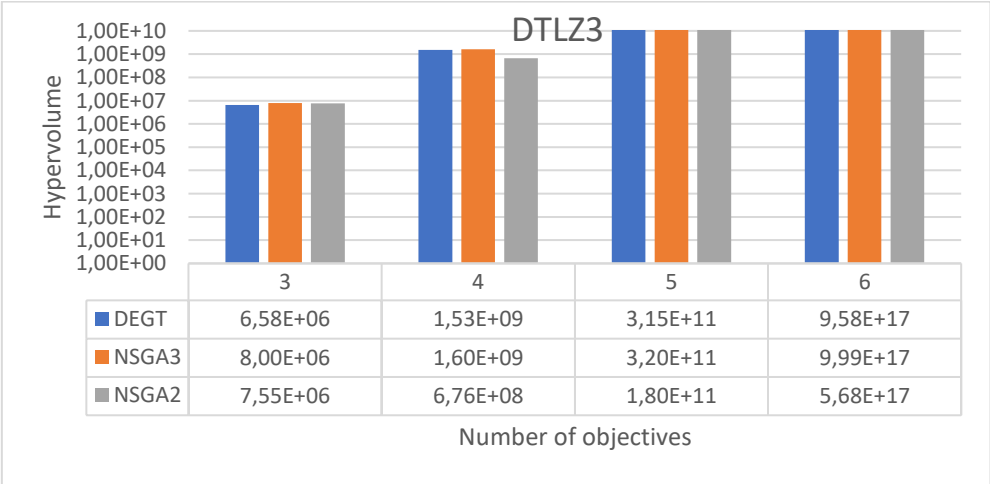


Fig. 36: Test function DTLZ3

In case of DTLZ3 test function, the results were quite similar although in favour of NSGA-III. Both NSGA-II and DEGT found more solutions on Pareto front than NSGA-III in case of problem scaled to 3 and 4 objectives.

4.2.4. DTLZ4

Results of comparison between 3 algorithms for DTLZ4 are shown in Tab. 6 and Fig 37. Best values for each respective test were highlighted.

Tab. 6: Test function DTLZ4

DTLZ4		*rp*	DEGT			NSGA-II			NSGA-III		
n_{obj}	n_{dv}	*h*	*hv*	*sd*	*ps*	*hv*	*sd*	*ps*	*hv*	*sd*	*ps*
3	10	1,5	**2,35603**	**0,0798805**	18,3667	2,3237	0,38922	**70**	2,13	0,46	25
4	10	1,5	4,028	**0,129**	34,4	3,2666	0,65374	**70,4667**	**4,29**	0,42	57
5	10	1,5	6,505	0,189	53,867	5,0251	1,1466	70,2333	**7,26**	**0,001**	**129**
6	20	2	58,961	1,629	36,5	37,1281	10,9649	73,5	**63,65**	**0,03**	**253**

For the DTLZ4 problem, DEGT was superior in terms of hypervolume and standard deviation for problem with 3 objectives. For problems with 4 objectives DEGT recorded lowest value of standard deviation among compared algorithms. For problems with 5 and 6 objectives NSGA-III outperformed remaining two algorithms.

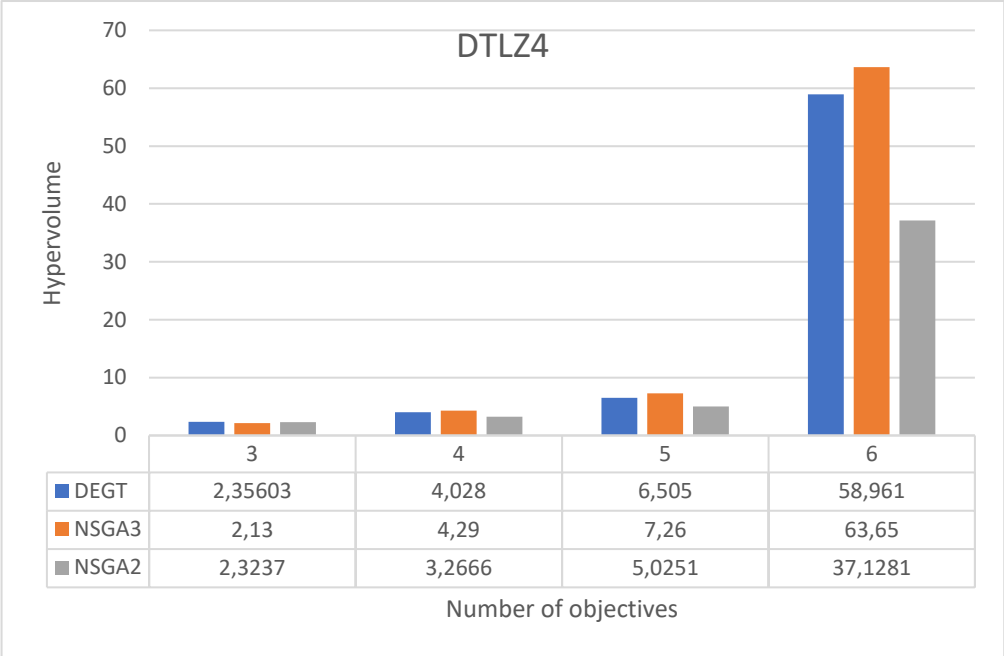


Fig. 37: Test function DTLZ4

4.2.5. WFG1

Results of comparison between 3 algorithms for WFG1 are shown in Tab. 7 and Fig 38. Best values for each respective test were highlighted.

Tab. 7: Test function WFG1

WFG1		*rp*	DEGT			NSGA-II			NSGA-III		
n_{obj}	n_{dv}	*h*	*hv*	*sd*	*ps*	*hv*	*sd*	*ps*	*hv*	*sd*	*ps*
3	10	1,5	5,81022	6,13936	**316**	2,789	**0,36315**	**200**	**7,6**	6,97	25
4	10	1,5	3,655	2,61	**444,1**	4,5895	**0,46396**	**200**	**8,5**	9,14	57
5	10	1,5	4,99	**0,113**	**461,633**	7,0836	0,66203	**200**	**7,25**	0,19	129
6	20	1,5	10,9098	15,6042	**453,233**	10,952	0,68744	200	**11,27**	**0,4**	**253**

NSGA-III obtained results with highest mean value of hypervolume. However, it is worth noting that for problems with 3, 4 and 6 objectives, obtained hypervolume for NSGA-III and DEGT recorded a particularly high value of standard deviation which indicates the results were not repetitive among 30 runs. For these problems NSGA-II obtained results with lower standard deviation. For problem with 5 objectives DEGT found the most repetitive results indicated by a low value of standard deviation. For this problem DEGT recorded the most solutions on the Pareto front for all examined cases.

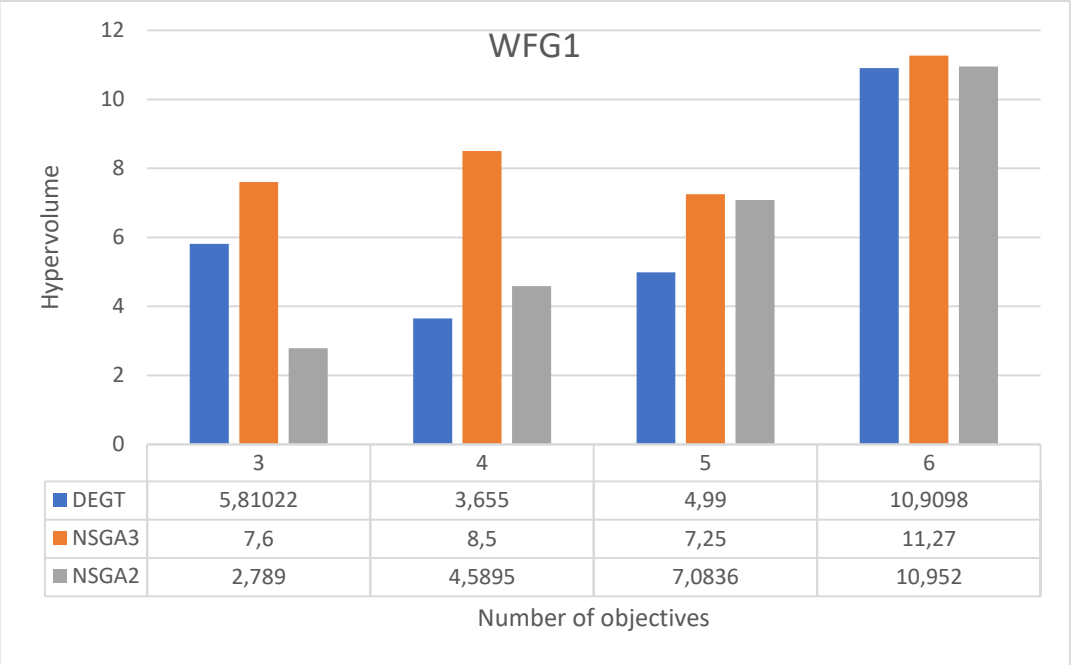


Fig. 38: Test function WFG1

4.2.6. WFG4

Results of comparison between 3 algorithms for WFG4 are shown in Tab. 8 and Fig 39. Best values for each respective test were highlighted.

Tab. 8: Test function WFG4

WFG4		*rp*	DEGT			NSGA-II			NSGA-III		
n_{obj}	n_{dv}	*h*	*hv*	*sd*	*ps*	*hv*	*sd*	*ps*	*hv*	*sd*	*ps*
3	10	100	**4,228**	2,005	192,8	2,6192	**0,05585**	**200**	2,64	0,006	25
4	10	100	4,414	1,136	236,633	3,9786	**0,12505**	**200**	**5,59**	3,38	57
5	10	100	**8,689**	2,357	170	5,9205	**0,23546**	**200**	7,87	5,75	129
6	20	500	**64,254**	19,139	164,633	8,7369	**0,30123**	200	61,79	0,09	**253**

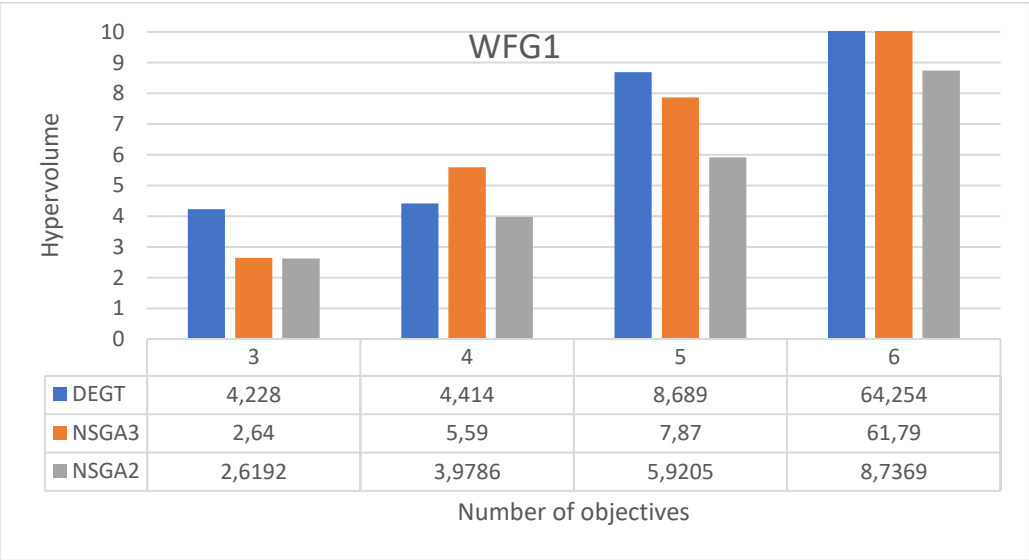


Fig. 39: Test function WFG1

It should be noted that in the Fig. 39 the limits on the vertical axis are set to a value lower than the maximum value obtained for the DEGT in case of *nobj = 6* because it significantly exceeds other results.

For test problem WFG4 DEGT outperformed other algorithms in terms of mean hypervolume for problem with 3, 5 and 6 objectives. For problem with 4 objectives NSGA-III obtained better solutions. It is worth noting that results obtained by NSGA-III end DEGT are characterised by high standard deviation and both these algorithms had a worse repetitiveness rate than NSGA-II nevertheless quality of solutions found by NSGA-II was worse. Over 30 runs in many cases for DEGT and NSGA-III there were several particularly good solutions, but the median hypervolume of solutions was much lower than the mean. For DEGT the results were more repetitive, with low standard deviation for all cases.

4.3. **Conclusions**

In general, the tests carried out on a set of benchmark functions indicate that DEGT can be considered a viable multiobjective optimization algorithm. The performance of the algorithm for most cases was noticeably superior compared to the performance of a NSGA-II algorithm and slightly worse than NSGA-III algorithm, although there were some instances in which DEGT performed better than both algorithms it was compared with. It should be noted, that for either of the compared values: mean hypervolume, standard deviation and mean Pareto size it is possible to find a test function and a selected number of objectives, for which DEGT outperformed both algorithms. For example: for WFG4, for most cases DEGT recorded the best mean values of HV. It is worth noting that WFG4 is a complex test function involving features characteristic to many real mechanical optimization problems. For WFG1 in all cases DEGT found the highest mean value of number of solutions on the Pareto front. For some cases in DTLZ4 and WFG1, the results obtained by DEGT were characterized with the lowest value of standard deviation of hypervolume among compared algorithms, which indicates a more repetitive nature of obtained set of solutions. Even though NSGA-III outperformed DEGT in many cases analysed, the difference was not significant and considerably lower than the difference between NSGA-II and DEGT performance, when DEGT was proven superior. The tests on mathematical benchmark functions produced satisfying results, proving that the algorithm can find good enough results in terms of examined metrics and further tests on problems considering mechanical systems with both analytical and numerical formulation of objective functions will be presented in the following part of this dissertation.

5. REAL MECHANICAL PROBLEMS

In this chapter performance of DEGT was presented on real mechanical multiobjective optimization problems. Six mechanical optimization problems with a varying number of objectives and design variables and diverse level of complexity were examined. The former three optimization problems are analytical problems, transformed from single objective optimization benchmark problems with constraints to form multiobjective unconstrained optimization problems. These problems do not require extensive numerical effort to solve any optimization objectives and therefore are relatively easy in terms of computational effort and time required to be solved. The latter three problems, on the other hand, are complex mechanical optimization problems, which require numerical solving of boundary-value problems during the course of optimization. In particular: static structural, modal, thermal and multiscale analyses need to be performed using FEM-based CAE software. These tasks are time consuming processes and hence the need to present an efficient tool optimization tool to produce satisfying results under satisfying time limits. The last problem, optimization of a multiscale porous material, is particularly time-consuming process, as calculation of any of the objective functions requires a preliminary set of 6 analyses in micro scale to determine effective elastic constants, 3 analyses in micro scale to determine effective thermal constants and 1 final macro-scale analysis to determine optimization functionals. In result, every single candidate solution during the optimization process needs 10 boundary-value problems to be solved in order to determine values of related objective functions.

5.1. Selected analytical mechanical problems

Real mechanical problems naturally deal with many of the difficulties which are artificially introduced in test functions. Most test problems related to real mechanical systems are single objective problems. Many of these problems include constraints and therefore can be transformed to multiobjective problems, by treating inequality constraints as additional fitness functions. Real mechanical problems take both discrete and continuous values as working parameters, unlike test functions described before, which only use continuous values. In some cases, the discrete values of working parameters can be substituted with continuous parameters (e.g., arbitrary thickness instead of a standard thickness) but in some cases it wouldn't be physically possible (e.g., number of teeth of a gear can't be represented by a continuous value). In this chapter a selection of 3 analytical mechanical problems based on transformed test problems is examined.

5.1.1. Pressure vessel design

Test problem considering pressure vessel design (PVD) was introduced in [116]. It is a problem with 4 design variables and 4 constraints and was considered by many researchers [117]–[120].

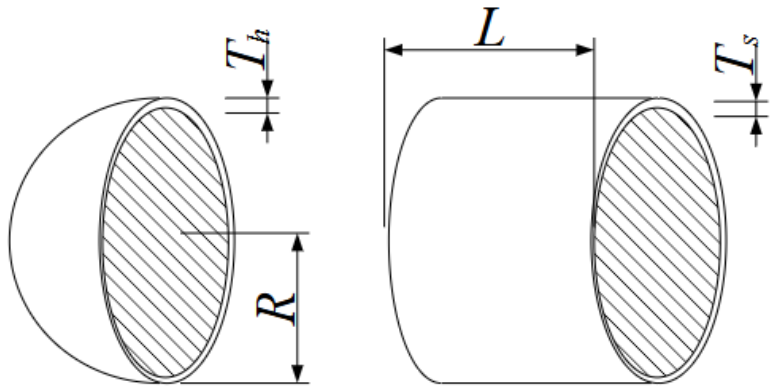


Fig. 40: Pressure vessel

A cylindrical pressure vessel is capped at both ends by hemispherical heads (Fig. 40) and is supposed to hold medium under pressure. Vessel is designed to work under pressure of 3000 psi and its volume must not be lower than 750 ft^3. Unlike other examples in this case imperial unit system is used instead of metric, following the way it was originally introduced. Design of the vessel must follow the American society of mechanical engineers (ASME) boiler and pressure vessel code [121]. Total cost, including cost of the materials, welding and forming is to be minimized. Design variables, optimization function and constraints are shown in Tab. 9:

Tab. 9: Pressure Vessel Design problem: design variables, fitness function and constraints

symbol	expression or description	notes
T_h	Thickness of head, $T_h \leq 99 \times 0.0625$, multiple of 0.0625 in.	design variable
T_s	Thickness of shell, $T_s \geq 0.0625$, multiple of 0.0625 in.	design variable
R	Radius, $R \geq 10$	design variable
L	Length of cylindrical section, $L \leq 200$	design variable
f	$f(T_s, T_h, R, L) = 0.6224 T_s R L + 1.7781 T_h R^2 + 3.1661 T_s^2 L + 19.84 T_h^2 L$	fitness function to be minimized
h_1	$-T_s + 0.0193R \leq 0$	geometrical constraint
h_2	$-T_h + 0.0095R \leq 0$	geometrical constraint
h_3	$-\pi R^2 L - \frac{4}{3}\pi R^3 + 750 \times 11728 \leq 0$	volume constraint
h_4	$L - 240 \leq 0$	geometrical constraint

It should be noted that for the aforementioned set of constraints, due to the upper bound of L being 200, the h_4 constraint is satisfied automatically. Some researchers [122]–[125] decided to expand the search region and adjust the upper bond of L to 240.

Left side of the h_i inequalities for $i \in \{1, \dots, 4\}$ can be considered as minimization functions for the purpose of transforming the single-objective problem into a many-objective problem.

In a result of transformation where left side of the constraints inequality is treated as function to be minimized, a new multiobjective optimization problem with 5 objective functions is created. All the design variables were considered in a continuous domain according to limits described in Tab. 9. A resulting set of optimization functions is shown in Tab. 10.

Tab. 10: Transformed Pressure Vessel Design optimization functions

expression	notes
$f_1(T_s, T_h, R, L) = 0.6224T_sRL + 1.7781T_hR^2 + 3.1661T_s^2L + 19.84T_h^2L$	minimization
$f_2(T_s, R) = -T_s + 0.0193R$	minimization
$f_3(T_h, R) = -T_h + 0.0095R$	minimization
$f_4(R, L) = -\pi R^2 L - \frac{4}{3}\pi R^3 + 750 \times 11728$	minimization
$f_5(L) = L - 240$	minimization

After transformation of the constraints into minimization functions, the solutions preciously violating the constraints are now deliberately accepted in the multiobjective problem. One of the aspects of multiobjective optimization and its significant advantage over single objective optimization is the fact that it provides extended information on the trade-offs between objectives and this feature is explored in this approach. To keep the information on the satisfied constrained, the functions are transformed in a way that resulting negative value indicates related constraint is satisfied and positive value means it is constrained. The proposed change significantly broadens the feasible region in the optimization problem and therefore results cannot be directly compared with those of a single-objective constrained problem. The best solutions for the single-objective problem found in the literature [126]–[128] produce the value of f_1 = 6059.7 approximately. The multiobjective problem was solved using DEGT and following parameters:

- generations DE_{iter} = 5,
- population size DE_{pops} = 6,
- crossover rate CR = 0.5,
- scaling factor F = 0.7,
- termination condition: 25 000 function calls.

Due to a simple nature of the analytical problem, the problem did not require extensive computational effort and was solved under aforementioned parameters in under one minute using a mid-range personal computer.

The results consist of a set of 449 non-dominated solutions, among which 58 solutions satisfy all the constraints imposed on the original problem. The best solution in terms of f_1 function only produces the value of f_1 = 424.962. For this solution, the value of f_4 is positive, which translates to a violation of h_3 (volume) constraint from the original problem. Among found solutions 76.17% fulfil the original h_1 constraint, 32.29% fulfil the original h_3 constraint and 100% solutions fulfil h_2 and h_4 constraints. This is an additional information on the nature of the problem and constraints imposed in it, which wouldn't have been obtained when considering the problem as single-objective only.

Discarding the solutions violating any of the constraints, the best solution found produces the value of f_1 = 11856.1 which is considerably worse than the solutions found previously by researchers considering single-objective problem, which was expected due to the expanded solution space which was explored by multiobjective optimizer disregarding constraints. The remaining value of the objective functions for aforementioned solution were f_2 = -0.3299, f_3 = -0.5821, f_4 = -43799.6 and f_5 = -179.496. For the purpose of visualisation and decision making, only the solutions satisfying all the constraints were taken. Solutions were presented in the form of a scatter-plot matrix including histograms (Fig 41) and in the form of 3D plot, with one of the objectives represented by colour (Fig. 42).

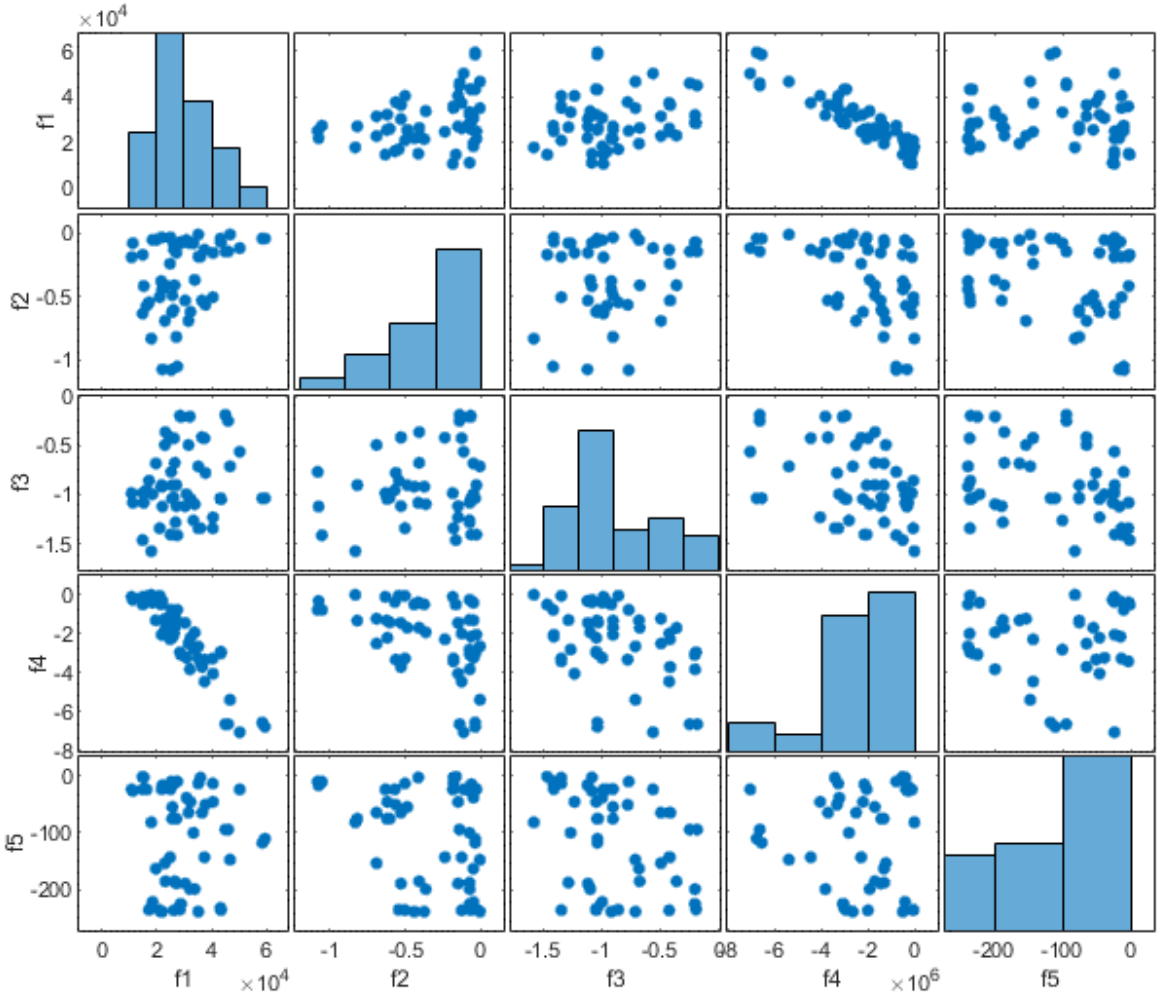


Fig. 41: Scatter plot matrix and histograms of obtained solutions for PVD.

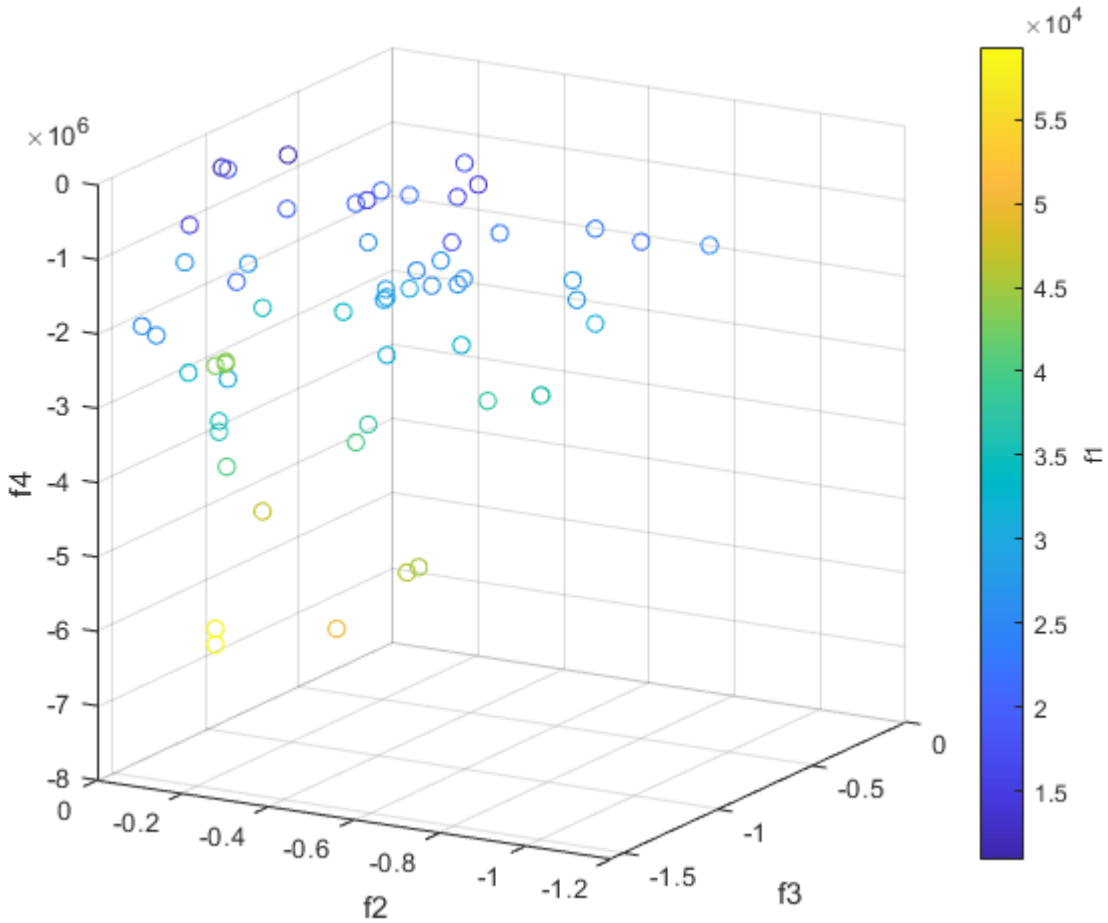


Fig. 42: 3D plot with a colour bar of obtained solutions for PVD.

In case of visualisation using 3D plot and a colour bar, the f_5 objective was disregarded, so the information conveyed in this way are limited. In case of this specific problem, the objective omitted is derived from a constraint which is always fulfilled due to limits imposed on design variables, but in general, visualisation of data in this manner in case of problems with more than 4 objectives is always encumbered with a loss of information.

Alternatively, obtained data set can be presented using parallel plotting. The same data presented using this approach is shown in the Fig. 43. For clarity, all the values of optimization functionals were normalized to values in the range [0,1] for each objective function separately, because the range of objective was of a significantly different magnitude. Normalized value of objectives is presented in the vertical axis, horizontal axis represents 5 objectives analysed and lines are solutions.

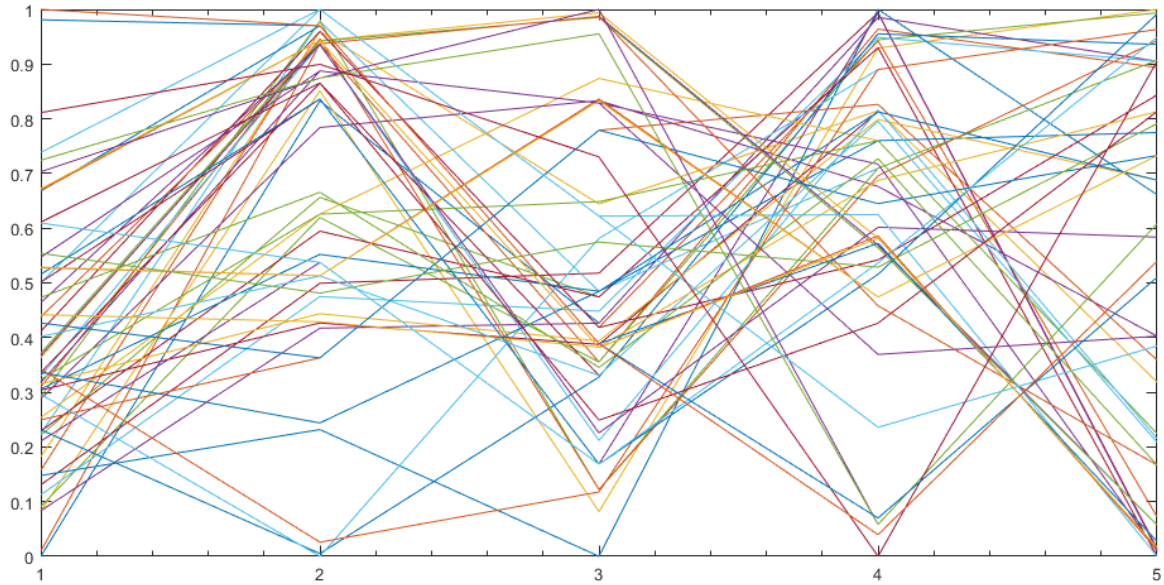


Fig. 43: PVD solutions visualised using parallel plotting method

To make conclusions on the values of optimization functionals based on the graph in Fig. 43, it is necessary to know the minimum and maximum value of each optimization function before normalization and therefore this way of presenting the results can only be used when supplemented with other methods or by additional information on the said values.

Including solutions violating original constraints new designs were found in a range between 425.521 and 60857.5 for the first objective with median value of 5261.26. For the remaining objectives, related to constraints, min, median and max values are shown in the Tab. 11.

Tab. 11: Min, median and max values of objectives.

Objective	**Min**	**Median**	**Max**
$\boldsymbol{f_2}$	-1.7654	-0.6019	1.2975
$\boldsymbol{f_3}$	-1.8965	-1.304	-0.1475
$\boldsymbol{f_4}$	-9.21363e+06	1.04989e+06	1.29088e+06
$\boldsymbol{f_5}$	-238.931	-148.271	-0.2115

Apart from information on the values of objective, some conclusions on the nature of trade-offs between them can be drawn. Results indicate a contradictory nature of objectives f_1 and f_4 whereas f_2 and f_3 seem to be consistent with each other in large areas of the search space.

5.1.2. Speed reducer design

Speed reducer design problem was introduced in [129] and considers a problem of minimization of volume of gear wheels and transmission shafts (Fig. 44). The problem is described by 7 design variables and 11 constraints related to permissible stresses and deflections in the system. Problem was solved by many researchers [130]–[133]. Design variables, simple bounds imposed on them, optimization function and constraints are listed in the Tab. 12.

Tab. 12: Speed Reducer problem: design variables, fitness function and constraints

symbol	expression or description	notes
b	face width, $2.6 \leq b \leq 3.6$	design variable
m	module of teeth, $0.7 \leq m \leq 0.8$	design variable
z	number of teeth, $17 \leq z \leq 28$, integer	design variable
l_1	length of the first shaft between bearings, $7.3 \leq l_1 \leq 8.3$	design variable
l_2	length of the second shaft between bearings, $7.3 \leq l_2 \leq 8.3$	design variable
d_1	diameter of the first shaft, $2.9 \leq d_1 \leq 3.9$	design variable
d_2	diameter of the second shaft, $5.0 \leq d_2 \leq 5.5$	design variable
f	$f(b,m,z,l_1,l_2,l_3,l_4,d_1,d_2) = 0.7854bm^2(3.3333z^3 + 14.9334z - 43.0934) - 1.508b(d_1^2 + d_2^2) + 7.477(d_1^3 + d_2^3) + 0.7854(l_1 d_1^2 + l_2 d_2^2)$	fitness function to be minimized
h_1	$\frac{27}{bm^2z} - 1 \leq 0$	constraint
h_2	$\frac{397.5}{bm^2z^2} - 1 \leq 0$	constraint
h_3	$\frac{1.925l_1^3}{mzd_1^4} - 1 \leq 0$	constraint
h_4	$\frac{1.925l_2^3}{mzd_2^4} - 1 \leq 0$	constraint
h_5	$\frac{\sqrt{\left(\frac{745l_1}{mz}\right)^2 + 1.69 \times 10^{-6}}}{110d_1^3} - 1 \leq 0$	constraint
h_6	$\frac{\sqrt{\left(\frac{745l_1}{mz}\right)^2 + 157.5 \times 10^{-6}}}{85d_2^3} - 1 \leq 0$	constraint
h_7	$\frac{mz}{40} - 1 \leq 0$	constraint
h_8	$\frac{5m}{b-1} - 1 \leq 0$	constraint
h_9	$\frac{b}{12m} - 1 \leq 0$	constraint
h_{10}	$\frac{1.5d_1 + 1.9}{l_1} - 1 \leq 0$	constraint
h_{11}	$\frac{1.1d_2 + 1.9}{l_2} - 1 \leq 0$	constraint

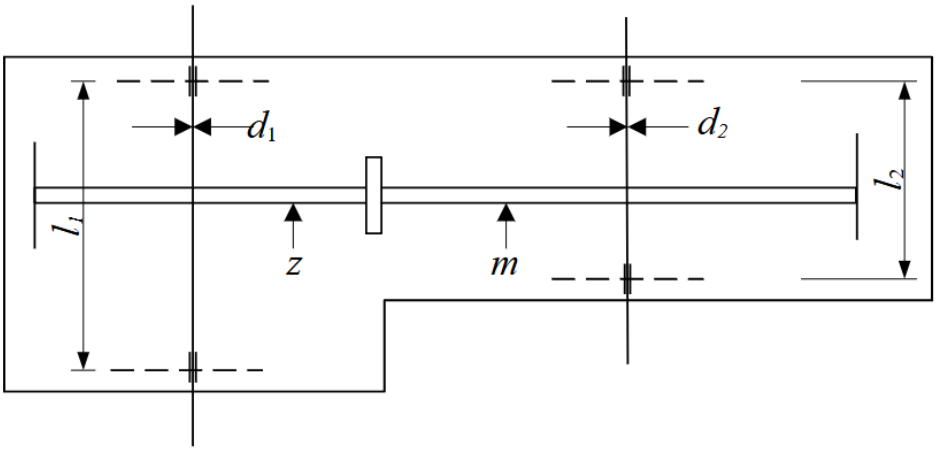


Fig. 44: Speed reducer design

Left side of the h_i inequalities for $i \in \{1, \dots, 11\}$ can be considered as minimization functions for the purpose of transforming the single-objective problem into a many-objective problem.

For the purpose of transforming this constrained single-objective problem into a multiobjective optimization problem the constraints are transformed to minimization optimization functions. In a result of transformation where left side of the constraints inequality is treated as function to be minimized, a new multiobjective optimization problem with 12 objective functions is created. All the design variables were considered in a continuous domain according to limits described Tab. 12. A resulting set of optimization functions is shown in Tab. 13.

Tab. 13: Transformed Speed Reducer optimization functions

expression	notes
$f_1(b,m,z,l_1,l_2,l_3,l_4,d_1,d_2) = 0.7854bm^2(3.3333z^3 + 14.9334z - 43.0934) - 1.508b(d_1^2 + d_2^2) + 7.477(d_1^3 + d_2^3) + 0.7854(l_1 d_1^2 + l_2 d_2^2)$	minimization
$f_2(b,m,z) = \frac{27}{bm^2 z} - 1$	minimization
$f_3(b,m,z) = \frac{397.5}{bm^2 z^2} - 1$	minimization
$f_4(m,z,d_1,l_1) = \frac{1.925 l_1^3}{mzd_1^4} - 1$	minimization
$f_5(m,z,d_2,l_2) = \frac{1.925 l_2^3}{mzd_2^4} - 1$	minimization
$f_6(m,z,d_1,l_1) = \frac{\sqrt{\left(\frac{745 l_1}{mz}\right)^2 + 1.69 \times 10^{-6}}}{110 d_1^3} - 1$	minimization
$f_7(m,z,d_1,l_1) = \frac{\sqrt{\left(\frac{745 l_1}{mz}\right)^2 + 157.5 \times 10^{-6}}}{85 d_2^3} - 1$	minimization
$f_8(m,z) = \frac{mz}{40} - 1$	minimization
$f_9(b,m) = \frac{5m}{b-1} - 1$	minimization
$f_{10}(b,m) = \frac{b}{12m} - 1$	minimization
$f_{11}(d_1,l_1) = \frac{1.5d_1 + 1.9}{l_1} - 1$	minimization
$f_{12}(d_2,l_2) = \frac{1.1d_2 + 1.9}{l_2} - 1$	minimization

Like in previous problem, the functions were transformed in a way that the resulting positive value of an objective function indicates violation of a related constraint in the original problem, but the violation of these constraints is allowed. The original constrained single-objective problem presents a difficulty for optimization algorithms in exploring the space of admissible solutions due to the number and nature of constraints [134]. The best solutions for the single-objective problem found in the literature [135] produce the value of f_1 = 3000 approximately, although many researches accept solutions with slightly violated constraints. The multiobjective optimization with DEGT was performed using the same parameters as in the previous problem and did not require extensive numerical effort due to the analytical formulation of the optimization functionals.

The results consist of a set of 358 non-dominated solutions, among which 6 solutions satisfy all the constraints imposed on the original problem. The best solution in terms of f_1 function only produces the value of f_1 = 2540.31. For this solution, the value of f_2, f_3, f_6, f_7 and f_9 are positive, which translates to a violation of four constraints from the original problem. The

number of solutions satisfying related constraints in the original problem are presented in Tab. 14. Constraints h_4, h_9 and h_{11} are fulfilled by 100% of solutions and were therefore omitted in the table.

Tab. 14: Percentage of solutions satisfying constraints

Constraint	h_1	h_2	h_3	h_5	h_6	h_8	h_{10}	h_{11}
% of satisfied solutions	98.04	99.44	99.44	48.88	43.3	5.87	85.2	90.78

It can be noted that for a problem formulated in this way, the constraint which was significantly more often violated than any others was the h_8 which is related to the ratio between module of teeth and face width of the speed reducer. This brings information that the algorithm tends to explore areas of the solution space which are originally excluded due to the constraint.

Discarding the solutions violating any of the constraints, the best solution found produces the value of f_1 = 3273.65. The remaining value of the objective functions for aforementioned solution were f_2 = -0.1145, f_3 = -0.248, f_4 = -0.6371, f_5 = -0.8994, f_6 = -0.3574, f_7 = -0.0091, f_8 = -0.6957, f_9 = -0.0156, f_{10} = -0.5767, f_{11} = -0.0405 and f_{12} = -0.0277.

It is worth noticing the value of f_7 which is the closest to positive value among constraint-related objectives and it is derived from the value of h_8, previously proven to be the most likely to be unsatisfied.

Solutions, including these which did not satisfy constraints are presented in the Fig 45 in the form of a scatter plot. Due to the large number of objectives, the data is hard to visualize and to present it in an alternative way Kohonen's Self-Organizing Maps (SOMs) can be used (Fig. 46), although it must be noted that SOM approach, unlike scatter plot matrix, is only an approximation of the data set.

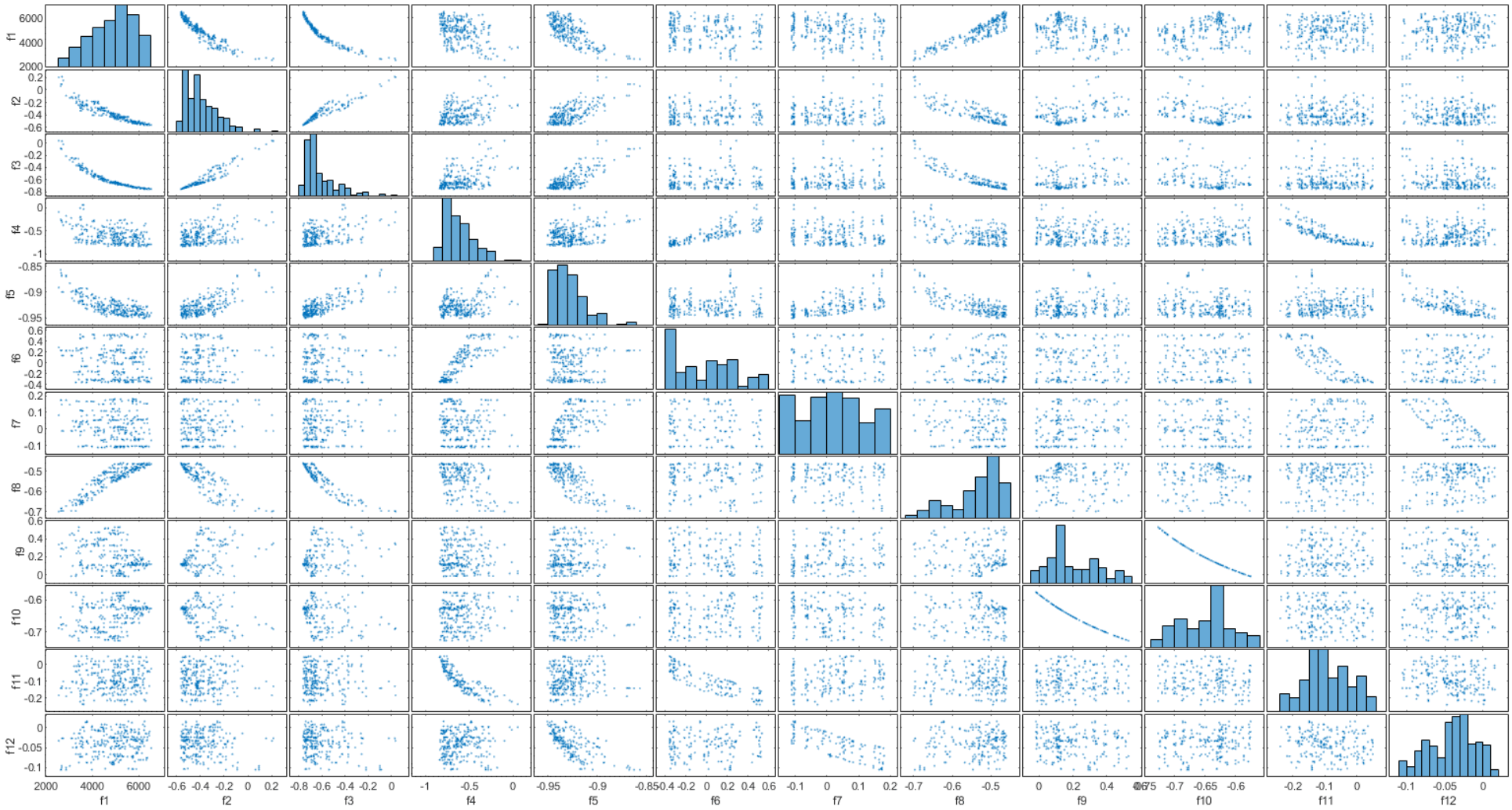


Fig. 45: Scatter plot of obtained solutions for SR

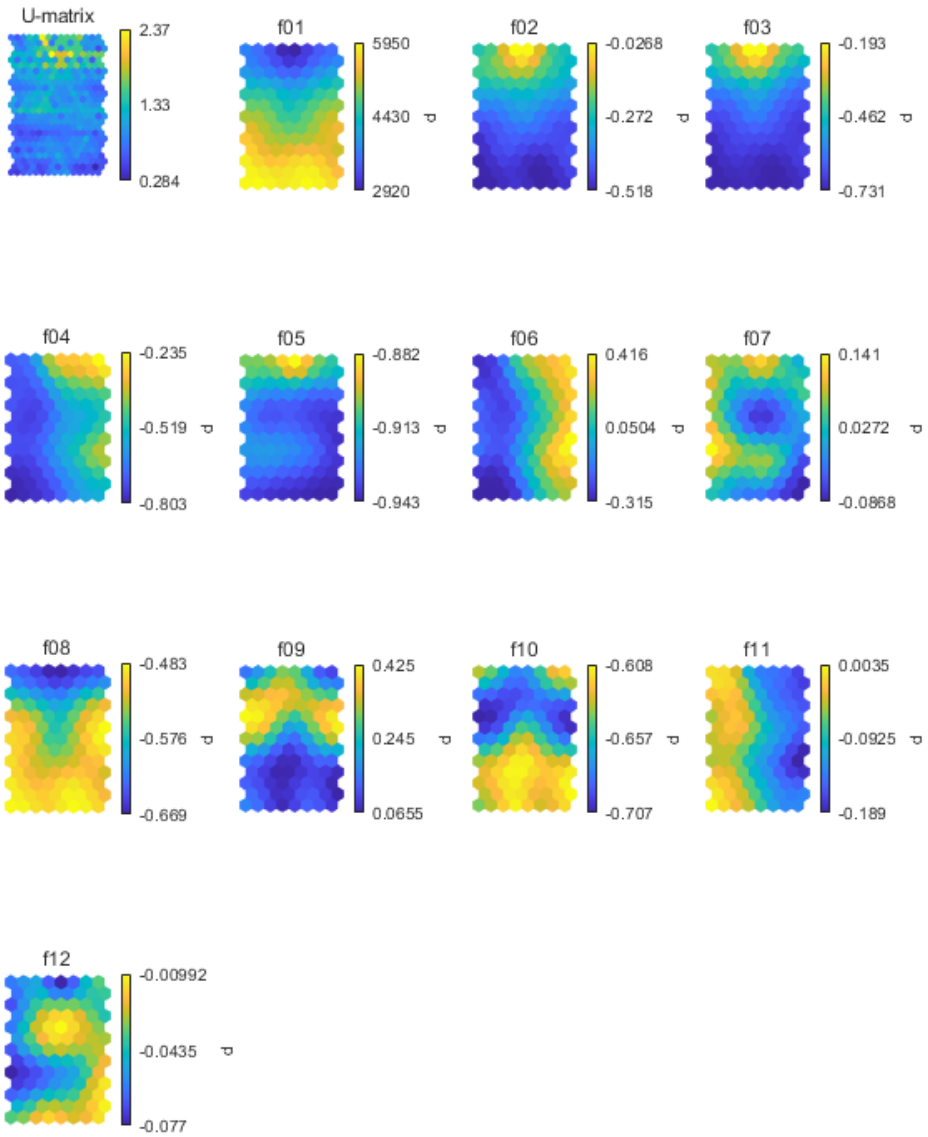


Fig. 46: SOMs of obtained solutions for SR

Based on the observation of Fig. 45 and Fig. 46 some conclusions on the nature of trade-offs between objectives can be drawn. Objectives f_2, f_3 and f_5 all seem to have a similar nature while objectives f_1 and f_8 are contradictory to them. Pairs of objectives f_9 with f_{10} and f_6 with f_{11} present contradictory nature as well.

5.1.3. Stepped cantilever beam

Stepped cantilever beam problem was introduced in [136] as a problem of a loaded beam with 10 design variables and 11 constraints. It can also be considered as a problem with 15 design variables, adding 5 design variables related to lengths of parts of the beam as shown in [137]. Constraints in this problem are related to deflection, stresses from bending and a geometrical constraint of a ratio of dimensions of cross section of the beam. Fitness function in this problem is total volume of the beam to be minimized. Problem was solved by many

researchers [136], [138]–[140]. Tab. 15 presents design variables, fitness function and constraints for the variant with 10 design variables, assuming lengths l of steps are equal.

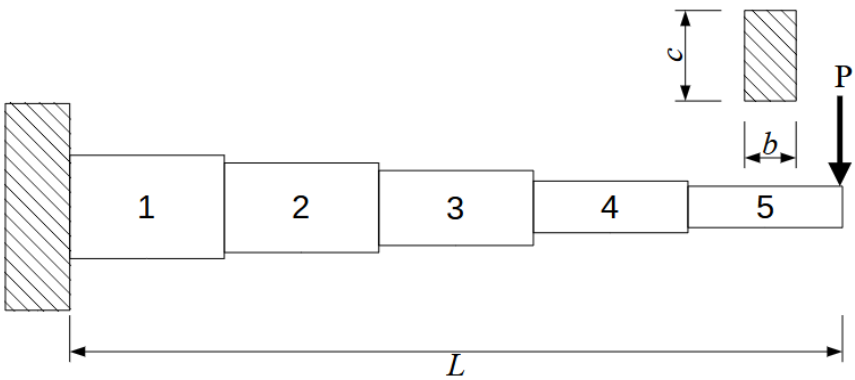


Fig. 47: Stepped cantilever beam

Tab. 15: Stepped Cantilever Beam problem: design variables, fitness function and constraints

symbol	expression or description	notes
b_i	width of step $i, b_i \leq 15cm$	design variable, $i \in \{1, \dots 5\}$
c_i	height of step $i, c_i \leq 150cm$	design variable, $i \in \{1, \dots 5\}$
f	$f = \sum_{i=1}^{5} l b_i a_i$	fitness function to be minimized, total volume of the beam
h_1	$\frac{6Pl}{b_5 c_5^2} - \sigma_d \leq 0$	constraint, bending stress
h_2	$\frac{12Pl}{b_4 c_4^2} - \sigma_d \leq 0$	constraint, bending stress
h_3	$\frac{18Pl}{b_3 c_3^2} - \sigma_d \leq 0$	constraint, bending stress
h_4	$\frac{24Pl}{b_2 c_2} - \sigma_d \leq 0$	constraint, bending stress
h_5	$\frac{30Pl}{b_1 c_1^2} - \sigma_d \leq 0$	constraint, bending stress
h_6	$\frac{Pl^3}{3E}\left(\frac{1}{I_5} + \frac{7}{I_4} + \frac{19}{I_3} + \frac{37}{I_2} + \frac{61}{I_1}\right) - \delta_{max} \leq 0$	constraint, allowable deflection
h_i	$\frac{c_{i-6}}{b_{i-6}} - 20 \leq 0$	constraint, geometrical ratio of width and height in a cross section of *(i-6)-th* step, $i \in \{7, \dots, 11\}$
l	$\frac{L}{5} = 100$ cm	uniform length of a step
σ_d	$14 \frac{\text{kN}}{\text{cm}^2}$	allowable bending stress
δ_{max}	2,7 cm	allowable deflection
P	50kN	load
E	210 GPa	Young's modulus

Left side of the h_i inequalities for $i \in \{1, \dots, 11\}$ can be considered as minimization functions for the purpose of transforming the single-objective problem into a many-objective problem.

The Stepped Cantilever Beam problem is a constrained single objective optimization problem with 10 design variables and 11 constraints in which optimization function is minimizing the total volume of the beam loaded with a transverse force. For the purpose of transforming this problem into a multiobjective optimization problem the constraints are transformed to minimization optimization functions. In a result of transformation where left side of the constraints inequality is treated as function to be minimized, a new multiobjective optimization problem with 12 objective functions is created. All the design variables were considered in a continuous domain according to limits described Tab. 15. A resulting set of optimization functions is shown in Tab. 16.

Tab. 16: Transformed Stepped Cantilever Beam optimization functionals

expression	notes
$f_1(b_1, b_2, b_3, b_4, b_5, c_1, c_2, c_3, c_4, c_5) = \sum_{i=1}^{5} l b_i c_i$	minimization
$f_2(b_5, c_5) = \frac{6Pl}{b_5 c_5^2} - \sigma_d$	minimization
$f_3(b_4, c_4) = \frac{12Pl}{b_4 c_4^2} - \sigma_d$	minimization
$f_4(b_3, c_3) = \frac{18Pl}{b_3 c_3^2} - \sigma_d$	minimization
$f_5(b_2, c_2) = \frac{24Pl}{b_2 c_2^2} - \sigma_d$	minimization
$f_6(b_1, c_1) = \frac{30Pl}{b_1 c_1^2} - \sigma_d$	minimization
$f_7(b_1, b_2, b_3, b_4, b_5, c_1, c_2, c_3, c_4, c_5) = \frac{Pl^3}{3E}\left(\frac{1}{I_5} + \frac{7}{I_4} + \frac{19}{I_3} + \frac{37}{I_2} + \frac{61}{I_1}\right) - \delta_{max}$	minimization
$f_i(b_{i-7}, c_{i-7}) = \frac{c_{i-7}}{b_{i-7}} - 20 \leq 0$	minimization $i \in \{8, \dots, 12\}$
$l = \frac{L}{5} = 100$ cm	uniform length of a step
$\sigma_d = 14 \frac{\text{kN}}{\text{cm}^2}$	allowable bending stress
$\delta_{max} = 2{,}7$ cm	allowable deflection
$P = 50$kN	load
$E = 210$ GPa	Young's modulus

Like in previous problem, the functions were transformed in a way that the resulting positive value of an objective function indicates violation of a related constraint in the original problem, but the violation of these constraints is allowed. In case of a constrained single objective problem, the values of f_1 reported in the literature [141]–[143] are f_1 = 65 000 approximately. The multiobjective optimization was performed using DEGT and a set of

parameters used for the previous problems. Similar to previous problems, this one did not require extensive computational effort due to analytical formulation of objective functions and computation took less than one minute.

The DEGT algorithm found a set of 467 non-dominated solutions, among them 20 solutions satisfied all the constraints from the original problem. The best solution in terms of f_1 displayed value of f_1 =21314. This solution violated constraints $g_{1:6}$ related to allowable stresses and deflections. Unlike in previous problem, neither of the constraints were fulfilled by all the solutions found. The constraint g_6 related to allowable deflection, represented by objective f_7 was the least satisfied constrained among found solutions (Tab. 17).

Tab. 17: Number of solutions satisfying constraints

Constr.	h_1	h_2	h_3	h_4	h_5	h_6	h_7	h_8	h_9	h_{10}	h_{11}
% of satisfied solutions	68.73	52.46	54.81	50.96	46.9	10.27	94.21	89.51	96.14	97	94

Considering only solutions which satisfied all the constraints, the best found solution in terms of f_1 presents values of optimization functionals $\mathbf{f} = [f_1, f_2, \dots, f_{12}] = [134040, -113.844, -90.9418, -59.2356, -36.8462, -40.5781, -7.5539, -10.7468, \dots$

$\dots -15.1291, -15.0449, -8.7626, -7.5086]$.

Again, the value of functional f_7 derived from constraint which was least likely to be satisfied displays low value, which could be understood as a situation in which fulfilling this condition and at the same time having it as low as possible results in a considerably good solution in terms of f_1.

All the solutions obtained during the optimization are shown in the Fig. 48.

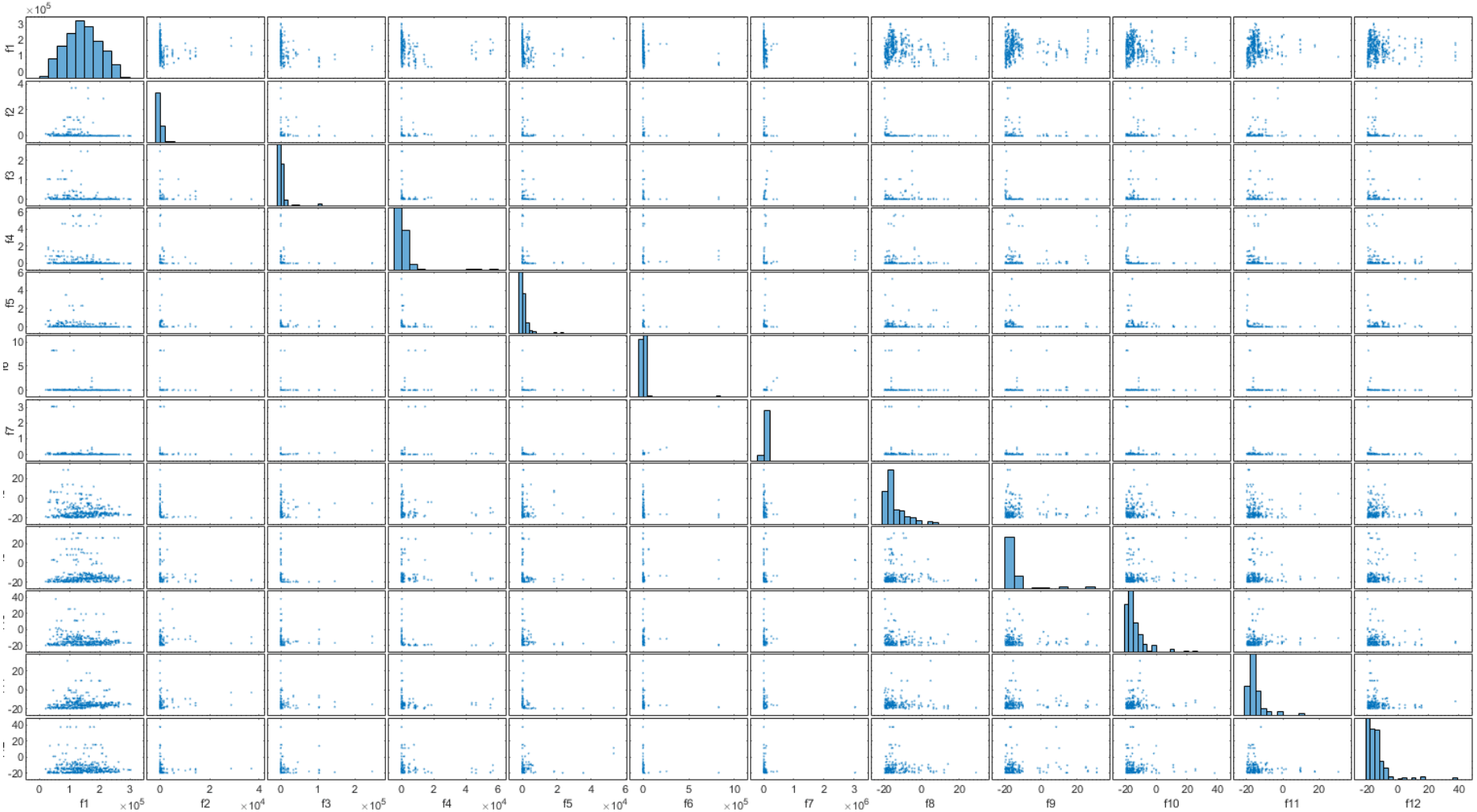


Fig. 48 Scatter plot of obtained solutions for SCB

Based on produced scatter plot it is hard to draw clear conclusions regarding trade-off between objectives in the problem. The information on max, min and median values of objectives is presented in the table.

Tab. 18: Min, median and max values of objectives.

Objective	Min	Median	Max
f_1	21314	139827	302134
f_2	-133.836	-109.81	36843.6
f_3	-128.288	-26.6082	245866
f_4	-123.173	-30.3791	56852.2
f_5	-114.919	-4.7904	53044.1
f_6	-111.165	42.3916	827778
f_7	-18.621	1148.98	3.04596e+06
f_8	-19.8827	-15.7228	28.9659
f_9	-19.9308	-16.3686	31.0357
f_{10}	-19.8054	-15.7488	38.1436
f_{11}	-19.8562	-15.9233	30.568
f_{12}	-19.7738	-15.4993	37.659

Analysing values in the above table some information on the objectives related to constraints which are hard to be satisfied can be obtained. Objectives f_6 and f_7 which correspond to constraints h_5 and h_6 exhibit positive median value, which provides information on their nature being relatively hard to fulfil compared to other constraints.

5.2. **Optimization of an airfoil**

Airfoil systems are used to develop an aerodynamic force when moving through the fluids. When designing the optimal shape of an airfoil many, often contradictory, criteria must be taken into consideration to provide the system with the desired properties, such as high endurance, low weight and high lifting force, low aerodynamic drag, high or low stiffness and more.

To meet the multiple requirements asked of airfoil systems multiobjective optimization methods can be taken advantage of and enhance the process of design of such structures.

Airfoils are used in many machines and devices including: propellers, rotors, turbines, sailing boats, windsurfers, flying vehicles, etc. In this study, the numerical example considered in the optimization problem is based on a real structure of a wing of unmanned aerial vehicle (UAV).

The outer shape of an airfoil consists of a suction surface and a pressure surface, which contribute to generation of a lifting force. Designing of an outer shape of an airfoil requires computational fluid dynamic simulation which is not covered in this example. The inner shape of an airfoil on the other hand has a significant impact on other properties of an airfoil systems, such as stiffness, durability, modal properties, and weight. To reduce the mass of a system, instead of using full profiles, girders are used to connect and support the outer surfaces. The

design of an inner shape of an airfoil is thus crucial in the process of an optimal design of airfoil systems as it accounts for fulfilling many requirements asked of them.

Modern airfoil systems are usually built of durable lightweight materials, such us composites, often having anisotropic mechanical properties.

Considered airfoil design consists of polyurethane foam and composite materials reinforced with carbon and glass fabric and a woven roving to connect the girder and the outer panel.

Optimization of airfoil systems was considered by many researchers previously. Lam et al. presented multiobjective optimization of an aerostructural design using Kriging Model [144]. Ceruti et al. compared hueristic techniques, such as genetic algorithm, particle swarm optmization and Monte Carlo method in multiobjective optimization of UAVs manufactured by means of rapid prototyping techniques [145]. Berci at al presented a multidisciplinary multifidelity approach to airfoil optimization of a small UAV [146].

5.2.1. Aims and assumptions

Aim of this study is to find an optimal design among a set of Pareto-optimal designs of an inner geometry of an airfoil system under mechanical loads representing working conditions. The optimization process is performed for four criteria at once and thus a multiobjective optimization approach is used. Main assumptions of the research include:

- Outer geometry of the airfoil is not subject to the optimization task. Only the structural optimization of an inner geometry is performed, and the outer shape is assumed to be fixed and thus no computational fluid dynamics (CFD) analyses are performed during the optimization course.
- Boundary conditions for the static structural analysis are based on a preliminary CFD analysis of an airfoil under working conditions.
- Parametric numerical model, with geometry based on a wing of a real unmanned aerial vehicle (UAV) is used as a numerical example.
- FEM software (MSC Patran/Nastran) is used to solve boundary value problems in order to determine values of optimization functionals.
- Linear-elastic material models are assumed, isotropic and orthotropic for selected areas of the model.

The preliminary CFD analysis was performed in Ansys Fluent software on a model consisting of 363 138 nodes and 257 276 tet10 elements (Fig. 49) assuming velocity magnitude $v = 36\frac{m}{s}$ and angle of attack $\alpha = 3°$.

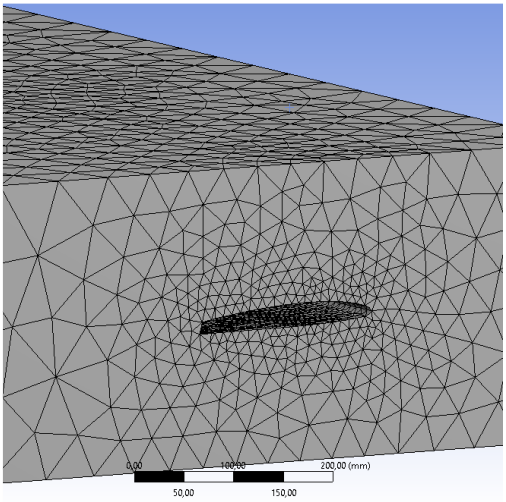


Fig. 49: Discretization of the CFD model

The resulting system holds Reynolds number $R_e = 506$. The magnitude of pressure applied to the wing in the working conditions was determined from a distribution of pressure in the model (Fig. 50).

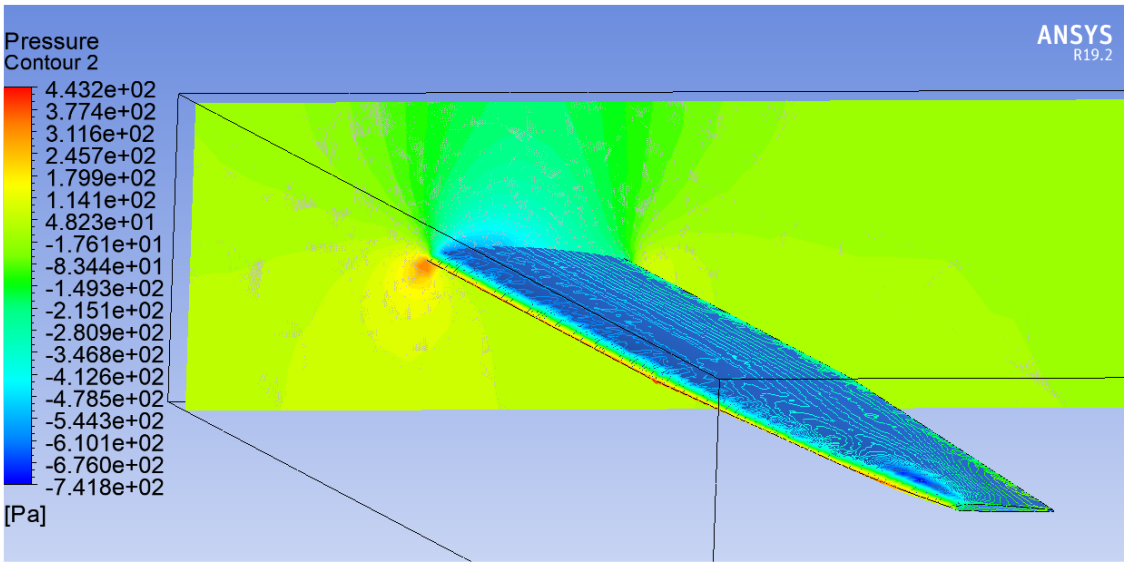


Fig. 50: Pressure distribution in the CFD model

Polyurethane foam is described with isotropic properties and the remaining materials with orthotropic properties. Linear-elastic constitutive model is described by formula (41):

$$\begin{bmatrix}\sigma_{11}\\ \sigma_{22}\\ \sigma_{12}\end{bmatrix}=\begin{bmatrix}c_{11} & c_{12} & 0\\ & c_{22} & 0\\ . & & c_{33}\end{bmatrix}\begin{bmatrix}\varepsilon_{11}\\ \varepsilon_{22}\\ \varepsilon_{12}\end{bmatrix} \tag{41}$$

Where σ_{ij}, ε_{ij}, c_{ij} are the elements of Cauchy stress, strain, and effective elastic constants tensors. These materials are used to define four laminates used in the airfoil. Distribution of materials is shown in the Fig. 51.

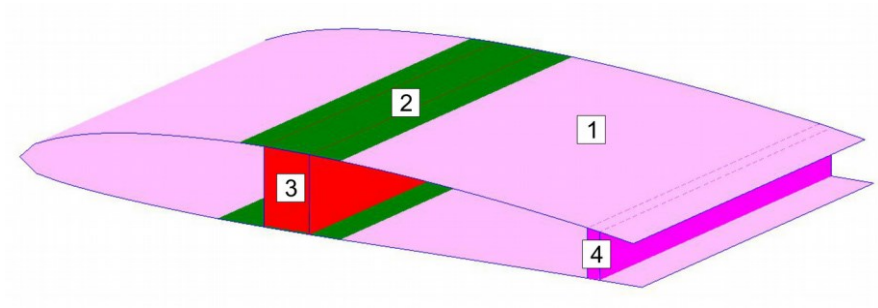


Fig. 51: Distribution of materials. 1 – carbon fabric 62 g/m^2, foam, glass fabric 24 g/m^2; 2 – carbon fabric 62 g/m^2, 3 – carbon fabric 160 g/m^2; 4 – glass fabric 24 g/m^2.

This work is an extension of previous studies [147], [148], during which the numerical model was experimentally validated. Three-point bending test was conducted on a real structure on universal testing machine (MTS Insight System) showing a satisfactory similarity of results with the numerical model.

5.2.2. Formulation of the problem

Objectives in the optimization problem are devoted to minimization of maximal equivalent stresses, minimization of maximal displacements, minimization of total mass of the model and maximization of difference between values of modal frequency of a system and a given frequency – these are the examples of conditions required of airfoil systems, although other criteria based on specific needs can be formulated. Values of functionals used as objectives are computed using FEM simulations. Boundary conditions and material properties are fixed during the optimization run and geometry of the model, described by design variables is changed to fit the declared needs, formally described by functionals (42) – (45):

Minimization of the maximal value of the equivalent (von Mises) stress of an airfoil under given load:

$$\min_{x} f_1 = \max(\sigma_{eq}) \tag{42}$$

Minimization of the maximal translational displacement of an airfoil under given load:

$$\min_{x} f_2 = \max(u) \tag{43}$$

Maximization of the minimal difference between any of first 10 modal frequencies f_n of an airfoil and a reference frequency $f_r = 30$ Hz:

$$\max_{x} f_3 = \min(|f_n - f_r|) \ \forall_{n\in\{1,2,\dots,10\}} \tag{44}$$

Minimization of the total mass of an airfoil:

$$\min_{x} \; f_4 = \int_{\Omega} d\Omega \tag{45}$$

Values of aforementioned functionals are calculated om the base of FEM results obtained in MSC Patran/Nastran software utilising a parametric model. Functionals f_1 and f_2 are calculated by performing a static structural analysis under a given load. Functional f_3 is calculated by running a modal analysis. Problem of maximization of f_3 is transformed to a minimization problem by multiplying a fitness function value by -1. Functional f_4 is calculated on the basis of geometry of the model in preprocessor Patran and does not require solving a boundary-value problem.

5.2.3. Numerical model

Example of parametric numerical was build based on real structure of an UAV wing (Fig. 52). Approximate dimensions of the structure are: total length ~1800 mm and transverse direction ~200 mm [147].

Fig. 52: Real structure of a UAV wing on which the numerical model was based.

Wing consists of two parts: centerwing and end part. The aileron is not included in the numerical model as its effect on the load transfer is negligible. Three characteristic cross sections of a wing can be distinguished: end cross section, middle cross section, and base cross section. These geometries are controlled by 24 design variables, defining the inner shape of the wing. Design variables and constraints imposed on them are described in Fig. 53 and Tab. 19.

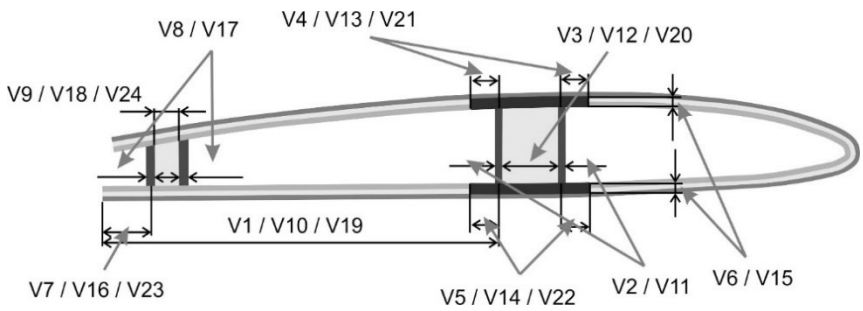


Fig. 53: Design variables controlling the inner shape of the wing.

Tab. 19: Design variables and limits imposed on them

Symbol	Design variable	Lower limit [mm]	Upper limit [mm]
V1	position of the main girder (base cross section)	-5	5
V2	thickness of the main girder (centerwing)	0.1	0.48
V3	thickness of the main girder (base cross section)	-7	7
V4	length of the upper roving (base cross section)	-5	5
V5	length of the bottom roving (base cross section)	-5	5
V6	thickness of the roving (centerwing)	0.1	2
V7	position of the small girder (base cross section)	-5	5
V8	thickness of the carbon fabric (centerwing)	0.05	1
V9	thickness of the small girder (base cross section)	-3	3
V10	position of the main girder (middle cross section)	-5	5
V11	thickness of the carbon fabric (end part of the wing)	0.1	0.48
V12	thickness of the main girder (middle cross section)	-7	7
V13	length of the upper roving (middle cross section)	-5	5
V14	length of the bottom roving (middle cross section)	-5	5
V15	thickness of the roving (end part of the wing)	0.1	2
V16	position of the small girder (middle cross section)	-5	5
V17	thickness of the carbon fabric (end part of the wing)	0.05	1
V18	thickness of the small girder (middle cross section)	-3	3
V19	position of the main girder (end cross section)	-4	4
V20	thickness of the main girder (end cross section)	-7	7
V21	length of the upper roving (end cross section)	-5	5
V22	length of the bottom roving (end cross section)	-5	5
V23	position of the small girder (end cross section)	-4	4
V24	thickness of the small girder (end cross section)	-1	2

The model is discretized using hex8 and quad4 elements, and the resulting mesh consists of approximately 21500 nodes and 19000 elements, depending on the shape of an inner structure.

Preliminary computational fluid dynamics study was performed in Ansys Fluent software to determine the pressure distribution of the wing under working conditions, assuming velocity magnitude: $v = 36\frac{\mathrm{m}}{\mathrm{s}}$ and angle of attack $\alpha = 3°$. The analysis performed helped establish simplified boundary conditions for the following static structural analyses performed during the optimization course.

Simplified boundary conditions applied in the numerical model include a fix in all degrees of freedom on one side of the model and an evenly distributed total load $P = 5\mathrm{kN}$ on the bottom surface. Boundary conditions are shown in the Fig. 54 with the geometry flipped upside down

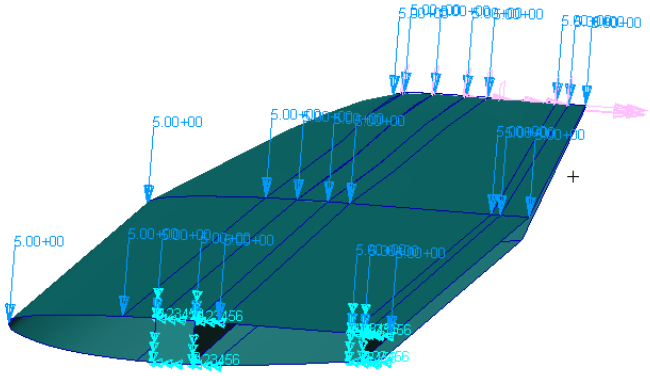


Fig. 54: Boundary conditions applied to the static structural model

5.2.4. Results

Results of this research were obtained after solving approximately 25 000 boundary value problems and consist of a set of 249 Pareto-optimal (non-dominated) solutions found during the course of optimization using the parameters described in chapter 3. Values of objective functions for the obtained solutions are shown in the figures in two ways. A 3D scatter plot with values of the f_1 function represented by colour, and the remaining fitness function as values on the axes is shown in the Fig. 55. Further examined solutions of interest are marked with a circle and named D1-D3 and ED. Scatter plot matrix of obtained solutions is shown in the Fig. 56. Along the diagonal are histogram plots of objective function values.

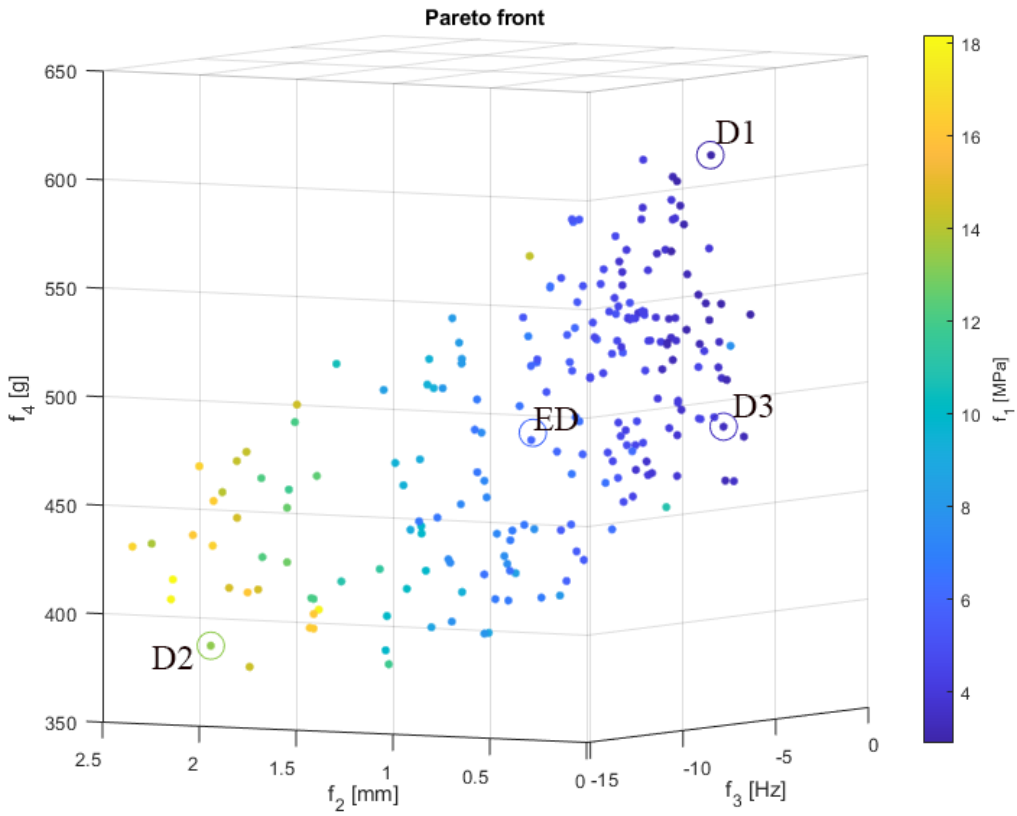


Fig. 55: 3D scatter plot of obtained solutions

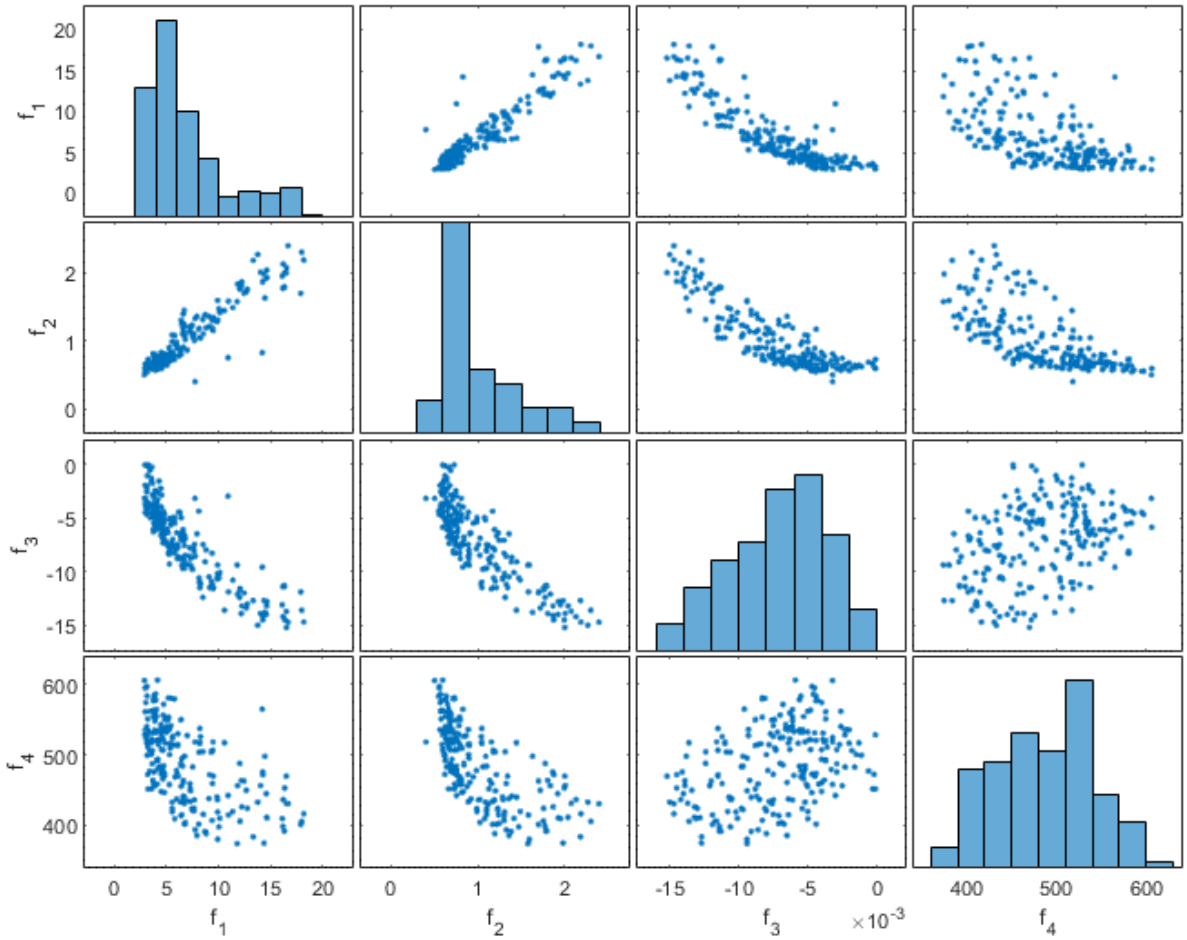


Fig. 56: Scatter plot matrix of obtained solutions.

Values of fitness functions of selected solutions are shown in the Tab. 2. Solution D1 represents the solution with the lowest values of f_1 and f_2 among the Pareto set. Solution D2 represents the solution with the lowest value of f_4 and solution D3 represents a compromise solution, with the value of f_4 similar to the existing design, but with improved values of f_1 and f_2. Additionally respective values of objectives of an existing design (ED) based on which the parametric numerical model was built are shown as a reference point in Tab. 20.

Tab. 20: Values of fitness functions and design variables of selected solutions

Design	ED	D1	D2	D3
f_1 [MPa]	6.06	2.9	13.32	3.42
f_2 [mm]	0.9	0.51	2.18	0.69
f_3 [Hz]	-8.7	-3.2	-12.7	-0.6
f_4 [g]	482.4	605.89	383.716	477.18
V1	10	-1.2583	3.4445	4.2395
V2	0.16	0.2137	0.3311	0.2465
V3	-5	0.2819	-6.8667	-5.7859
V4	0	4.3274	-2.2546	4.7214
V5	0	1.9522	0.1727	-2.3486
V6	1.2	1.9379	0.1291	1.9846
V7	0	3.4981	2.5754	1.987
V8	0.1	0.8806	0.0639	0.8092
V9	0	1.4691	2.3692	0.0192
V10	10	-0.7859	-4.7336	3.9142
V11	0.16	0.1726	0.3601	0.118
V12	0	5.6053	-6.8564	5.8716
V13	0	3.1475	1.4107	-3.3297
V14	0	4.768	-0.8138	1.344
V15	1.2	1.997	0.4045	0.526
V16	0	-3.9373	4.6358	3.7997
V17	0.1	0.9711	0.4932	0.2214
V18	0	-1.1374	-2.0907	-0.2444
V19	0	-2.5771	0.8685	-3.5369
V20	0	-6.0362	3.7904	-0.0829
V21	0	0.9502	-0.3279	-4.8359
V22	0	-3.2653	-3.4725	4.6151
V23	0	-3.4795	-3.8835	0.0078
V24	0	-0.0928	0.3497	0.4858

Geometric configurations of selected designs, post-processed to show the distribution of equivalent stresses are shown in the Fig. 57. Colour spectrum to stress value relation is not uniform for the displayed solutions. Colours represent range between the lowest and highest values of equivalent stresses for every single solution.

In general, it can be noticed that f_1 and f_2 appear to be consistent with each other and therefore their nature is not conflicting. Both of these objectives are however contradictory to objective f_3 and in some areas of solution space to f_4.

Tab. 21: Min, median and max values of objectives.

Objective	Min	Median	Max
f_1	2.8946	5.5001	18.1866
f_2	0.4053	0.80785	2.3929
f_3	-15.2	-6.7	-0.1
f_4	373.893	487.734	606.363

In Tab. 21 it can be observed that median values of objective function resemble values of objective functions obtained for an existing design. It shows that search space was explored towards min and max values in a similar magnitude. In the ideal scenario, solution space should be explored only in the direction for which values of objective problems are decreasing assuming minimization problem. This situation is however impossible due to contradictory nature of objectives.

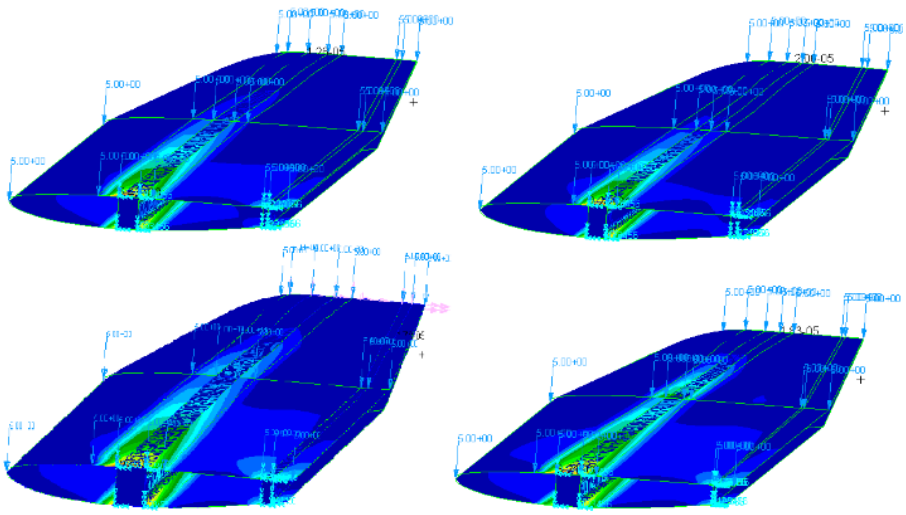


Fig. 57: Geometric configuration and equivalent stress distribution of selected solutions. Existing design in the top left, D1 – bottom left, D2 – Top right, D3 – bottom right.

5.2.5. Final remarks

Multiobjective optimization of airfoil systems was presented. Optimization of the parameters for the airfoil was solved for four different proposed functionals. More functionals related to particular needs asked of airfoil systems can be formulated. New designs better than the existing design were found for each optimized objective. Stress and displacement values are acceptable even for the lightest structure found. No sweet spots were found in the solution space, which was expected due to the contradictory nature of optimized objectives. Application of presented method of optimization of an airfoil system can significantly enhance the process of its design with respect to the many requirements asked of such systems. Further choice of a design among a set of non-dominated designs should be taken according to additionally established preferences.

5.3. Optimization of electrothermal microactuators

Electrothermal microactuators are components of machines and devices in which motion is induced as a result of thermal contraction or expansion of conductor materials. To achieve temperature change, resistive heating is generated in actuator due to electrical current flow utilising Joule's effect. The material from which they are made must display a high electrical resistance value and for that purpose polycrystalline silicone is used. Thermal actuators can be considered as microelectromechanical systems (MEMS) used to translate electrical signal into force or displacement and in this context are an object of interest of nanoscale and microscale engineering [149]. Electrothermal microactuators have found application in areas such as automotive, telecommunication, aviation, and medical industries, for example in filters, microgrippers, modulators and switches. Electrothermal actuators offer numerous advantages over other types of actuators (e.g., electrostatic, magnetostatic, and piezoelectric):

- generate large forces or displacements per unit of volume,
- work with low actuation voltage,
- exhibit linear stress-strain relationship even in high temperature,
- have high accuracy and long lifecycle,
- require little maintenance,
- are relatively cheap and easy to manufacture.

On the other hand, among disadvantages of microthermal actuators there are: slow response time and relatively high energy consumption.

There are two main types of electrothermal microactuators used in MEMS: U-beam and V-beam. In the early 1990s U-beam type actuators were first developed and demonstrated and it was until the late 1990s when V-beam type actuators were introduced and quickly rose in popularity. U-beam actuators are made of two beams (arms), each with a different cross-section. When electric current passes through actuator, beams generate a different amount of heat due to the difference in beams' cross section sizes which results in the presence of a hot arm and a cold arm. The expansion of the hot arm causes a rotation of both arms therefore generation displacement of the tip of the actuator (Fig. 58).

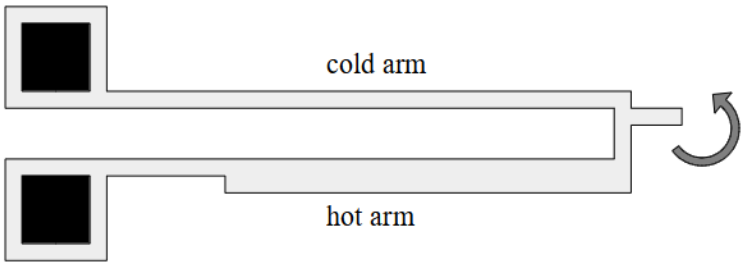


Fig. 58: U-type electrothermal microactuator

In case of V-type actuators, the arms have the same cross sections and expand uniformly, but the moving central shaft of the actuator, is connected to arms from both sides and a pre-bending angle is applied in the manufacturing process. Arms are connected on one side to fixed anchors and on the other side to a moving shaft. As the electric current passes through, the arms are expanded and the bending angle between arms and the shaft is increased therefore moving the central shaft. V-type actuator can aggregate force generated by multiple beams [150]

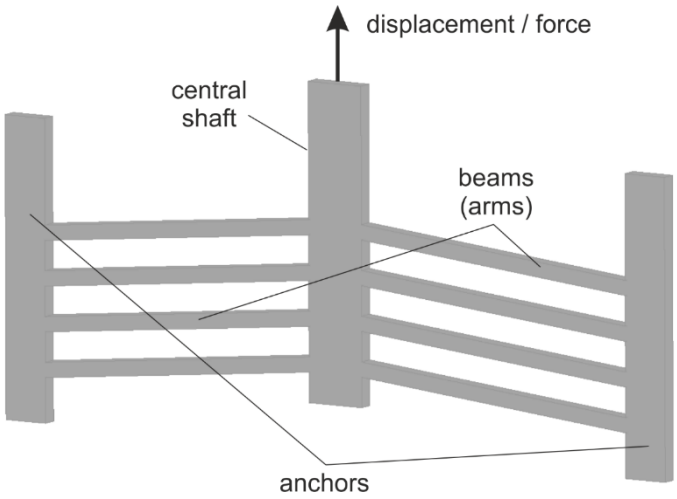


Fig. 59: V-type electrothermal microactuator

resulting in a larger total force compared to a U-type actuator. Chevron type, bent-beam and symmetric thermal actuator are other names of V-type actuator which is considered in this study. In the Fig. 59 a chevron type actuator consisting of four pairs of arms is presented on the left and deformed system including distribution of equivalent stresses is shown on the right.

During the design of such structures many, often contradictory, requirements asked of actuators must be taken into consideration. In the optimization process these requirements are expressed in the form of objective functions and constraints. These requirements for actuators can concern quantities like: generated displacement and force, electrical or heat loss, maximum equivalent stress, dynamic characteristics, and risk of buckling. All the aforementioned requirements depend on the geometry of the actuators. For some actuators, with simple geometry, analytical solutions related to mechanical, electrical, and thermal quantities can be derived, however in case of a complex geometries (multiple arms, shape other than straight) these values must be obtained by means of numerical simulations. As the problem involves multiple physical fields a coupled electro-thermo-mechanical analysis is necessary. Objective functions based on coupled-fields computations tend to exhibit a strongly multimodal nature, meaning they have multiple local optima. Hard computing methods are expected to be less effective in dealing with such problems and instead soft computing methods are advised to be used to perform an efficient optimization process [151]. Many researchers have contributed papers devoted to shape optimization of thermal actuators: in [152]–[154] laterally-driven, electro-thermal microactuators are concerned in the context of structural and geometrical optimization, in [155] optimal design of actuators with non-uniform lengths, cross sections and pre-bending angles of beams was considered, variant analysis in the optimal design of electrothermal actuators was investigated in [156], [157] and Particle Swarm Optimization in [158]. A lot of researchers investigated topology optimization of electrothermal actuators, resulting in complex geometrical designs: output displacement optimization was considered in [159]–[161], level-set method in topology optimization was investigated in [162], [163] and rotary thermal actuators optimization was studied in [164]. Results of topology optimization can provide valuable ideas for the designers of actuators, although applicability of these results is limited by complex and thus hard to manufacture shapes of resulting designs. Preceding papers dealt with single-objective optimization of actuators and multiobjective problems have attracted less attention of researchers: a novel biogeography-inspired multiobjective algorithm

and NSGA-II were used to solve two-objective optimization of U-type actuator in [165] and another novel optimization method, combining genetic algorithm, particle swarm optimization and improved gradient descent algorithm investigated a similar problem in [166]. NSGA-II and multiobjective evolutionary algorithm were used to solve a two-objective optimization problem of U-type electrothermal and piezoelectric actuators in [167]. Immune Game Theory Multiobjective Algorithm was used to optimize the shape of V-type electrothermal microactuator with respect to six objectives in [168] and the example presented further in this study can be considered an extension of this work, as it concerns the same optimization problem resolved with a different optimization tool.

5.3.1. Aims and assumptions

This example concerns a search of an optimal design among a set of Pareto-optimal designs of V-type electro-thermal microactuator. Such system works under electric current, generating Joule heat and due to thermal expansion displacement and force are generated in the central shaft. The optimization process is performed for six criteria at once and thus a multiobjective optimization approach is used. Main assumptions of the research include:

- Weak coupling between thermal, electrical, and mechanical fields.
- Coupling is performed by subsequently transferring loads between first electrical, then thermal and lastly mechanical analyses.
- Linear buckling is assumed.
- Friction is neglected.
- Contact problem is modelled as rigid – deformable and frictionless.
- Linear-elastic, isotropic material model is assumed for actuator.
- Rigid surfaced is placed in an offset position within an arbitrary distance from actuator to analyse contact and determine force generated by the central shaft.
- Solutions not generating contact (not enough value of vertical displacement, central shaft doesn't reach the contact surface) are discarded.
- No penetration is allowed.
- Parametric numerical model of an actuator is modelled utilising NURBS curves.

5.3.2. Formulation of the problem

Objectives in the optimization problem are devoted to minimization of total volume, minimization of maximal equivalent stresses, maximization of vertical displacement of central shaft, minimization of total heat and maximization of buckling factor and maximization of total force generated in the central shaft – these are the examples of requirements asked of electrothermal microactuator systems, although other criteria based on specific needs can be formulated. Values of functionals used as objectives are computed using FEM simulations. Material properties are constant during the optimization run and geometry of the model, described by design variables is changed to fit the declared needs, formally described by functionals (46) – (51).

Minimization of the volume of actuator:

$$\min_{x} \; f_1 = \int_{\Omega} d\Omega \tag{46}$$

Minimization of the maximal equivalent stress in the actuator (assuming Huber-Mises-Hencky hypothesis):

$$\min_{x} f_2 = \max(\sigma_{eq}) \tag{47}$$

Maximization of the vertical displacement of the tip of central shaft:

$$\max_{x} f_3 = u_i \tag{48}$$

Minimization of the total heat generated in the actuator:

$$\min_{x} f_4 = q_{total} \tag{49}$$

Maximization of the buckling factor:

$$\max_{x} f_5 = f_{buck}(P_v) \tag{50}$$

Maximization of the force generated by the central shaft:

$$\max_{x} f_6 = F_c(d_g) \tag{51}$$

P_v is a vertical external force applied to the top of the central shaft. Force F_c is calculated on the basis of a contact force generated between displaced central shaft and a rigid surface placed in an offset position within an arbitrary distance d_g from the central shaft before deformation of the actuator.

Values of aforementioned functionals are calculated numerically by means of FEM in MSC Patran/Nastran and Marc/Mentat software based on a parametric model. Functional f_1 is calculated on the basis of geometry of the model in preprocessor Mentat and does not require solving a boundary-value problem. Functionals f_2, f_3 and f_6 are calculated by running an electro-thermo-mechanical analysis. Functional f_4 is calculated by running an electro-thermal analysis. Functional f_5 is calculated by running a buckling analysis. Problems of maximization are transformed to minimization problems by multiplying a fitness function value by -1.

5.3.3. Numerical model

In electrothermal microactuators displacement and force are indirectly generated by the electrical current I passing through conductor material, generating resistive heat according to the Joule's Law:

$$Q = RI^2t \tag{52}$$

where Q is Joule heat generated, R is electrical resistivity and t is time.

Displacement of the central shaft is caused by the thermal strain of the arms due to resistive heat generated. Microactuators are usually subjected to the voltage of a magnitude of a few volts, which is enough to cause temperature increase to the values over 1200K. Despite high temperatures, system maintains its linear characteristic with respect to thermal, electric, and mechanical fields. These phenomena are described by partial differential equation of electrostatics (53), heat conduction (54), and thermoelasticity (55).

$$\phi_{,ii} - \frac{\rho}{\varepsilon_0} = 0 \tag{53}$$

$$kT_{,ii} + Q = 0 \tag{54}$$

$$\mu u_{i,jj} + (\mu + \lambda)u_{i,ji} - (3\lambda + 2\mu)\alpha_t T_{,i} = 0 \tag{55}$$

where ϕ is electric potential, T is temperature, u represents the displacement values, ρ is charge flux density, ε_0 is vacuum permittivity (electric constant), k is thermal conductivity, Q is internal heat source, α_t is the linear expansion coefficient and μ and λ are the Lamé constants, which can be expressed in the following way:

$$\mu = G = \frac{E}{2(1+\nu)}, \lambda = \frac{E\nu}{(1-2\nu)(1+\nu)} \tag{56}$$

where G is shear modulus, E is Young's modulus and ν is Poisson's ratio.

Partial differential equations (53)-(55) must be supplemented by boundary conditions.

- Boundary conditions for electrostatic problem:

$$\begin{gathered} x \in \Gamma_\phi: \phi(x) = \phi^0(x) \\ x \in \Gamma_\rho: \rho(x) = \rho^0(x) \end{gathered} \tag{57}$$

where ϕ^0 and ρ^0 are known electric potential and electric charge flux density on the respective parts of the boundary Γ_ϕ and Γ_ρ.

- Boundary conditions for heat conduction problem:

$$\begin{gathered} x \in \Gamma_T: T(x) = T^0(x) \\ x \in \Gamma_q: q(x) = q^0(x) \\ x \in \Gamma_c: q(x) = \alpha_c(T(x) - T^\infty(x)) \end{gathered} \tag{58}$$

where T^0 and q^0 are known temperature and heat flux on the respective parts of the boundary Γ_T and Γ_q. Last boundary condition is a convection condition, where α_c stands for heat convection coefficient and $T^\infty$is ambient temperature around part of the boundary Γ_c.

- Boundary conditions for elasticity problem:

$$\begin{gathered} x \in \Gamma_u: u(x) = u^0(x) \\ x \in \Gamma_p: p(x) = p(x) \end{gathered} \tag{59}$$

where u^0 and p^0 are known displacements and mechanical loads on the respective parts of the boundary Γ_u and Γ_p.

Boundary of the body is defined by parts of the boundary where electric, thermal, and mechanical boundary conditions are applied. To solve the electro-thermo-mechanical boundary value problem, FEM is used. Partial differential equations governing the problem are discretized and transformed into a system of algebraic equations. After taking into consideration boundary conditions, a resulting system of equations in the matrix form is obtained:

$$\mathbf{K_E V = I} \tag{60}$$

$$\mathbf{K_T T = Q + Q_E} \tag{61}$$

$$\mathbf{K_M U = F + F_T} \tag{62}$$

where $\mathbf{K_E}$, $\mathbf{K_T}$ and $\mathbf{K_M}$ are global matrices of: electrical conductivity, thermal conductivity, and stiffness respectively, assembled by aggregation of local matrices according to FEM discretization scheme. **V**, **I**, **T**, **Q**, **U** and **F** are global vectors of: voltage, electric current, temperatures, heat fluxes, displacement, and mechanical loads respectively. As weakly coupling between physical fields is assumed, electric, thermal, and mechanical analyses are solved in a successive manner. Coupling is realised by transferring of loads into subsequent analyses. In case of full electro-thermo-mechanical analysis, in the first step equation (60) is solved and a vector of heat flux $\mathbf{Q_E}$ caused by Joule effect is obtained, which is then included in the equation (61), whereas after solving equation (61) a vector of loads $\mathbf{F_T}$ due to thermal strain is obtained, which is then included in the equation (62).

In case of buckling analysis, the FEM formulation of a problem takes the following form:

$$[\mathbf{K_M} + \lambda_i \Delta \mathbf{K_G}]\mathbf{v_i} = \mathbf{0} \tag{63}$$

Assuming linear buckling, equation (63) describes the eigenvalue problem, solved by the Lanczos method. Matrix $\Delta\mathbf{K_G}$ is a function of the load increment ΔP, λ_i is the eigenvalue and eigenvector $\mathbf{v_i}$ is used to calculate post-buckling deformation mode.

For the contact analysis, required to determine value of the actuation force, an additional rigid surface is placed in an offset position within an arbitrarily chosen distance from the location of the tip of the central shaft before deformation of the system. After the deformation,

contact between the tip and the surface is established and contact force is measured. As the movement of the shaft is in the direction normal to the surface, friction between these elements is neglected. Contact is modelled as rigid–deformable and frictionless. Additional constraint responsible for maintaining lack of penetration is implemented.

The parametric numerical model of the actuator is solved by means of FEM to obtain values of optimization functionals. It is however possible to obtain related values in an analytical way for the actuator with straight arms. The analytical and numerical solutions for elasticity problem for such actuators were compared to provide verification of the numerical model. Considering constant thickness of the actuator and its relatively small magnitude compared to other dimensions as well as constant boundary conditions, two-dimensional analytical solution can be used. Analytical solution for displacement u_y of V-type actuator (Fig. 60) is provided in [150], [153] as:

$$u_y = \frac{\alpha_t \Delta T E A \sin\varphi}{\frac{EA}{L}\sin^2\varphi + \frac{12EI_z}{L^3}\cos^2\varphi} + \frac{-F_i}{\frac{EA}{L}\sin^2\varphi + \frac{12EI_z}{L^3}\cos^2\varphi} \tag{64}$$

where α_t is a thermal expansion coefficient, ΔT is average increase of temperature due to the resistive heat, E is Young's modulus, A is cross section area of an arm, φ is pre-bending angle, L is the distance between anchor and center of the actuator in the x direction, I_z is moment of inertia of cross section of an arm and F_i is an external force applied to the central shaft in the direction against u_y.

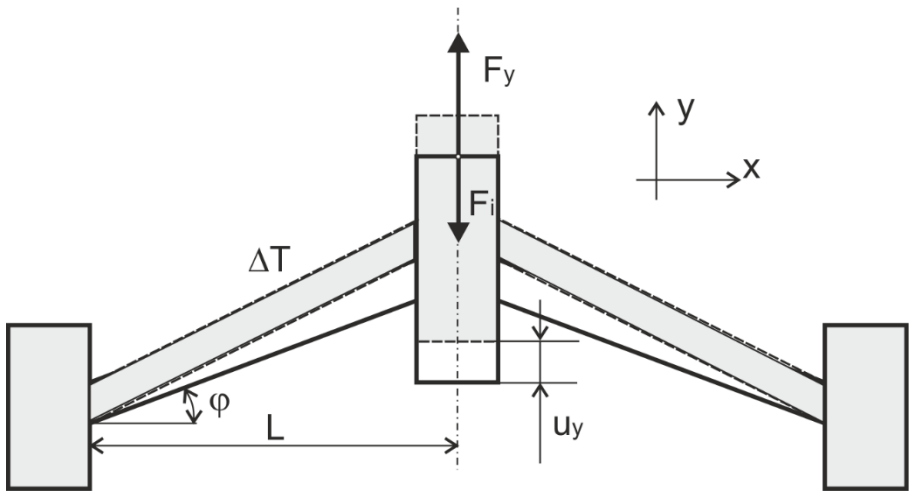


Fig. 60: V-type actuator in the analytical solution

The comparison between FEM numerical model and analytical solution (64) was performed for the following parameters:

- average temperature rise $\Delta T = 450$ K,
- Young's modulus $E = 210$ GPa,
- length of beams $L = 220$ µm,
- square cross section area $A = 25$ µm^2.

Numerical model was prepared in MSC Marc/Mentat utilising plane stress model and consisting of quadrilateral four node elements. Mesh density was established after preliminary analyses resulting in a model with 6-8 elements per width of a beam to ensure valid results for bending arms and therefore the entire actuator model consists of around 10 000 elements and 11 000 nodes (Fig. 61).

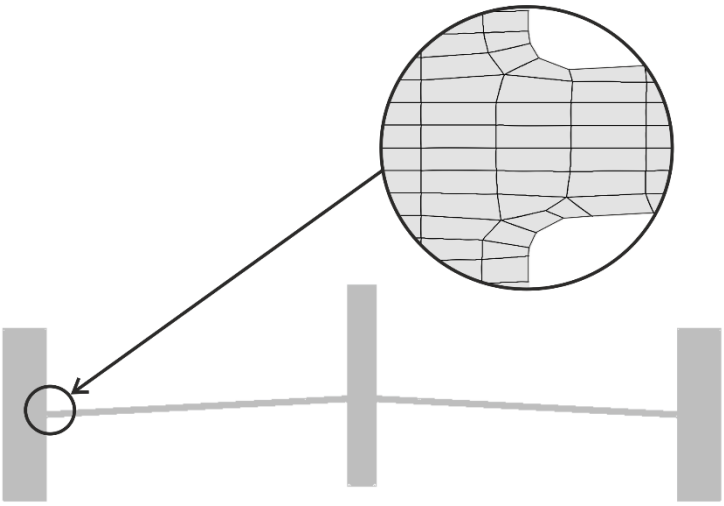


Fig. 61: V-type actuator in the FEM numerical solution

The analysis was performed for a range of pre-bending angles φ between $\varphi = 1°$ and $\varphi = 5°$. Obtained results (Fig. 62) prove a good compliance between numerical model and analytical solution. The relative error of numerical solution was no more than 2.3% observed at the location of the highest displacement value.

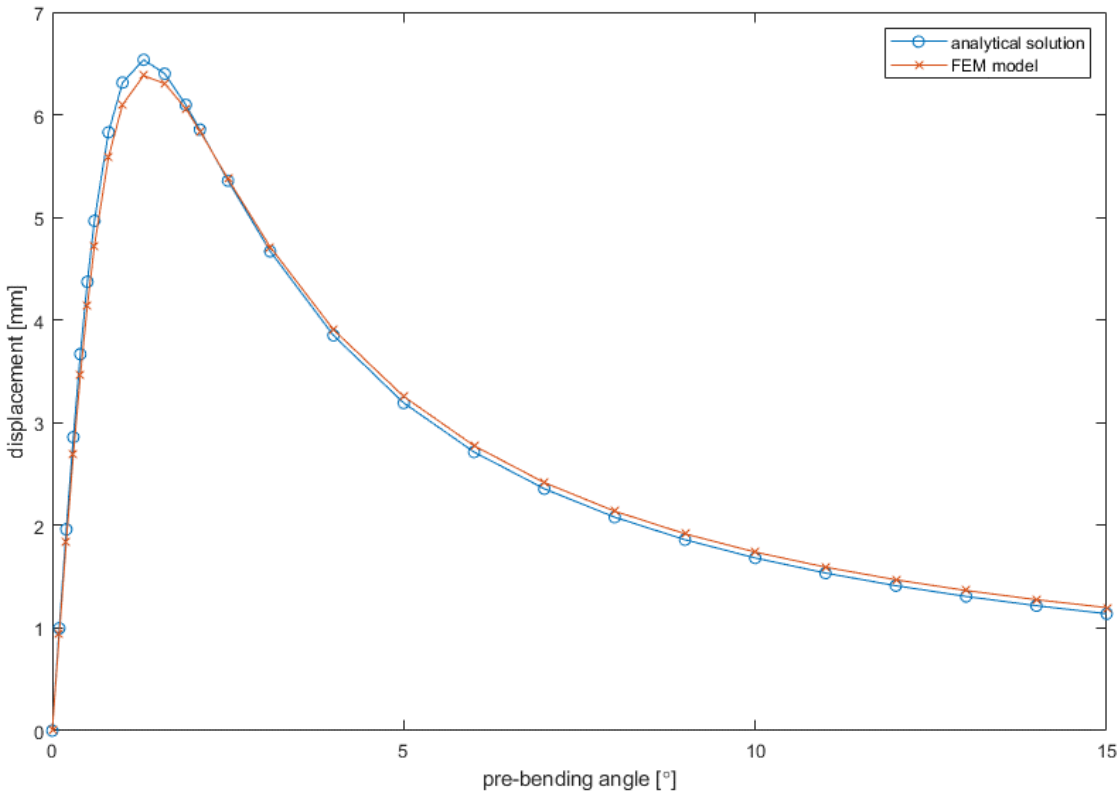


Fig. 62: Comparison between FEM numerical solution and analytical solution

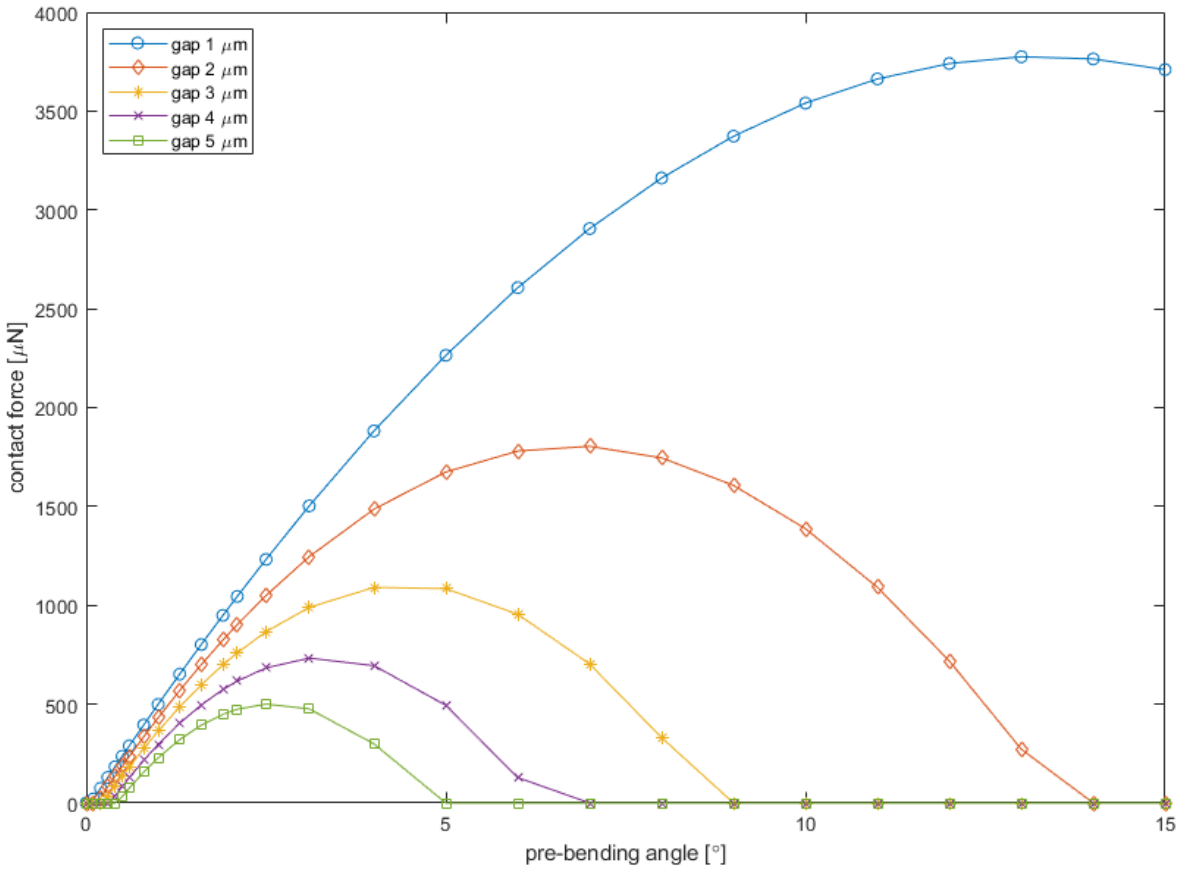


Fig. 63: Relationship between pre-bending angle and contact force

Analytical solution (64) can be transformed to investigate the value of force generated by the actuator although it is proven to underrepresent interaction between force and displacement in case of low displacements or high value of forces [150]. For the practical purpose it is best to investigate value of actuation force numerically by introducing a rigid contact surface to the model and observe value of contact force after deformation. Said surface must be placed within a distance from the tip of the moving part of the actuator in the direction of working movement. Value of this distance, further also referred to as a "gap" is crucial in the ability to measure contact force. Fig. 63 presents relationship between contact force measured on the rigid surface and value of pre-bending angle for different variants of gap value. It can be noticed that with the increase of the gap, the maximum measured contact force decreases and that for low value of pre-bending angle it is not possible to observe high contact forces nevertheless the relationship is not linear and increasing pre-bending angle further at some point does not result in the increase of measured contact force and a clear maximum can be observed.

Numerical investigation of the buckling effect can be compared to analytical solution to verify the numerical model. Buckling appear when an internal compression force in axially loaded arms is greater than the value of critical buckling force. The axial load in arms is equal to the reaction force in the anchor of an actuator and can be calculated as [169]:

$$R = \alpha \Delta T E A \frac{\cos^2\varphi}{\frac{AL^2}{12I_z}\sin^2\varphi + \cos^2\varphi} \tag{65}$$

and critical force is:

$$F_{cr} = \frac{\pi E I_{min}}{\beta L^2} \tag{66}$$

where I_{min} is a minor moment of inertia of cross section of beam and β is a buckling length coefficient. For the purpose of verification, a simple actuator with the same geometry was analysed analytically and numerically, for which the value of critical force according to (66) was calculated and took value of $F_{cr} = 419.5$ µm. For the numerical investigation, the anchors of the actuator were fixed in all degrees of freedom and the central shaft was allowed to move along the direction of axis x only. Previously calculated critical force F_{cr} was applied to the moving shaft and the Critical Loading Factor (CLF) was obtained numerically by means of FEM. CLF is a ratio between critical and applied load and should take the value of 1.0 to match the analytical solution when applied load is equal to the critical force. Obtained value of CLF = 1.058 indicates a good compliance of analytical and numerical results for the analysed case. Moreover, values of compressive stress obtained from analytical and numerical solution deliver similar results, however it is worth noting that the analytical solution takes into consideration only the uniaxial stress, while numerical analysis additionally incorporates concentration of stresses.

Presented verification of numerical model compared to analytical solutions for an actuator with straight arms can be considered successful as results obtained numerically seem to come to a close agreement with analytical solutions. On the basis of analysed geometry of the actuator, a new parametric model was established. Parametric model was designed in a way to provide a flexible shape whose geometry is controlled by design variables in the optimization process. The main requirement for the parametric model was to cover a wide range of candidate

solutions, including solutions for which analytical formulas are unknown. To achieve this goal, a model described by seven design variables, presented in the Fig. 64 was constructed. To ensure axial movement of the shaft symmetry along the vertical axis was assumed. Shape of the beam was based on a NURBS curve constituted of four control points. Uniform distribution, cubic interpolation of the knot vector and constant value of 1.0 of points' weights is assumed. The four control points create a NURBS control polygon, whose shape can be manipulated by design variables and therefore change the shape of a beam. Symmetry of the beam is imposed along the centerline. First four design variables define the distance of control points from the centreline. The remaining three design variable define value of pre-bending angle φ and fillet radii R_1 and R_2 between the beam and anchor and beam and shaft respectively.

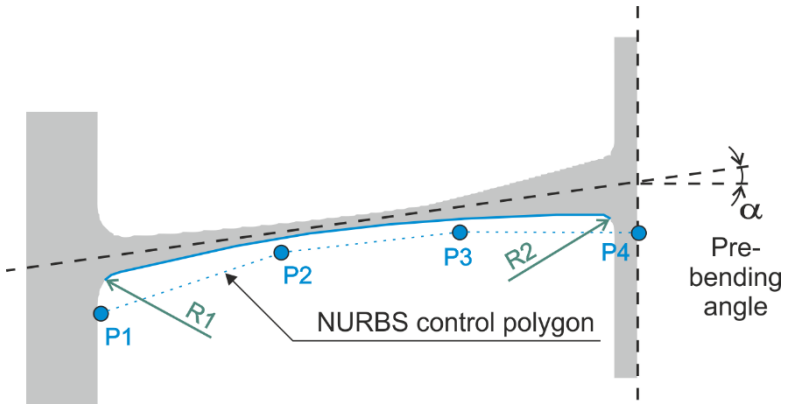


Fig. 64: Geometry and parametrization of the model

Design variables and limits imposed on then are summarised in Tab. 22.

Tab. 22: Design variables

Symbol	Design variable	Lower limit	Upper limit
P1	position of control point P1	1 µm	20 µm
P2	position of control point P2	1 µm	20 µm
P3	position of control point P3	1 µm	20 µm
P4	position of control point P4	1 µm	20 µm
$\boldsymbol{\varphi}$	pre-bending angle	0.001°	12°
R1	fillet radius between beam and anchor	1 µm	20 µm
R2	fillet radius between beam and shaft	1 µm	20 µm

Actuator was modelled in a plane stress state with the following material properties of a polycrystalline silicon:

- Young's modulus $E = 158 \times 10^3$ MPa,
- Poisson's ratio $\nu = 0.23$,
- thermal conductivity $k = 140 \times 10^8 \frac{\mathrm{W}}{\mu\mathrm{mK}}$,
- resistivity $R = 3.3 \times 10^{-11} T\ \Omega\mu\mathrm{m}$,
- linear expansion coefficient $\alpha_t = 3 \times 10^{-6}\ \mathrm{K}^{-1}$.

Thickness of the actuator in the z-direction is 10 μm, length of an undeformed beam is 220 μm, the distance between the tip of undeformed central shaft and rigid contact surface (gap) was assumed as $d_g = 2$ μm and μMKS system of units is used.

Following boundary conditions were imposed on the system:

- anchors fixed in all degrees of freedom,
- temperature $T = 300$ K,
- electric potential difference between anchors 5 V,
- vertical load $P_v = 500$ μN for the buckling analysis.

5.3.4. Results

Results of this research were obtained after three independent runs of the DEGT algorithm using following parameters:

- generations $DE_{iter} = 5$,
- population size $DE_{pops} = 6$,
- crossover rate $CR = 0.5$,
- scaling factor $F = 0.7$.

For the three runs, different value of objective function calls as termination condition were used: 50 000, 100 000 and 127 000, giving a total number of 277 000 objective function calls. Algorithm produced respectively: 828, 1440 and 1061 Pareto-optimal (non-dominated) solutions. The results from three runs were then aggregated to form a set of 2891 solutions. After merging three sets, it was necessary to discard solutions dominated by members of a combined set of solutions hence the number of non-dominated solutions in a combined set is lower than a sum of its components members.

Values of objective functions for the obtained solutions are shown in the figures in two ways. Scatter plot matrix of obtained solutions is shown in the Fig. 65. Along the diagonal are histogram plots of objective function values. Additionally, solutions are presented in the form of Kohonen's Self-Organizing Maps, described in section 2.5. Selected solutions are shown in Tab. X including values of objective functions and design variables. Geometry of selected solutions is presented in the Fig. 66.

On the basis of obtained sets of non-dominated solutions and presented graphs some general information on relationships between objective functions in the problem can be drawn. Objectives f_1 and f_4 have a similar nature. Colours representing objective values on the SOM maps for them are in a close agreement and related plot on the scatter plot matrix is approximately proportional. On the other hand, objective f_5 displays conflicting nature towards these objectives.

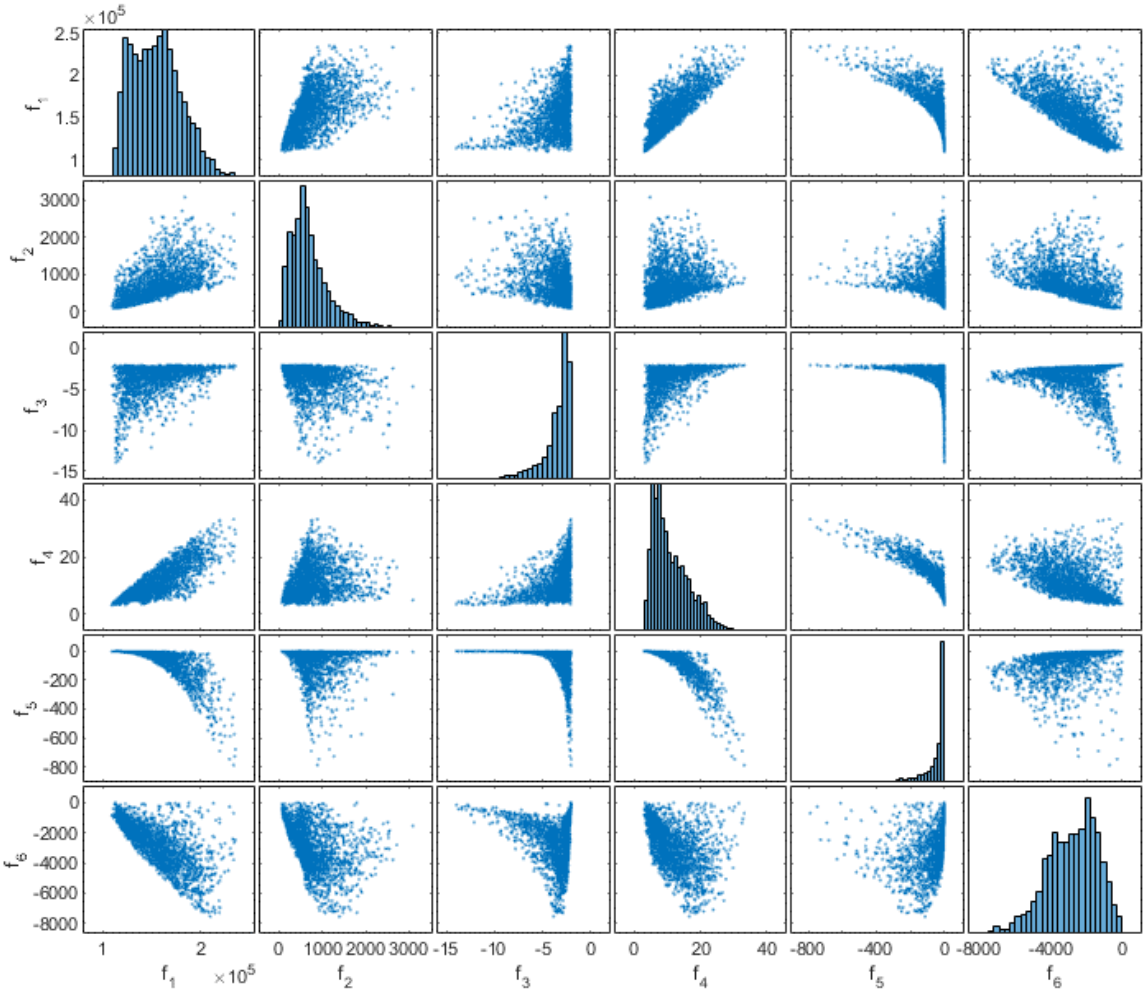


Fig. 65: Scatter plot matrix of solutions for actuator optimization

Tab. 23: Values of fitness functions and design variables of selected solutions

Design	D1	D2	D3	D4	D5	D6
f_1 [µm³]	**109521,3**	110828,7	113578,7	123329,3	233154,4	209377,3
f_2 [MPa]	137,156	**66,729**	907,749	800,637	888,74	1397,96
f_3 [µm]	-4,5103	-2,2723	**-13,9523**	-9,229	-2,0415	-3,8537
f_4 [µW]	3,1261	3,914	3,4224	**3,0979**	33,2006	16,6609
f_5 [−]	-0,3673	-0,9527	-0,0668	-0,4854	**-789**	-112,7
f_6 [µN]	-841,451	-406,973	-145,786	-801,946	-703,381	**-7544,8**
P1 [µm]	1,0895	1,307	1,0123	1,0516	2,3209	1,0336
P2 [µm]	1,0586	1,4216	1,109	1,132	19,498	19,3231
P3 [µm]	1,1676	1,0225	1,7186	1,0048	18,2076	19,1267
P4 [µm]	1,0644	1,3784	1,2592	9,1116	13,8266	5,3813
φ [°]	5,7963	11,785	0,521	2,5687	11,3195	6,8521
R1 [µm]	1,7875	1,2453	7,6393	1,5735	19,4211	1,0779
R2 [µm]	5,3327	3,9615	15,5259	13,5593	19,9716	3,6833

Selected solutions are chosen so as to exhibit extreme values of consecutive objective functions. Solution D1 exhibits the lowest value of f_1, solution D2 – lowest value of f_2 and further selected solutions likewise. Extreme values are shown using bold font in Tab. 23.

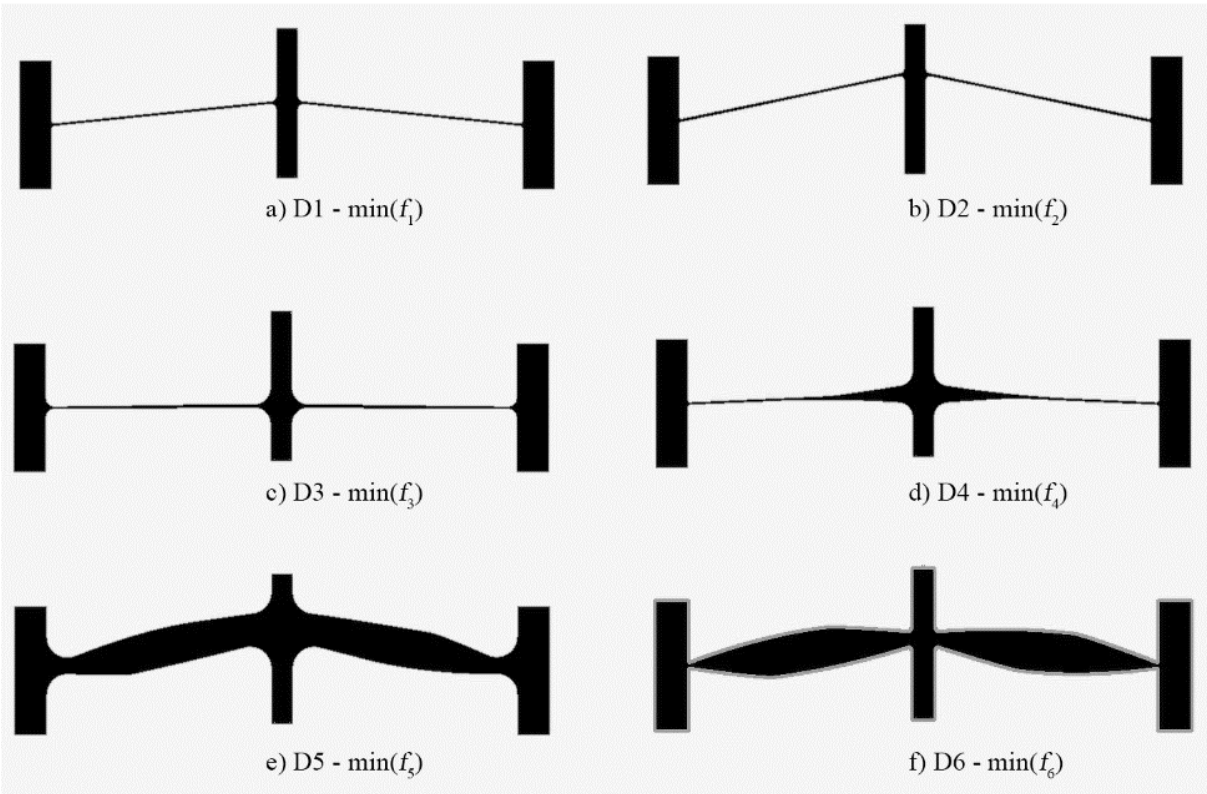


Fig. 66: Geometry of selected solutions

Except for solutions displaying extreme values of functionals it is crucial to be able to choose a compromise solution among a non-dominated set according to established preferences. These preferences can be expressed in the form of maximum acceptable value of a group of optimization functions. The decision making process utilising SOM, considering threshold values of objectives f_2, f_4 and f_6 and final decision with respect to lowest value of f_1 was presented in the Fig. 67.

In the first step a threshold value of $f_2 = 900$ MPa is assumed and only solutions satisfying this constraint are accepted. Related area is marked on the SOM using red colour and solutions with higher level of equivalent stress are disregarded in the further investigation. The decision making process then proceeds to check next threshold for objective f_4 and the area of interest is further reduced, only the intersection of areas of acceptable solution is taken into further steps. Threshold for f_6 is applied in the similar manner and in the last step only a considerably narrowed down area of interest remains. Among this area the best solution with regards to objective f_1 is sought by analysing values of codebook vectors of related SOM units. Ultimately a compromise solution displaying following values of objective functions and design variables is found:

$$\mathbf{f} = [164937.5\ \mu\text{m}^3, 614.1\text{MPa}, -3.16\mu\text{m}, 14.89\mu\text{W}, -70.88, -5023.49\mu\text{N}]$$

and $\mathbf{x} = [2.47\mu\text{m}, 6.58\mu\text{m}, 14.09\mu\text{m}, 1.95\mu\text{m}, 8.79°, 9.29\mu\text{m}, 17.15\mu\text{m}]$.

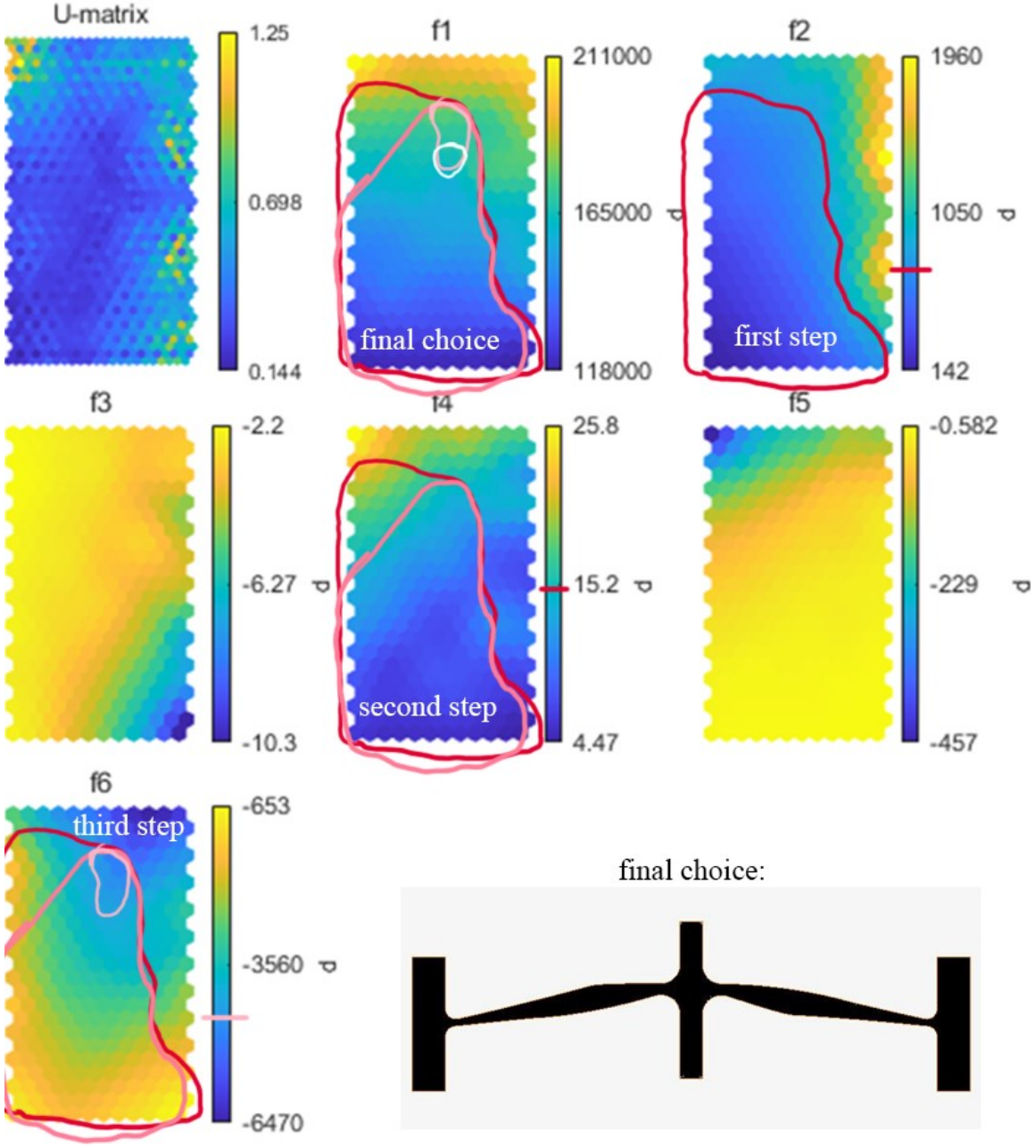


Fig. 67: SOM trained on solutions for actuator optimization and post-optimization decision making process

5.3.5. Final remarks

Multiobjective optimization of electrothermal microactuators was presented. Optimization of the parameters for the actuator was solved for six different proposed functionals. More functionals related to particular needs asked of actuators can be formulated. A large set of non-dominated solutions was found. Proposed methods of visualisation help to draw conclusions on natures of the objectives. Among found solutions, some designs related to extreme value of optimization functions were highlighted, compared in terms of design variables and geometry of these solutions was shown. Moreover, the process of decision making according to established preferences enhanced by SOM was presented and applied to choose a single compromise solution. Application of presented methods can improve the design of multiobjective actuators.

5.4. **Optimization of multiscale porous material**

Microscale analyses are an important tool in the design process of advanced materials, in which geometry and properties of microstructure are taken into consideration. Microstructure of materials can strongly influence physical properties such as strength, heat conductivity, electrical conductivity, density etc. Microscale characteristics play a crucial role in behaviour of materials such as: porous materials or materials reinforced with fibres or particles. Microstructure composed of more than one material with dissimilar physical properties can provide properties which wouldn't be available to be obtained using a homogenous material. Numerical simulation of such structures in one scale is a very computationally demanding task as it would require extremely fine mesh and complex models in order to adequately reflect the differences in properties of the microstructure sections. For this reason, multiscale modelling can be applied to reduce the complexity of models concerning two or more scales. Many phenomena happening on a different magnitude of scale and interacting with each other can be investigated in the multiscale model. In the macroscale: mechanical and thermal boundary conditions (loads, supports etc.). In the meso- and microscale: inclusions, cavities and even defects in the crystal lattice. Multiscale modelling can even be used in problems concerning effects happening on a molecular level [170], [171]. In order to analyse the behaviour of a system in the macroscale it is necessary to establish information on effective material properties. In the process of homogenization, a heterogenous material is transformed to a homogenous one described by effective material properties obtained through microscale analyses. In case of a simple microstructure geometry, analytical solutions can be used although for more complex shapes it is necessary to employ numerical methods. Boundary element methods was used to perform numerical homogenization of porous structures in [172]. Finite element method, however, is the most popular tool used for numerical homogenization due to its universal applicability and availability in the CAE software [170], [173], [174]. Multiobjective optimization problems concerning multiscale materials are remarkably computationally demanding tasks. During the optimization process in order to search the solution space for desirable designs it is necessary to obtain values of optimization functions multiple times. In case of multiscale problem, each analysed candidate solution additionally requires multiple analyses in the microscale to perform numerical homogenization. For this reason, an efficient multiobjective optimization tool is crucial to govern the optimization process in such problem to achieve satisfying results in limited time.

5.4.1. Aims and assumptions

Aim of this study is to find an optimal design of a microstructure of a porous system under thermal and mechanical loads with respect to the macroscopic properties. The optimization process is performed for four criteria at once and thus a multiobjective optimization approach is used. Main assumptions of the research include:

- Two-scale thermoelastic model is considered.
- Analysed system in the macroscale consists of an aluminium block under thermal and mechanical loads, fixed on one side.
- Geometry of the model in macroscale is not subject to optimization and does not change during the course of optimization.
- Linear-elastic, isotropic material model is assumed.
- Microstructure exhibits local periodicity.
- Representative volume element (RVE) approach coupled with finite element method is used for numerical homogenization.
- Four optimization functions are formulated on the basis of mechanical and thermal quantities observed on the micro- and macroscopic level.
- Parametric numerical model of microstructure is modelled utilising closed B-spline curves to describe the geometry of the pore.

5.4.2. Formulation of the problem

Objectives in the optimization problem are devoted to minimization of maximal displacements, minimization of maximal equivalent stress, maximization of the heat flux and maximization of porosity, which is related to the total mass and cost of the structure. These objectives are the examples of conditions required of porous mechanical systems, although other criteria based on specific needs can be formulated. Values of functionals f_1 and f_2 used as objectives are computed using FEM simulations in macroscale, supplemented by FEM simulations in microscale, required in the process of numerical homogenization. Objectives f_3 and f_4 are obtained in the microscale. Geometry of the microstructure, described by design variables is changed to fit the declared needs, formally described by functionals (67) – (70):

Minimization of the maximal translational displacement of a system under given load

$$\min_x f_1 = \max(u) \tag{67}$$

Minimization of the maximal value of the equivalent (von Mises) stress of a system under given load:

$$\min_x f_2 = \max(\sigma_{eq}) \tag{68}$$

Maximization of thermal conductivity in the direction along the axis of the bar:

$$\max_x f_3 = k_{11} \tag{69}$$

Maximization of porosity of the microstructure:

$$\max_x f_4 = \frac{\int_{\Omega_{por}} d\Omega_{por}}{\int_{\Omega_{RVE}} d\Omega_{RVE}} \tag{70}$$

Values of aforementioned functionals are calculated numerically by means of FEM in MSC Patran/Nastran and MSC Marc/Mentat software based on a parametric model. Functionals f_1 and f_2 are calculated by running a static structural analysis under a given load. These functionals are obtained in macroscale analyses proceeded by a series of microscale analyses to establish values of effective material properties. Functional f_3 is calculated by running a thermal analysis of RVE in microscale. Functional f_4 is calculated on the basis of geometry of the model in microscale and does not require solving any boundary-value problems. Problems of maximization of f_3 and f_4 are transformed to minimization problems by multiplying a fitness function value by -1.

5.4.3. Numerical model

Cuboid aluminium solid of dimensions $100 \times 20 \times 20$ mm fixed on one side and subject to thermal and mechanical loads is considered (Fig. 68). Uniform distributed load P is applied on the surface opposite of the fixed (displacement u_0) side. On both these surfaces temperatures T_1 and T_2 are known.

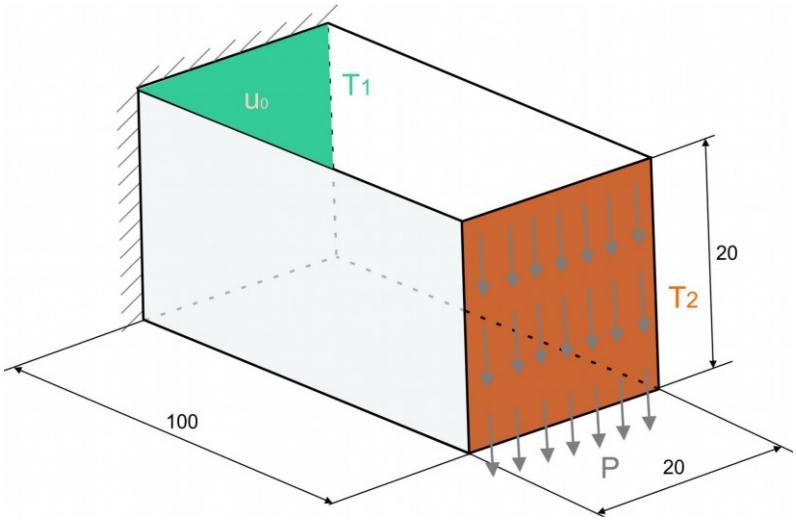


Fig. 68: Macromodel under thermal and mechanical loads

Boundary conditions and material properties of aluminium are presented in Tab. 24

Tab. 24: Boundary conditions and material properties

Boundary condition or material property	Symbol	Value
Displacement	u_0	0
Load	P	360 N (total)
Temperature 1	T_1	0 ºC
Temperature 2	T_2	100 ºC
Young's modulus	E	70 GPa
Poisson's ratio	ν	0.35
Thermal conductivity	K	200 W/(m·K)
Thermal expansion coefficient	α_t	$23 \cdot 10^{-6}$ K^{-1}

Numerical homogenization utilising RVE concept and finite element method is carried out to solve the two-scale thermoelastic problem for a porous system. The aim of homogenization is to determine effective material properties of a non-homogenous structure and use them to solve macroscale problem. For thermo-elastic problems, the determined properties include elasticity constants, thermal expansion coefficients or thermal conductivity coefficients [175]. Thermal expansion coefficient does not require homogenization in analysed example as it does not depend on geometric configuration of microstructure. Linear thermoelasticity problem is described by a set of partial differential equations of heat conduction and elasticity, considering thermal strains [176]–[179]:

$$\begin{gathered} kT_{,ii} = 0 \\ \mu u_{i,jj} + (\mu + \lambda) u_{j,ji} - (3\lambda + 2\mu)\alpha_T T_{,i} = 0 \end{gathered} \tag{71}$$

where k is thermal conductivity, T is temperature, u is displacement, α_T is linear expansion coefficient, μ and λ are Lamé constants.

Differential equations (71) need to be supplemented with a set of mechanical boundary conditions:

$$\begin{gathered} x \in \Gamma_u : u(x) = u^0(x) \\ x \in \Gamma_p : p(x) = p(x) \end{gathered} \tag{72}$$

and thermal boundary conditions:

$$\begin{gathered} x \in \Gamma_T : T(x) = T^0(x) \\ x \in \Gamma_q : q(x) = q^0(x) \\ x \in \Gamma_c : q(x) = \alpha(T(x) - T^\infty(x)) \end{gathered} \tag{73}$$

where u^0 and p^0 are known displacements and mechanical loads on the respective parts of the boundary Γ_u and Γ_p, T^0 and q^0 are known temperature and heat flux on the respective parts of the boundary Γ_T and Γ_q. Last boundary condition is a convection condition, where α stands for heat convection coefficient and $T^\infty$is ambient temperature around part of the boundary Γ_c.

Boundary of the body is defined by parts of the boundary where thermal and mechanical boundary conditions are applied. To solve the thermo-mechanical boundary value problem, FEM is used. Partial differential equations governing the problem are discretized and transformed into a system of algebraic equations. After taking into consideration boundary conditions, a resulting system of equations in the matrix form is obtained:

$$\mathbf{K_T T} = \mathbf{Q} \tag{74}$$

$$\mathbf{K_M U} = \mathbf{F} + \mathbf{F_T} \tag{75}$$

where $\mathbf{K_T}$ and $\mathbf{K_M}$ are global matrices of thermal conductivity and stiffness respectively, assembled by aggregation of local matrices according to FEM discretization scheme. **T**, **Q**, **U** and **F** are global vectors of: temperatures, heat fluxes, displacement, and mechanical loads respectively. As weakly coupling between physical fields is assumed, electric, thermal, and mechanical analyses are solved in a successive manner. Coupling is realised by transferring of loads into subsequent analyses. $\mathbf{F_T}$ is a vector of loads due to thermal strain obtained from equation (74).

A set of assumptions is taken into consideration in the process of numerical homogenization using RVE including:

- principle of scale separation:

$$\frac{l}{L} \ll 1 \tag{76}$$

where l and L are characteristic dimensions of a structure in a micro- (RVE) and macroscale,

- averaging theorem:

$$\langle \cdot \rangle = \frac{1}{|\Omega_{RVE}|} \int_{\Omega_{RVE}} (\cdot) d\Omega_{RVE} \tag{77}$$

where $\langle \cdot \rangle$ denotes the average macroscopic value of a given field over the volume V of the RVE,

- Hill's condition providing the equality of the averaged micro-scale energy density and the macro-scale energy density at the selected point of macro-structure corresponding to the RVE:

$$\langle \sigma_{ij} \varepsilon_{ij} \rangle = \langle \sigma_{ij} \rangle \langle \varepsilon_{ij} \rangle \tag{78}$$

where σ_{ij} and ε_{ij} are stress and strain tensors.

For heat conduction problem Hill's condition takes the following form:

$$\langle T_{,i} q_i \rangle = \langle T_{,i} \rangle \langle q_i \rangle \tag{79}$$

where $T_{,i}$ and q_i are temperature gradient and heat flux.

Numerical homogenization using RVE approach with periodic boundary conditions is used. FEM analysis in the microscale is performed to obtain average stress and heat fluxes in RVE to determine effective material properties according to equation (77).

Hooke's law in the microscale takes the following form:

$$\langle \sigma_{ij} \rangle = \mathbf{c}'_{ijkl} \langle \varepsilon_{ij} \rangle \tag{80}$$

and Fourier's law:

$$\langle q_i \rangle = \mathbf{k}'_{ij} \langle T_{,j} \rangle \tag{81}$$

where $\mathbf{c}'_{ijkl}$ is a tensor of elastic constants (using Voigt notation) of the RVE, described by a set of nine independent constants and takes the following form:

$$\mathbf{c}'_{ij} = \begin{bmatrix} c_{11} & c_{12} & c_{13} & 0 & 0 & 0 \\ c_{21} & c_{22} & c_{23} & 0 & 0 & 0 \\ c_{31} & c_{32} & c_{33} & 0 & 0 & 0 \\ 0 & 0 & 0 & c_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & c_{55} & 0 \\ 0 & 0 & 0 & 0 & 0 & c_{66} \end{bmatrix} \tag{82}$$

whereas $\mathbf{k}'_{ij}$ is a tensor of thermal conductivity coefficients for non-crystalline anisotropic materials which is described by three independent constants and takes the following form:

$$\mathbf{k}'_{ij} = \begin{bmatrix} k_{11} & 0 & 0 \\ 0 & k_{22} & 0 \\ 0 & 0 & k_{33} \end{bmatrix} \tag{83}$$

Determining effective elastic constants requires running six analyses and to determine effective thermal constants three analyses are necessary. Initial unitary strain is applied to the RVE model and microscale analysis is performed to solve a single column or row of the tensor of effective elastic constraints. The same approach is used to calculate values of elements of the tensor of effective thermal constants. In total there are 9 microscale analyses required as a prerequisite to eventually solving a macroscale analysis resulting in obtaining value of

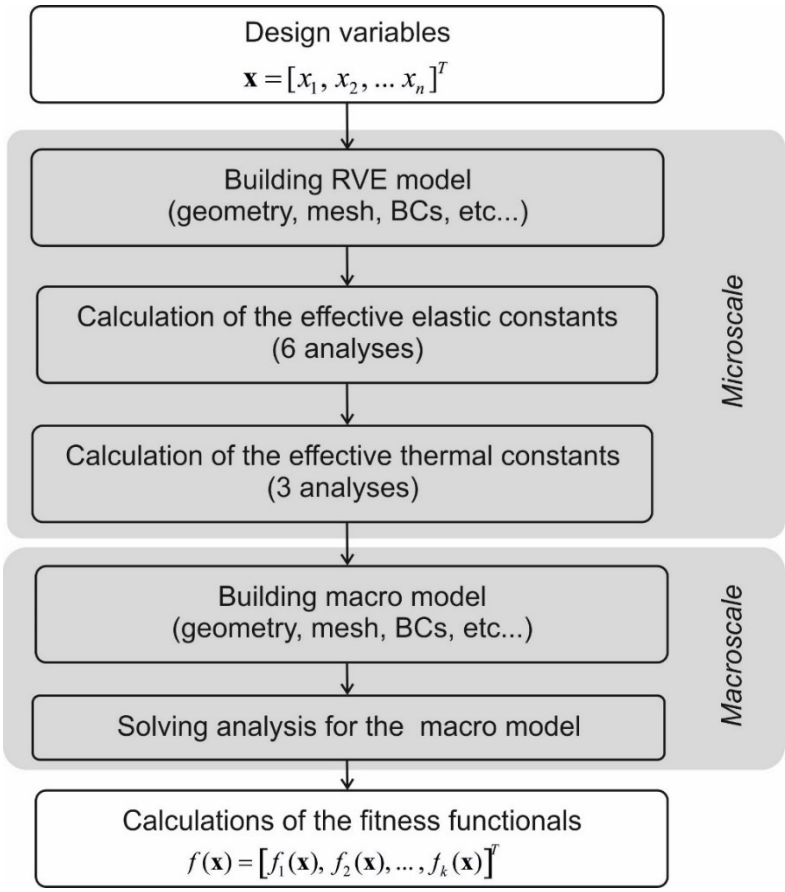


Fig. 69: Fitness function evaluation procedure

optimization functional (Fig. 69). MSC.Mentat/Marc software was used for FEM computations in both micro- and macroscale. Automatic generation of models for the multiscale analysis was performed utilising an in-house procedure implemented in C++ programming language and internal MSC script language.

Analyses in the microscale concern RVE model and are based on the parametric model in which the geometry of a microstructure is governed by a set of design variables, are shown in the Tab. 25 supplemented with limits imposed on them.

Tab. 25: Design variables

Symbol	Design variable	Lower limit	Upper limit
l	length of revolve axis	0.1	0.8
x_1	position of control point 1	0.01	0.3
x_2	position of control point 2	0.01	0.45
x_3	position of control point 3	0.01	0.45
x_4	position of control point 4	0.01	0.45
x_5	position of control point 5	0.01	0.3
α_1	rotation angle 1	0°	90°
α_2	rotation angle 2	0°	90°

Geometry of the RVE represents geometry of a single pore in a periodic microstructure. Parametric model is developed in a way to provide a high flexibility of shapes, which can be described by a limited number of design variables. Geometry of the pore is assumed to be axisymmetric and is constructed by revolving a closed B-spline curve of order 4 around an axis. Shape of the curve is controlled by 5 design variables representing position of control points (Fig. 70b) and length of an axis is controlled by an additional design variable (Fig. 70a). After the pore volume is generated (Fig. 70c) it is furthermore rotated around two axes (Fig. 70d). The shape of a pore is eventually subtracted from a volume of a full unit block.

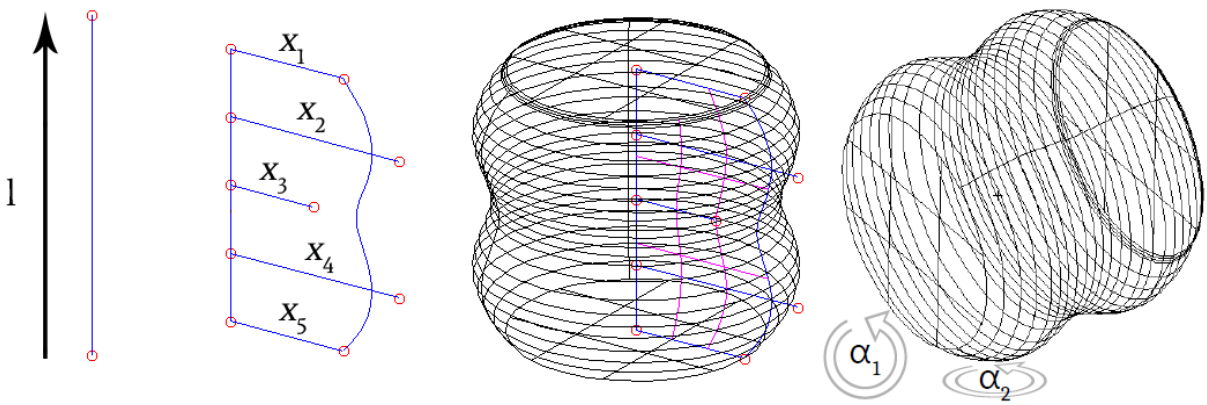


Fig. 70: Design variables

a) b) c) d)

After creating the geometry of a RVE it is discretized using tet4 elements (Fig. 71).

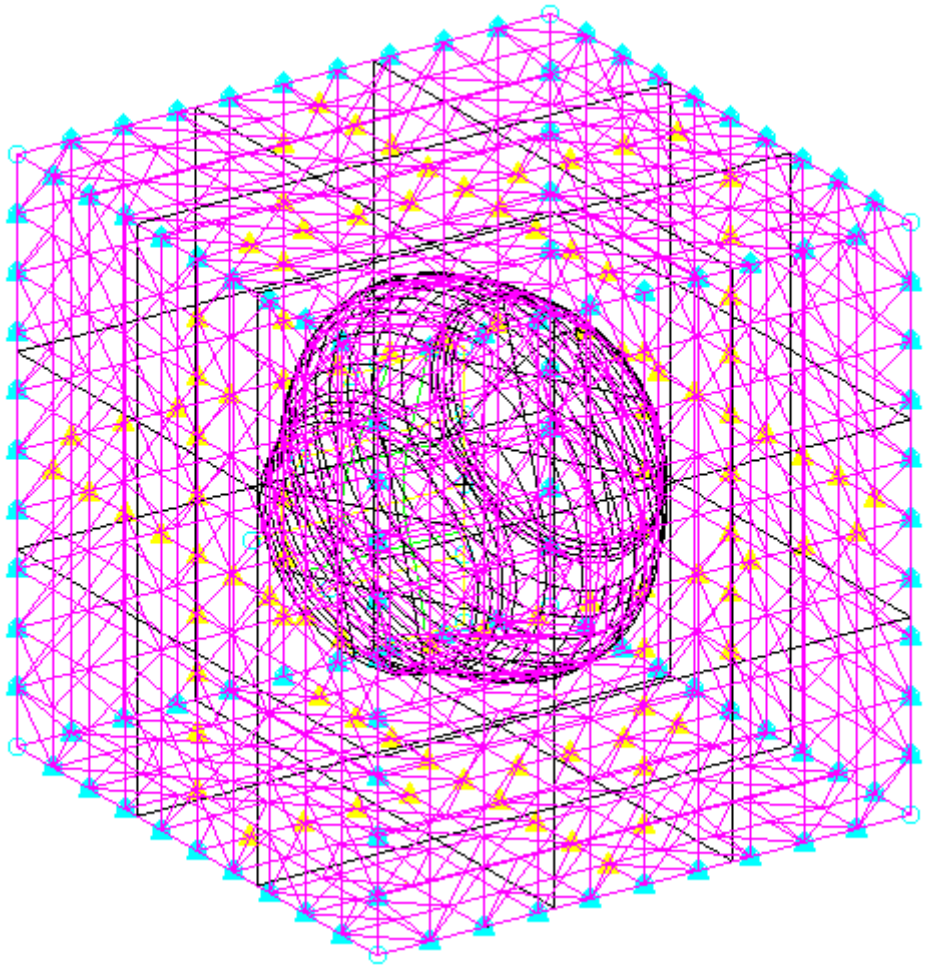


Fig. 71: Discretization of the RVE

5.4.4. Results

Results of this research were obtained after solving approximately 10 000 boundary value problems and consist of a set of 100 Pareto-optimal (non-dominated) solutions found during the course of optimization using the parameters described in chapter 3. Values of objective functions for the obtained solutions are shown in the figures in two ways. A 3D scatter plot with values of the f_2 function represented by colour, and the remaining fitness function as values on the axes is shown in the Fig. 72. Scatter plot matrix of obtained solutions is shown in the Fig. 73. Along the diagonal are histogram plots of objective function values.

On the basis of presented plots of non-dominated solution some conclusions on the relationships between objectives can by drawn. On the 4D (3D + colour) plot, the solutions are arranged approximately in line, which indicates the existence of a degenerate real Pareto front in the problem. Looking at the scatter plot matrix it can be noticed that this is caused by

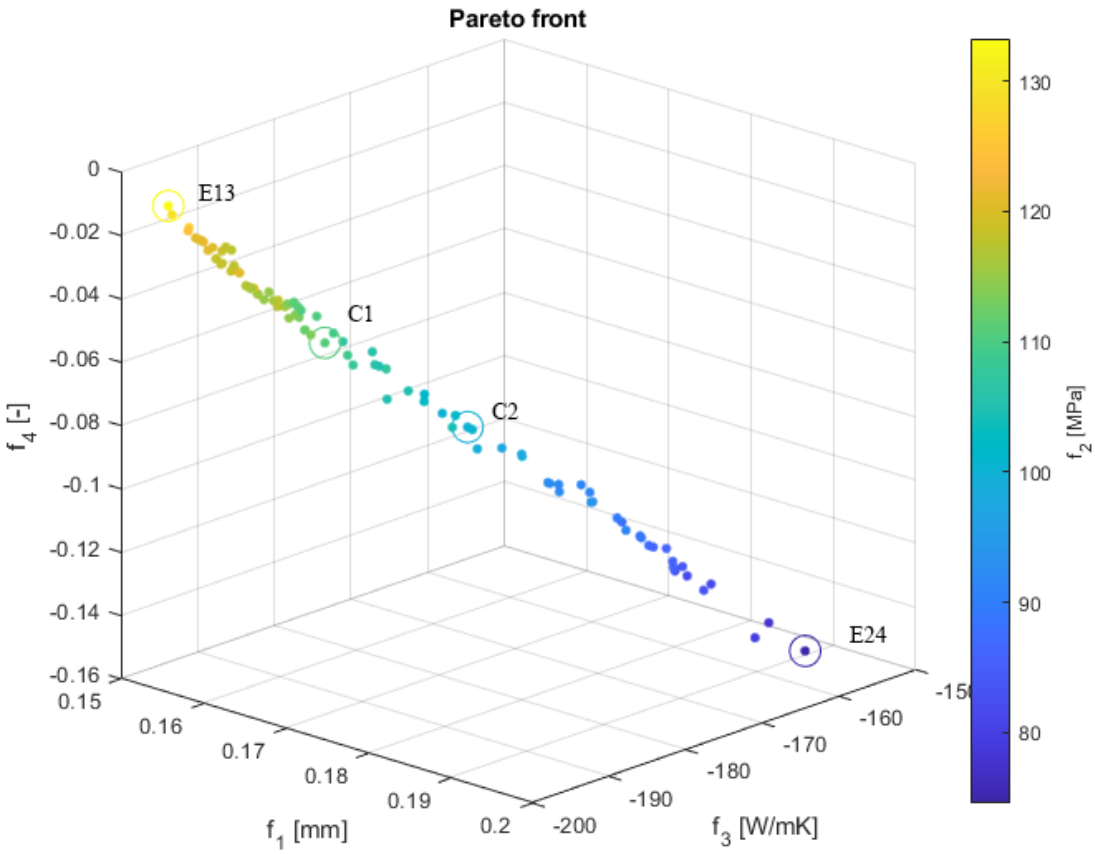


Fig. 72: 3D scatter plot of obtained solutions with f_2 in colour

solutions displaying a similar nature pairwise. Pairs of solutions in this example are proportional ($f_1 - f_3$ and $f_2 - f_4$) or inversely proportional (all remaining pairs). Examining the physical nature of considered objectives such situation can be explained. For solutions in which the size of pore was small, the thermal conductivity and stiffness was increased and therefore values of optimization functionals f_1 and f_3 improved, which however stand in conflicting nature of objectives related to volume and equivalent stress in the model, expressed in functionals f_2 and f_4. In general, it might be possible to find solutions with low value of both objectives which should come in agreement. Such a scenario is possible due to areas of increased stress concentration existing in the micromodel – pore of low volume, even though usually results in low stress level, might in case of specific shapes exhibit locations of stress concentration. Nevertheless, such solutions are not interesting from the optimization point of view as they are dominated by other solutions in the solution space.

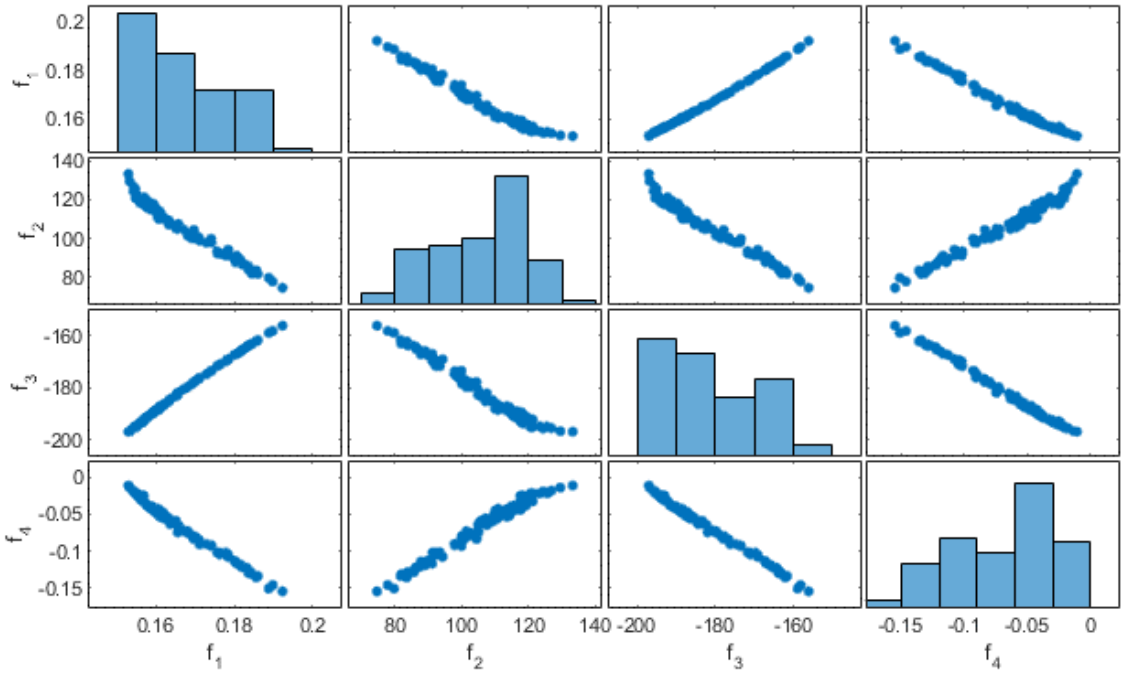


Fig. 73: Scatter plot matrix of obtained solutions

Among found solutions some solutions were highlighted. Solutions displaying extreme values of objectives f_1 and f_3 (ES13), f_2 and f_4 (ES24) as well as two compromise solutions (C1 and C2) are presented in detail. Geometry of these solutions is shown in Fig. 74 and values of objective functions and design variables in Tab. 26.

Tab. 26: Values of fitness functions and design variables of selected solutions

Design	E13	E24	C1	C2
f_1 [mm]	0.152926	0.19227	0.162077	0.170551
f_2 [MPa]	133.239	74.6041	110.688	99.7959
f_3 [W/(m·K)]	-196.962	-156.136	-186.349	-176.84
f_4 [-]	-0.0109179	-0.154978	-0.0559121	-0.0838825
l [μm]	0.2543	0.5926	0.2365	0.3875
x_1 [μm]	0.093	0.3076	0.329	0.2941
x_2 [μm]	0.165	0.3437	0.3301	0.343
x_3 [μm]	0.1092	0.3335	0.3454	0.0925
x_4 [μm]	0.1047	0.1397	0.1448	0.3018
x_5 [μm]	0.1437	0.3241	0.1708	0.3064
α_1 [°]	88.4619	42.3712	22.1711	1.8684
α_2 [°]	48.3682	76.4589	89.3652	32.425

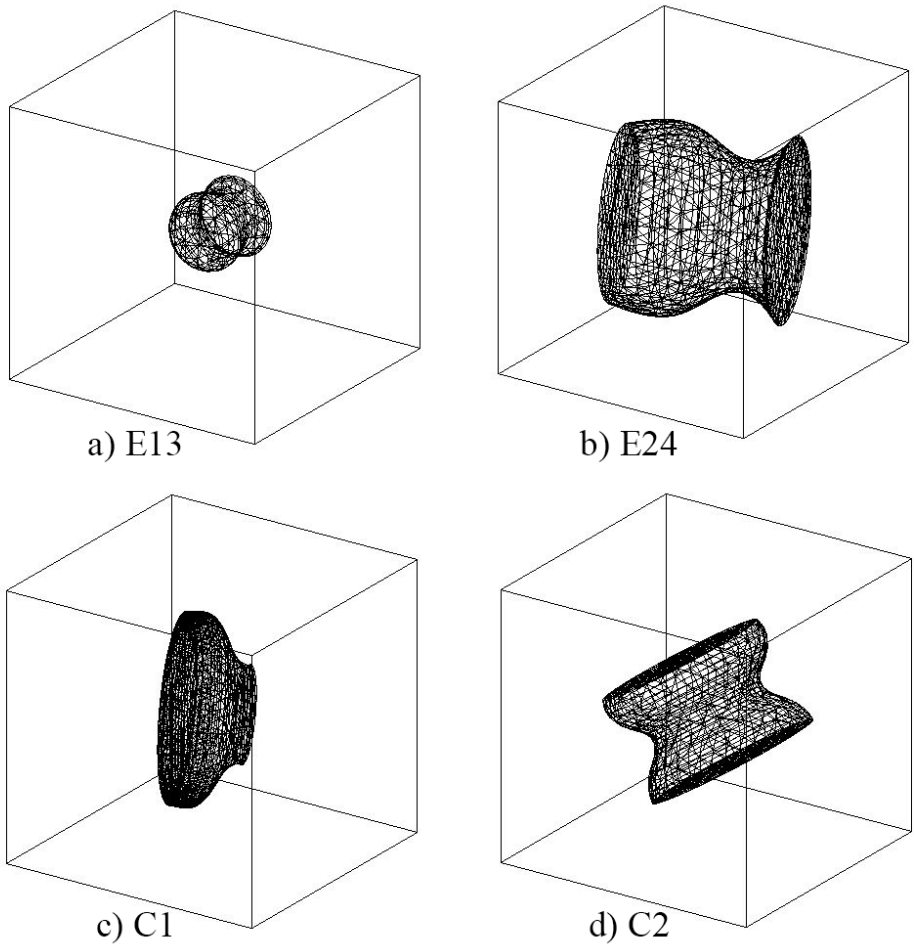


Fig. 74: Geometry of selected solutions

5.4.5. Final remarks

Multiobjective optimization of porous material in the multiscale was presented. Optimization of the geometry of microstructure was solved for four different proposed functionals. More functionals related to particular needs asked of a porous structure can be formulated. A set of 100 non-dominated solutions was found. Proposed methods of visualisation help to draw conclusions on natures of the objectives. Among found solutions, some designs related to extreme value of optimization functions were highlighted, compared in terms of design variables and geometry of these solutions was shown. Application of presented methods can improve the multiscale design of porous systems.

6. SUMMARY

The dissertation investigates the subject of optimal design of mechanical systems for many criteria. In the literature review basic information on the process of design are presented with special attention given to the optimization tasks. Overview of finite element method as a simulation tool from the perspective of optimization of mechanical systems is included. Metrics and benchmark functions are reviewed in the context of evaluating performance of multiobjective optimization algorithms. Visualisation techniques enhancing post-optimization decision making process are described.

Next part of the dissertation presents developed multiobjective optimization algorithm belonging to a group of soft computing methods. Proposed algorithm is based on a differential evolution and game theory paradigms. Suggested algorithm takes advantage of a game theoretic cooperative approach and eliminates some of drawbacks of other soft computing methods in the optimization of mechanical systems. Elements of game theory are introduced along with basics of differential evolution. General idea a novel multiobjective algorithm is described and presented using a flowchart, a pseudocode and supplemented with an example. Implementation of the algorithm in C++ programming language is briefly described. Methods of information exchange between developed algorithm and FEM software responsible for numerically obtaining values of objective functions during the course of optimization are discussed.

In the subsequent part of the dissertation, the algorithm is comprehensively tested using previously introduced benchmark functions and performance metrics. A set of mathematical test functions exhibiting features distinctive of mechanical systems is utilised. Quality of results is assessed using a hypervolume indicator. Mean and standard deviation of metrics are calculated over 30 runs of algorithm and compared with results obtained by other multiobjective optimization algorithms: NSGA-II and NSGA-III. Test problems with a varied number of design variables (up to 20) and objective functions (up to 8) are investigated. Within analysed framework the developed algorithm was proven to be efficient and competitive with compared algorithms.

Ensuing part of the dissertation provides examples on application of the developed tool in the optimal design of mechanical systems. A set of three analytical problems: pressure vessel design, speed reducer design and stepped cantilever beam are examined. These analytical problems are produced by a transformation of constrained single-objective optimization problems into multiobjective problems. Furthermore, the algorithm is used to solve complex, numerical problems: airfoil optimization, electrothermal microactuators optimization and multiscale porous material optimization. In these problems, values of objective function are obtained by means of FEM based on parametric models. For each of analysed problems, algorithm found a set of new diverse designs. Results were presented in the form of tables, graphs, and figures. On the basis of additional established preferences and using proposed visualisation tools, a post-optimization decision making process was aided resulting in a narrowed down set of solutions.

Considering results of provided tests and examples a conclusion can be drawn that an algorithm based on differential evolution and elements of game theory can be used as an efficient tool in the optimal design of mechanical systems concerning multiple criteria, which completes the proof of the thesis.

Directions of further development of presented method can be expected in the following areas:

- Modifications of a differential evolution algorithm can be used.
- Application of parallel computation in the optimization of mechanical systems can reduce the time required to solve optimization tasks.
- Application of meta-models instead of solving boundary value-problems can further improve efficiency of the algorithm.
- The algorithm can be hybridised with a local-search tool to improve the exploitation of search space in the regions of interests.
- Extensive tests on the effect of parameters on the results of optimization can be performed, including introducing the idea of self-adaptation.

REFERENCES

[1] M. Dietrich *et al.*, *Podstawy konstrukcji maszyn. 1 1*. Warszawa: Wydawnictwo Naukowe PWN, 2017.

[2] P. R. N. Childs, *Mechanical design engineering handbook*. Amsterdam: Butterworth-Heinemann, 2014.

[3] R. G. Budynas, J. K. Nisbett, and J. E. Shigley, *Shigley's mechanical engineering design*, Tenth edition. New York, NY: McGraw-Hill Education, 2015.

[4] T. Burczyński, W. Kuś, W. Beluch, A. Długosz, A. Poteralski, and M. Szczepanik, *Intelligent computing in optimal design*. Cham: Springer, 2020.

[5] R. Roy, S. Hinduja, and R. Teti, 'Recent advances in engineering design optimisation: Challenges and future trends', *CIRP Annals*, vol. 57, no. 2, pp. 697–715, Jan. 2008, doi: 10.1016/j.cirp.2008.09.007.

[6] W. Sztoff, *Modelowanie i filozofia*. Państwowe Wydawnictwo Naukowe, 1971.

[7] S. Rangavajhala and A. Messac, 'Towards a Better Understanding of the Equality Constraints in Robust Design Optimization', presented at the 11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Portsmouth, Virginia, Sep. 2006. doi: 10.2514/6.2006-6925.

[8] M.-H. Lin, J.-F. Tsai, and C.-S. Yu, 'A Review of Deterministic Optimization Methods in Engineering and Management', *Mathematical Problems in Engineering*, vol. 2012, pp. 1–15, 2012, doi: 10.1155/2012/756023.

[9] A. Cauchy, 'Méthode générale pour la résolution des systemes d'équations simultanées', *Comp. Rend. Sci. Paris*, vol. 25, no. 1847, pp. 536–538, 1847.

[10] C. Lemaréchal, *Cauchy and the Gradient Method*. 2012.

[11] J. Hadamard, *Mémoire sur le problème d'analyse relatif à l'équilibre des plaques élastiques encastrées*. Imprimerie nationale, 1908.

[12] H. B. Curry, 'The method of steepest descent for non-linear minimization problems', *Quart. Appl. Math.*, vol. 2, no. 3, pp. 258–261, 1944, doi: 10.1090/qam/10667.

[13] A. Stachurski and A. Wierzbicki, *Podstawy optymalizacji*. Warszawa: Oficyna Wydawnicza Politechniki Warszawskiej, 1999.

[14] D. C. Liu and J. Nocedal, 'On the limited memory BFGS method for large scale optimization', *Mathematical Programming*, vol. 45, no. 1, pp. 503–528, Aug. 1989, doi: 10.1007/BF01589116.

[15] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, 'A Limited Memory Algorithm for Bound Constrained Optimization', *SIAM J. Sci. Comput.*, vol. 16, no. 5, pp. 1190–1208, Sep. 1995, doi: 10.1137/0916069.

[16] B. Choudhury and R. M. Jha, Eds., 'Soft Computing Techniques', in *Soft Computing in Electromagnetics: Methods and Applications*, Cambridge: Cambridge University Press, 2016, pp. 9–44. doi: 10.1017/CBO9781316402924.003.

[17] M. Abdel-Basset, L. Abdel-Fatah, and A. K. Sangaiah, 'Metaheuristic Algorithms: A Comprehensive Review', in *Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications*, Elsevier, 2018, pp. 185–231. doi: 10.1016/B978-0-12-813314-9.00010-4.

[18] L. Bianchi, M. Dorigo, L. M. Gambardella, and W. J. Gutjahr, 'A survey on metaheuristics for stochastic combinatorial optimization', *Nat Comput*, vol. 8, no. 2, pp. 239–287, Jun. 2009, doi: 10.1007/s11047-008-9098-4.

[19] A. E. Eiben and C. A. Schippers, 'On Evolutionary Exploration and Exploitation', *Fundamenta Informaticae*, vol. 35, no. 1–4, pp. 35–50, 1998, doi: 10.3233/FI-1998-35123403.

[20] M. Črepinšek, S.-H. Liu, and M. Mernik, 'Exploration and exploitation in evolutionary algorithms: A survey', *ACM Comput. Surv.*, vol. 45, no. 3, pp. 1–33, Jun. 2013, doi: 10.1145/2480741.2480752.
[21] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, 3rd ed. Berlin Heidelberg: Springer-Verlag, 1996. doi: 10.1007/978-3-662-03315-9.
[22] C. A. Coello Coello, 'An Introduction to Evolutionary Algorithms and Their Applications', in *Advanced Distributed Systems*, Berlin, Heidelberg, 2005, pp. 425–442. doi: 10.1007/11533962_39.
[23] A. Slowik and H. Kwasnicka, 'Evolutionary algorithms and their applications to engineering problems', *Neural Comput & Applic*, vol. 32, no. 16, pp. 12363–12379, Aug. 2020, doi: 10.1007/s00521-020-04832-8.
[24] L. G. de la Fraga and C. A. Coello Coello, 'A Review of Applications of Evolutionary Algorithms in Pattern Recognition', in *Pattern Recognition, Machine Intelligence and Biometrics*, P. S. P. Wang, Ed. Berlin, Heidelberg: Springer, 2011, pp. 3–28. doi: 10.1007/978-3-642-22407-2_1.
[25] J. H. Holland, J. H. Holland, and P. of P. and of E. E. and C. S. J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. University of Michigan Press, 1975.
[26] J. H. Holland, 'Concerning efficient adaptive systems', *Self-Organizing Systems*, vol. 230, 1962.
[27] J. H. Holland, 'Outline for a Logical Theory of Adaptive Systems', *J. ACM*, vol. 9, no. 3, pp. 297–314, Jul. 1962, doi: 10.1145/321127.321128.
[28] J. R. Koza and J. R. Koza, *Genetic programming: on the programming of computers by means of natural selection*, vol. 1. MIT press, 1992.
[29] I. Rechenberg, *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog, 1973. [Online]. Available: https://books.google.pl/books?id=iw2hcQAACAAJ
[30] L. J. Alvin J. Owens and Michael J. Walsh FOGEL, *Artificial Intelligence through Simulated Evolution*. 1967.
[31] J. Kennedy and R. Eberhart, 'Particle swarm optimization', in *Proceedings of ICNN'95 - International Conference on Neural Networks*, Nov. 1995, vol. 4, pp. 1942–1948 vol.4. doi: 10.1109/ICNN.1995.488968.
[32] X.-S. Yang, *Nature-inspired optimization algorithms*, 1st edition. Amsterdam Boston Heidelberg London New York Oxford Paris San Diego San Francisco Singapore Sydney Tokyo: Elsevier, 2014.
[33] J. Kephart, 'A biologically inspired immune system for computers', 1994, doi: 10.7551/mitpress/1428.003.0017.
[34] D. Dasgupta, Ed., *Artficial immune systems and their applications*. Berlin ; New York: Springer, 1998.
[35] L. N. de Castro, L. N. Castro, and J. Timmis, *Artificial Immune Systems: A New Computational Intelligence Approach*. Springer Science & Business Media, 2002.
[36] Z. KLAWIKOWSKA and B. PUCHALSKI, 'SKUTECZNOŚĆ NOWOCZESNYCH ALGORYTMÓW OPTYMALIZACJI CZERPIĄCYCH INSPIRACJĘ Z PROCESÓW NATURALNYCH', *ZN WEiA PG*, vol. 2020, no. 71, pp. 35–40, Dec. 2020, doi: 10.32016/1.71.04.
[37] M. Milano and P. Van Hentenryck, Eds., *Hybrid optimization: the ten years of CPAIOR*. New York: Springer, 2011.
[38] K. Sorensen, M. Sevaux, and F. Glover, 'A History of Metaheuristics', *arXiv:1704.00853 [cs]*, Apr. 2017, Accessed: Sep. 03, 2021. [Online]. Available: http://arxiv.org/abs/1704.00853

[39] D. Molina, J. Poyatos, J. D. Ser, S. García, A. Hussain, and F. Herrera, 'Comprehensive Taxonomies of Nature- and Bio-inspired Optimization: Inspiration Versus Algorithmic Behavior, Critical Analysis Recommendations', *Cogn Comput*, vol. 12, no. 5, pp. 897–939, Sep. 2020, doi: 10.1007/s12559-020-09730-8.

[40] K. Deb and D. Kalyanmoy, *Multi-Objective Optimization Using Evolutionary Algorithms*. USA: John Wiley & Sons, Inc., 2001.

[41] R. C. Purshouse and P. J. Fleming, 'On the Evolutionary Optimization of Many Conflicting Objectives', *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 770–784, Dec. 2007, doi: 10.1109/TEVC.2007.910138.

[42] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, 'A fast and elitist multiobjective genetic algorithm: NSGA-II', *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, Apr. 2002, doi: 10.1109/4235.996017.

[43] MathWorks, 'Company Overview'. Feb. 2020. Accessed: Jun. 24, 2021. [Online]. Available: https://uk.mathworks.com/content/dam/mathworks/handout/2020-company-factsheet-8-5x11-8282v20.pdf

[44] Y. Yusoff, M. S. Ngadiman, and A. M. Zain, 'Overview of NSGA-II for Optimizing Machining Process Parameters', *Procedia Engineering*, vol. 15, pp. 3978–3983, Jan. 2011, doi: 10.1016/j.proeng.2011.08.745.

[45] A. Parmar, P. Ramkumar, and K. Shankar, 'Optimization of Planetary Gearbox Using NSGA-II', in *Recent Advances in Mechanical Engineering*, Singapore, 2021, pp. 367–376. doi: 10.1007/978-981-15-7711-6_38.

[46] W. Yahui, S. Ling, Z. Cai, F. Liuqiang, and J. Xiangjie, 'NSGA-II algorithm and application for multi-objective flexible workshop scheduling', *Journal of Algorithms & Computational Technology*, vol. 14, p. 1748302620942467, Jan. 2020, doi: 10.1177/1748302620942467.

[47] X. Xu, W. Zhang, and X. Ding, 'Modular design method for filament winding process equipment based on GGA and NSGA-II', *Int J Adv Manuf Technol*, vol. 94, no. 5, pp. 2057–2076, Feb. 2018, doi: 10.1007/s00170-017-0929-2.

[48] A. V. Pombo, V. F. Pires, and J. M. Pina, 'Application of NSGA-II Algorithm to Multiobjective Optimization of Switching Devices Placement in Electric Power Distribution Systems', in *Technological Innovation for Collective Awareness Systems*, Berlin, Heidelberg, 2014, pp. 380–387. doi: 10.1007/978-3-642-54734-8_42.

[49] Q. Ma, D. Xu, P. lv, and Y. Shi, 'Application of NSGA-II in Parameter Optimization of Extended State Observer', in *Challenges of Power Engineering and Environment*, Berlin, Heidelberg, 2007, pp. 587–592. doi: 10.1007/978-3-540-76694-0_109.

[50] K. Deb and H. Jain, 'An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints', *IEEE Trans. Evol. Computat.*, vol. 18, no. 4, pp. 577–601, Aug. 2014, doi: 10.1109/TEVC.2013.2281535.

[51] H. Li, K. Deb, Q. Zhang, P. Suganthan, and L. Chen, 'Comparison between MOEA/D and NSGA-III on a set of many and multi-objective benchmark problems with challenging difficulties', *Swarm and Evolutionary Computation*, vol. 46, Feb. 2019, doi: 10.1016/j.swevo.2019.02.003.

[52] C. Tsung-Che, *nsga3cpp: A C++ implementation of NSGA-III*. 2014. [Online]. Available: http://web.ntnu.edu.tw/~tcchiang/publications/nsga3cpp/nsga3cpp.htm

[53] A. D. Belegundu and T. R. Chandrupatla, *Optimization concepts and applications in engineering*, 2nd ed. New York: Cambridge University Press, 2011.

[54] T. R. Chandrupatla and A. D. Belegundu, *Introduction to finite elements in engineering*, 3rd ed. Upper Saddle River, N.J: Prentice Hall, 2002.

[55] J. Branke, K. Deb, K. Miettinen, and R. Slowiński, Eds., *Multiobjective Optimization: Interactive and Evolutionary Approaches*. Berlin Heidelberg: Springer-Verlag, 2008. doi: 10.1007/978-3-540-88908-3.
[56] T. Kohonen, *Self-Organizing Maps*, 3rd ed. Berlin Heidelberg: Springer-Verlag, 2001. doi: 10.1007/978-3-642-56927-2.
[57] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, 'Scalable Test Problems for Evolutionary Multiobjective Optimization', in *Evolutionary Multiobjective Optimization: Theoretical Advances and Applications*, A. Abraham, L. Jain, and R. Goldberg, Eds. London: Springer, 2005, pp. 105–145. doi: 10.1007/1-84628-137-7_6.
[58] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, 'Scalable multi-objective optimization test problems', in *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)*, Honolulu, HI, USA, 2002, vol. 1, pp. 825–830. doi: 10.1109/CEC.2002.1007032.
[59] S. Huband, P. Hingston, L. Barone, and L. While, 'A review of multiobjective test problems and a scalable test problem toolkit', *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 5, pp. 477–506, Oct. 2006, doi: 10.1109/TEVC.2005.861417.
[60] S. Huband, L. Barone, L. While, and P. Hingston, 'A Scalable Multi-objective Test Problem Toolkit', in *Evolutionary Multi-Criterion Optimization*, Berlin, Heidelberg, 2005, pp. 280–295. doi: 10.1007/978-3-540-31880-4_20.
[61] H. Ishibuchi, H. Masuda, and Y. Nojima, 'Pareto Fronts of Many-Objective Degenerate Test Problems', *IEEE Trans. Evol. Computat.*, vol. 20, no. 5, pp. 807–813, Oct. 2016, doi: 10.1109/TEVC.2015.2505784.
[62] 'Comparison of Multiobjective Evolutionary Algorithms: Empirical Results | Evolutionary Computation'. https://dl.acm.org/doi/10.1162/106365600568202 (accessed Feb. 24, 2021).
[63] D. A. V. Veldhuizen and G. B. Lamont, *Multiobjective Evolutionary Algorithm Research: A History and Analysis*. 1998.
[64] X. Li, J. Zheng, and J. Xue, 'A Diversity Metric for Multi-objective Evolutionary Algorithms', in *Advances in Natural Computation*, Berlin, Heidelberg, 2005, pp. 68–73. doi: 10.1007/11539902_8.
[65] Y.-N. Wang, L.-H. Wu, and X.-F. Yuan, 'Multi-objective self-adaptive differential evolution with elitist archive and crowding entropy-based diversity measure', *Soft Comput*, vol. 14, no. 3, p. 193, Jan. 2009, doi: 10.1007/s00500-008-0394-9.
[66] L. While, L. Bradstreet, L. Barone, and P. Hingston, 'Heuristics for optimizing the calculation of hypervolume for multi-objective optimization problems', in *2005 IEEE Congress on Evolutionary Computation*, Sep. 2005, vol. 3, pp. 2225-2232 Vol. 3. doi: 10.1109/CEC.2005.1554971.
[67] L. While, P. Hingston, L. Barone, and S. Huband, 'A faster algorithm for calculating hypervolume', *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 1, pp. 29–38, Feb. 2006, doi: 10.1109/TEVC.2005.851275.
[68] C. M. Fonseca, L. Paquete, and M. Lopez-Ibanez, 'An Improved Dimension-Sweep Algorithm for the Hypervolume Indicator', in *2006 IEEE International Conference on Evolutionary Computation*, Jul. 2006, pp. 1157–1163. doi: 10.1109/CEC.2006.1688440.
[69] N. Beume and G. Rudolph, 'Faster S-metric calculation by considering dominated hypervolume as Klee s measure problem', Technische Universität Dortmund, Jul. 2006. doi: 10.17877/DE290R-12786.
[70] A. Ameljańczyk, 'Wektorowa optymalizacja w modelach decyzyjnych ze szczególnym uwzględnieniem modeli growych', *Dodatek do Biuletyn WAT*, no. 2, 1979.

[71] S. Özyıldırım and N. M. Alemdar, 'Learning the optimum as a Nash equilibrium', *Journal of Economic Dynamics and Control*, vol. 24, no. 4, pp. 483–499, Apr. 2000, doi: 10.1016/S0165-1889(99)00012-3.
[72] R. Spallino and S. Rizzo, 'Multi-objective discrete optimization of laminated structures', *Mechanics Research Communications*, vol. 29, no. 1, pp. 17–25, Jan. 2002, doi: 10.1016/S0093-6413(02)00227-6.
[73] J. Periaux, H. Q. Chen, B. Mantel, M. Sefrioui, and H. T. Sui, 'Combining game theory and genetic algorithms with application to DDM-nozzle optimization problems', *Finite Elements in Analysis and Design*, vol. 37, no. 5, pp. 417–429, May 2001, doi: 10.1016/S0168-874X(00)00055-X.
[74] M. Sefrioui and J. Periaux, *Nash Genetic Algorithms: Examples and applications*, vol. 1. 2000, p. 516 vol.1. doi: 10.1109/CEC.2000.870339.
[75] D. Lee, L. Gonzalez, J. Periaux, and S. Karkenahalli, 'Efficient Hybrid-Game Strategies Coupled to Evolutionary Algorithms for Robust Multidisciplinary Design Optimization in Aerospace Engineering', *Evolutionary Computation, IEEE Transactions on*, vol. 15, pp. 133–150, May 2011, doi: 10.1109/TEVC.2010.2043364.
[76] D. Lee, L. F. Gonzalez, J. Periaux, K. Srinivas, and E. Onate, 'Hybrid-Game Strategies for multi-objective design optimization in engineering', *Computers & Fluids*, vol. 47, no. 1, pp. 189–204, Aug. 2011, doi: 10.1016/j.compfluid.2011.03.007.
[77] A. K. Dhingra and S. S. Rao, 'A cooperative fuzzy game theoretic approach to multiple objective design optimization', *European Journal of Operational Research*, vol. 83, no. 3, pp. 547–567, Jun. 1995, doi: 10.1016/0377-2217(93)E0324-Q.
[78] K.-B. Sim, D.-W. Lee, and J. Kim, 'Game theory based coevolutionary algorithm: A new computational coevolutionary approach', *International Journal of Control Automation and Systems*, vol. 2, pp. 463–474, Dec. 2004.
[79] Rui Meng, Ye Ye, and Neng-gang Xie, 'Multi-objective optimization design methods based on game theory', in *2010 8th World Congress on Intelligent Control and Automation*, Jul. 2010, pp. 2220–2227. doi: 10.1109/WCICA.2010.5554307.
[80] P. Jarosz and T. Burczyski, 'Coupling of Immune Algorithms and Game Theory in Multiobjective Optimization', in *Artifical Intelligence and Soft Computing*, Berlin, Heidelberg, 2010, pp. 500–507. doi: 10.1007/978-3-642-13232-2_61.
[81] P. Jarosz and T. Burczyñski, 'Artificial Immune System Based on Clonal Selection and Game Theory Principles for Multiobjective Optimization', in *Artificial Immune Systems*, Berlin, Heidelberg, 2011, pp. 321–333. doi: 10.1007/978-3-642-22371-6_28.
[82] P. Jarosz and T. Burczyński, 'Biologically-inspired Methods and Game Theory in Multi-criterion Decision Processes', in *Intelligent Decision Systems in Large-Scale Distributed Environments*, P. Bouvry, H. González-Vélez, and J. Kołodziej, Eds. Berlin, Heidelberg: Springer, 2011, pp. 101–124. doi: 10.1007/978-3-642-21271-0_5.
[83] K. Devlin, *The Unfinished Game: Pascal, Fermat, and the Seventeenth Century Letter that Made the World Modern.* New York: Basic Books, 2008.
[84] G. Shafer, 'Pascal's and Huygens's game-theoretic foundations for probability', p. 27.
[85] J. v. Neumann, 'Zur Theorie der Gesellschaftsspiele', *Math. Ann.*, vol. 100, no. 1, pp. 295–320, Dec. 1928, doi: 10.1007/BF01448847.
[86] H. W. Kuhn and A. W. Tucker, 'John von Neumann's work in the theory of games and mathematical economics', *Bull. Amer. Math. Soc.*, vol. 64, no. 3, pp. 100–122, 1958, doi: 10.1090/S0002-9904-1958-10209-8.
[87] D.-Z. Du and P. M. Pardalos, 'Minimax and Applications'.
[88] F. Brandt, M. Brill, and W. Suksompong, 'An ordinal minimax theorem', *Games and Economic Behavior*, vol. 95, pp. 107–112, Jan. 2016, doi: 10.1016/j.geb.2015.12.010.

[89] J. Von Neumann and O. Morgenstern, *Theory of games and economic behavior*, 60th anniversary ed. Princeton, N.J. ; Woodstock: Princeton University Press, 2007.
[90] R. D. Luce and H. Raiffa, *Games and Decisions: Introduction and Critical Survey*, Revised ed. edition. Dover Publications, 2012.
[91] J. F. Nash, 'Equilibrium points in n-person games', *Proceedings of the National Academy of Sciences*, vol. 36, no. 1, pp. 48–49, Jan. 1950, doi: 10.1073/pnas.36.1.48.
[92] J. F. Nash, 'The Bargaining Problem', *Econometrica*, vol. 18, no. 2, p. 155, Apr. 1950, doi: 10.2307/1907266.
[93] J. Nash, 'Non-Cooperative Games', *The Annals of Mathematics*, vol. 54, no. 2, p. 286, Sep. 1951, doi: 10.2307/1969529.
[94] J. Nash, 'Two-Person Cooperative Games', *Econometrica*, vol. 21, no. 1, p. 128, Jan. 1953, doi: 10.2307/1906951.
[95] D. Fudenberg and J. Tirole, *Game Theory*. MIT Press, 1991.
[96] M. J. Osborne and A. Rubinstein, *A Course in Game Theory*, First Edition, Twelf. Cambridge, Mass: The MIT Press, 1994.
[97] C. Coello, 'Recent Trends in Evolutionary Multiobjective Optimization', 2006, pp. 7–32. doi: 10.1007/1-84628-137-7_2.
[98] R. Storn and K. Price, 'Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces', *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, Dec. 1997, doi: 10.1023/A:1008202821328.
[99] S. Das and P. N. Suganthan, 'Differential Evolution: A Survey of the State-of-the-Art', *IEEE Trans. Evol. Computat.*, vol. 15, no. 1, pp. 4–31, Feb. 2011, doi: 10.1109/TEVC.2010.2059031.
[100] J. Vesterstrom and R. Thomsen, 'A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems', in *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)*, Jun. 2004, vol. 2, pp. 1980-1987 Vol.2. doi: 10.1109/CEC.2004.1331139.
[101] M. Di Carlo, M. Vasile, and E. Minisci, 'Multi-population inflationary differential evolution algorithm with Adaptive Local Restart', in *2015 IEEE Congress on Evolutionary Computation (CEC)*, May 2015, pp. 632–639. doi: 10.1109/CEC.2015.7256950.
[102] M. Vasile, E. Minisci, and M. Locatelli, 'An Inflationary Differential Evolution Algorithm for Space Trajectory Optimization', *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 2, pp. 267–281, Apr. 2011, doi: 10.1109/TEVC.2010.2087026.
[103] A. Glotić and A. Zamuda, 'Short-term combined economic and emission hydrothermal optimization by surrogate differential evolution', *Applied Energy*, vol. 141, pp. 42–56, Mar. 2015, doi: 10.1016/j.apenergy.2014.12.020.
[104] L. D. Arya, P. Singh, and L. S. Titare, 'Differential evolution applied for anticipatory load shedding with voltage stability considerations', *International Journal of Electrical Power & Energy Systems*, vol. 42, no. 1, pp. 644–652, Nov. 2012, doi: 10.1016/j.ijepes.2012.04.006.
[105] P. G. Anil Kumar, K. Alex, A. Jeyanthy, and D. Devaraj, 'Energy Management using DE Algorithm with Two Grids', in *2019 IEEE International Conference on Intelligent Techniques in Control, Optimization and Signal Processing (INCOS)*, Apr. 2019, pp. 1–4. doi: 10.1109/INCOS45849.2019.8951421.
[106] Z. Fan, J. Liu, T. Sorensen, and P. Wang, 'Improved Differential Evolution Based on Stochastic Ranking for Robust Layout Synthesis of MEMS Components', *IEEE Transactions on Industrial Electronics*, vol. 56, no. 4, pp. 937–948, Apr. 2009, doi: 10.1109/TIE.2008.2006935.

[107] L. Mingyong and C. Erbao, 'An improved differential evolution algorithm for vehicle routing problem with simultaneous pickups and deliveries and time windows', *Engineering Applications of Artificial Intelligence*, vol. 23, no. 2, pp. 188–195, Mar. 2010, doi: 10.1016/j.engappai.2009.09.001.

[108] K. R. Opara and J. Arabas, 'Differential Evolution: A survey of theoretical analyses', *Swarm and Evolutionary Computation*, vol. 44, pp. 546–558, Feb. 2019, doi: 10.1016/j.swevo.2018.06.010.

[109] S. Das, S. S. Mullick, and P. N. Suganthan, 'Recent advances in differential evolution – An updated survey', *Swarm and Evolutionary Computation*, vol. 27, pp. 1–30, Apr. 2016, doi: 10.1016/j.swevo.2016.01.004.

[110] N. Padhye, P. Mittal, and K. Deb, 'Differential evolution: Performances and analyses', in *2013 IEEE Congress on Evolutionary Computation*, Cancun, Mexico, Jun. 2013, pp. 1960–1967. doi: 10.1109/CEC.2013.6557799.

[111] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, 'Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems', *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 646–657, Dec. 2006, doi: 10.1109/TEVC.2006.872133.

[112] A. K. Qin, V. L. Huang, and P. N. Suganthan, 'Differential Evolution Algorithm With Strategy Adaptation for Global Numerical Optimization', *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 398–417, Apr. 2009, doi: 10.1109/TEVC.2008.927706.

[113] Q. Zhang and H. Li, 'MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition', *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, Dec. 2007, doi: 10.1109/TEVC.2007.892759.

[114] H. Li and Q. Zhang, 'Multiobjective Optimization Problems With Complicated Pareto Sets, MOEA/D and NSGA-II', *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 284–302, Apr. 2009, doi: 10.1109/TEVC.2008.925798.

[115] 'Find Pareto front of multiple fitness functions using genetic algorithm - MATLAB gamultiobj'. https://www.mathworks.com/help/releases/R2021a/gads/gamultiobj.html (accessed Jun. 08, 2021).

[116] E. Sandgren, 'Nonlinear Integer and Discrete Programming in Mechanical Design Optimization', *Journal of Mechanical Design*, vol. 112, no. 2, pp. 223–229, Jun. 1990, doi: 10.1115/1.2912596.

[117] B. Kannan and S. Kramer, 'An Augmented Lagrange Multiplier Based Method for Mixed Integer Discrete Continuous Optimization and Its Applications to Mechanical Design', 1994, doi: 10.1115/1.2919393.

[118] Y. J. Cao and Wu. Q.H., 'Mechanical design optimization by mixed variable evolutionary programming.', Indianapolis, 1997, pp. 443-446.

[119] K. Deb, 'GeneAS: A Robust Optimal Design Technique for Mechanical Component Design', in *Evolutionary Algorithms in Engineering Applications*, D. Dasgupta and Z. Michalewicz, Eds. Berlin, Heidelberg: Springer, 1997, pp. 497–514. doi: 10.1007/978-3-662-03423-1_27.

[120] C. A. C. Coello, 'Self-adaptive penalties for GA-based optimization', in *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, Jul. 1999, vol. 1, pp. 573-580 Vol. 1. doi: 10.1109/CEC.1999.781984.

[121] J. W. Stokes, L. Holly, and K. H. Mayer, 'The ASME boiler and pressure vessel code', *Non-Destructive Testing*, vol. 7, no. 3, pp. 145–151, Jun. 1974, doi: 10.1016/0029-1021(74)90077-2.

[122] S. He, E. Prempain, and Q. Wu, 'An improved particle swarm optimizer for mechanical design optimization problems', *Engineering Optimization - ENG OPTIMIZ*, vol. 36, pp. 585–605, Oct. 2004, doi: 10.1080/03052150410001704854.
[123] A.-R. Hedar and M. Fukushima, 'Derivative-Free Filter Simulated Annealing Method for Constrained Continuous Global Optimization', *J Glob Optim*, vol. 35, no. 4, pp. 521–549, Aug. 2006, doi: 10.1007/s10898-005-3693-z.
[124] G. G. Dimopoulos, 'Mixed-variable engineering optimization based on evolutionary and social metaphors', *Computer Methods in Applied Mechanics and Engineering*, vol. 196, no. 4, pp. 803–817, Jan. 2007, doi: 10.1016/j.cma.2006.06.010.
[125] A. H. Gandomi, X.-S. Yang, and A. H. Alavi, 'Mixed variable structural optimization using Firefly Algorithm', *Computers & Structures*, vol. 89, no. 23, pp. 2325–2336, Dec. 2011, doi: 10.1016/j.compstruc.2011.08.002.
[126] B. Akay and D. Karaboga, 'Artificial bee colony algorithm for large-scale problems and engineering design optimization', *J Intell Manuf*, vol. 23, no. 4, pp. 1001–1014, Aug. 2012, doi: 10.1007/s10845-010-0393-4.
[127] L. Cagnina, S. Esquivel, and C. Coello, 'Solving Engineering Optimization Problems with the Simple Constrained Particle Swarm Optimizer', *Informatica (Slovenia)*, vol. 32, pp. 319–326, Jan. 2008.
[128] L. dos S. Coelho, 'Gaussian quantum-behaved particle swarm optimization approaches for constrained engineering design problems', *Expert Systems with Applications*, vol. 37, no. 2, pp. 1676–1683, Mar. 2010, doi: 10.1016/j.eswa.2009.06.044.
[129] J. Golinski, 'An adaptive optimization system applied to machine synthesis', *Mechanism and Machine Theory*, vol. 8, no. 4, pp. 419–436, Dec. 1973, doi: 10.1016/0094-114X(73)90018-9.
[130] K. Deb, 'Optimal design of a welded beam via genetic algorithms', *AIAA Journal*, vol. 29, no. 11, pp. 2013–2015, Nov. 1991, doi: 10.2514/3.10834.
[131] K. M. Ragsdell and D. T. Phillips, 'Optimal Design of a Class of Welded Structures Using Geometric Programming', *Journal of Engineering for Industry*, vol. 98, no. 3, pp. 1021–1025, Aug. 1976, doi: 10.1115/1.3438995.
[132] J. N. Siddall, *Analytical decision-making in engineering design*. Englewood Cliffs, N.J: Prentice-Hall, 1972.
[133] S. Akhtar, K. Tai, and T. Ray, 'A socio-behavioural simulation model for engineering design optimization', *Engineering Optimization*, vol. 34, no. 4, pp. 341–354, Jan. 2002, doi: 10.1080/03052150212723.
[134] S. Akhtar, K. Tai, and T. Ray, 'A socio-behavioural simulation model for engineering design optimization', *Engineering Optimization*, vol. 34, no. 4, pp. 341–354, Jan. 2002, doi: 10.1080/03052150212723.
[135] T.-H. Kim, M. Cho, and S. Shin, 'Constrained Mixed-Variable Design Optimization Based on Particle Swarm Optimizer with a Diversity Classifier for Cyclically Neighboring Subpopulations', *Mathematics*, vol. 8, no. 11, p. 2016, Nov. 2020, doi: 10.3390/math8112016.
[136] P. B. Thanedar and G. N. Vanderplaats, 'Survey of Discrete Variable Optimization for Structural Design', *Journal of Structural Engineering*, vol. 121, no. 2, pp. 301–306, Feb. 1995, doi: 10.1061/(ASCE)0733-9445(1995)121:2(301).
[137] A. H. Gandomi and X.-S. Yang, 'Benchmark Problems in Structural Optimization', in *Computational Optimization, Methods and Algorithms*, S. Koziel and X.-S. Yang, Eds. Berlin, Heidelberg: Springer, 2011, pp. 259–281. doi: 10.1007/978-3-642-20859-1_12.
[138] F. Erbatur, O. Hasançebi, İ. Tütüncü, and H. Kılıç, 'Optimal design of planar and space structures with genetic algorithms', *Computers & Structures*, vol. 75, no. 2, pp. 209–224, Mar. 2000, doi: 10.1016/S0045-7949(99)00084-X.

[139] M. R. Ghasemi, E. Hinton, and R. D. Wood, 'Optimization of trusses using genetic algorithms for discrete and continuous variables', *Engineering Computations*, vol. 16, no. 3, pp. 272–303, Jan. 1999, doi: 10.1108/02644409910266403.
[140] L. A. Schmit and H. Miura, 'A New Structural Analysis/Synthesis Capability-ACCESS 1', *AIAA Journal*, vol. 14, no. 5, pp. 661–671, May 1976, doi: 10.2514/3.61405.
[141] T. Y. Chen and H. C. Chen, 'Mixed–discrete structural optimization using a rank-niche evolution strategy', *Engineering Optimization*, vol. 41, no. 1, pp. 39–58, Jan. 2009, doi: 10.1080/03052150802344535.
[142] L. Lamberti and C. Pappalettere, 'Move limits definition in structural optimization with sequential linear programming. Part II: Numerical examples', *Computers & Structures*, vol. 81, no. 4, pp. 215–238, Mar. 2003, doi: 10.1016/S0045-7949(02)00443-1.
[143] P. B. Thanedar and G. N. Vanderplaats, 'Survey of Discrete Variable Optimization for Structural Design', *Journal of Structural Engineering*, vol. 121, no. 2, pp. 301–306, Feb. 1995, doi: 10.1061/(ASCE)0733-9445(1995)121:2(301).
[144] X. B. Lam, Y. S. Kim, A. D. Hoang, and C. W. Park, 'Coupled Aerostructural Design Optimization Using the Kriging Model and Integrated Multiobjective Optimization Algorithm', *J Optim Theory Appl*, vol. 142, no. 3, pp. 533–556, Sep. 2009, doi: 10.1007/s10957-009-9520-9.
[145] A. Ceruti, G. Caligiana, and F. Persiani, 'Comparative evaluation of different optimization methodologies for the design of UAVs having shape obtained by hot wire cutting techniques', *International Journal on Interactive Design and Manufacturing (IJIDeM)*, vol. 7, no. 2, pp. 63–78, May 2013, doi: 10.1007/s12008-012-0164-x.
[146] M. Berci, V. V. Toropov, R. W. Hewson, and P. H. Gaskell, 'Multidisciplinary multifidelity optimisation of a flexible wing aerofoil with reference to a small UAV', *Struct Multidisc Optim*, vol. 50, no. 4, pp. 683–699, Oct. 2014, doi: 10.1007/s00158-014-1066-2.
[147] A. Długosz and W. Klimek, 'The optimal design of UAV wing structure', *AIP Conference Proceedings*, vol. 1922, no. 1, p. 120009, Jan. 2018, doi: 10.1063/1.5019124.
[148] T. Schlieter and A. Długosz, 'Structural Optimal Design of an Airfoil for Many Criteria', *41st Solid Mechanics Conference*, pp. 382–383, Aug. 2018.
[149] M. Gad-el-Hak and W. Seemann, 'MEMS handbook', *Applied Mechanics Reviews*, vol. 55, p. 109, Nov. 2002, doi: 10.1115/1.1508147.
[150] J. J. Brown and V. M. Bright, 'Thermal Actuators', in *Encyclopedia of Nanotechnology*, B. Bhushan, Ed. Dordrecht: Springer Netherlands, 2016, pp. 4117–4138. doi: 10.1007/978-94-017-9780-1_313.
[151] L. T. Bui and S. Alam, Eds., *Multi-objective optimization in computational intelligence: theory and practice*. Hershey: Information Science Reference, 2008.
[152] C.-C. Lee and W. Hsu, 'Optimization of an electro-thermally and laterally driven microactuator', *Microsystem Technologies*, vol. 9, no. 5, pp. 331–334, May 2003, doi: 10.1007/s00542-002-0253-z.
[153] Q.-A. Huang and N. K. S. Lee, 'Analytical modeling and optimization for a laterally-driven polysilicon thermal actuator', *Microsystem Technologies*, vol. 5, no. 3, pp. 133–137, Feb. 1999, doi: 10.1007/s005420050152.
[154] V. Ionescu, 'Numerical Investigation of a MEMS Thermal Actuator Performance by Modifying its Geometric Dimensions', *Procedia Manufacturing*, vol. 32, pp. 820–830, Jan. 2019, doi: 10.1016/j.promfg.2019.02.290.
[155] A. M. H. Kwan *et al.*, 'Improved Designs for an Electrothermal In-Plane Microactuator', *Journal of Microelectromechanical Systems*, vol. 21, no. 3, pp. 586–595, Jun. 2012, doi: 10.1109/JMEMS.2012.2185820.

[156] P. Shivhare, G. Uma, and M. Umapathy, 'Design enhancement of a chevron electrothermally actuated microgripper for improved gripping performance', *Microsyst Technol*, vol. 22, no. 11, pp. 2623–2631, Nov. 2016, doi: 10.1007/s00542-015-2561-0.
[157] A. Thangavel, R. Rengaswamy, P. K. Sukumar, and R. Sekar, 'Modelling of Chevron electrothermal actuator and its performance analysis', *Microsyst Technol*, vol. 24, no. 4, pp. 1767–1774, Apr. 2018, doi: 10.1007/s00542-018-3791-8.
[158] E. O. Salah and Z. Musaab, 'Optimized V-Shaped Beam Micro-Electrothermal Actuator Using Particle Swarm Optimization (PSO) Technique', *Micro and Nanosystems*, vol. 11, no. 1, pp. 62–67, May 2019.
[159] R. Ansola, E. Veguería, J. Canales, and C. Alonso, 'Electro-thermal compliant mechanisms design by an evolutionary topology optimization method', *Engineering Computations*, vol. 30, no. 7, pp. 961–981, Jan. 2013, doi: 10.1108/EC-12-2011-0150.
[160] S. M. Giusti, Z. Mróz, A. A. Novotny, and J. Sokołowski, 'Topology design of thermomechanical actuators', *Struct Multidisc Optim*, vol. 55, no. 5, pp. 1575–1587, May 2017, doi: 10.1007/s00158-016-1593-0.
[161] O. Sigmund, 'Design of multiphysics actuators using topology optimization – Part I: One-material structures', *Computer Methods in Applied Mechanics and Engineering*, vol. 190, no. 49, pp. 6577–6604, Oct. 2001, doi: 10.1016/S0045-7825(01)00251-1.
[162] K. Furuta, K. Izui, K. Yaji, T. Yamada, and S. Nishiwaki, 'Level set-based topology optimization for the design of a peltier effect thermoelectric actuator', *Struct Multidisc Optim*, vol. 55, no. 5, pp. 1671–1683, May 2017, doi: 10.1007/s00158-016-1609-9.
[163] Q. Xia, L. Xia, and T. Shi, 'Topology optimization of thermal actuator and its support using the level set based multiple–type boundary method and sensitivity analysis based on constrained variational principle', *Struct Multidisc Optim*, vol. 57, no. 3, pp. 1317–1327, Mar. 2018, doi: 10.1007/s00158-017-1814-1.
[164] S. Heo and Y. Y. Kim, 'Optimal design and fabrication of MEMS rotary thermal actuators', *J. Micromech. Microeng.*, vol. 17, no. 11, pp. 2241–2247, Oct. 2007, doi: 10.1088/0960-1317/17/11/010.
[165] P. Di Barba, M. E. Mognaschi, P. Venini, and S. Wiak, 'Biogeography-inspired multiobjective optimization for helping MEMS synthesis', *Archives of Electrical Engineering*, 2017, doi: 10.1515/aee-2017-0046.
[166] H. Chen, X. Wang, Y. Cao, J. Wang, Z. Xi, and W. Nie, 'Double-objective optimization of electro-thermal (E-T) micro-actuator for fiber switch', *J. Micromech. Microeng.*, vol. 31, no. 4, p. 045003, Feb. 2021, doi: 10.1088/1361-6439/abde91.
[167] A. Długosz, 'Multiobjective evolutionary optimization of MEMS structures', *Computer Assisted Methods in Engineering and Science*, vol. 17, no. 1, Art. no. 1, Jan. 2017.
[168] A. Długosz, P. Jarosz, and T. Schlieter, 'Optimal Design of Electrothermal Microactuators for Many Criteria by Means of an Immune Game Theory Multiobjective Algorithm', *Applied Sciences*, vol. 9, no. 21, Art. no. 21, Jan. 2019, doi: 10.3390/app9214654.
[169] Y. Zhu, A. Corigliano, and H. D. Espinosa, 'A thermal actuator for nanoscalein situmicroscopy testing: design and characterization', *J. Micromech. Microeng.*, vol. 16, no. 2, pp. 242–253, Jan. 2006, doi: 10.1088/0960-1317/16/2/008.
[170] J. Fish, 'Bridging the scales in nano engineering and science', *J Nanopart Res*, vol. 8, no. 5, pp. 577–594, Oct. 2006, doi: 10.1007/s11051-006-9090-9.
[171] J.-L. Auriault, C. Boutin, and C. Geindreau, *Homogenization of coupled phenomena in heterogenous media*. London, UK : Hoboken, N.J: ISTE ; J. Wiley, 2009.
[172] J. Ptaszny and M. Hatłas, 'Evaluation of the FMBEM efficiency in the analysis of porous structures', *Engineering Computations*, vol. 35, pp. 00–00, Feb. 2018, doi: 10.1108/EC-12-2016-0436.

[173] V. Buryachenko, *Micromechanics of Heterogeneous Materials*. Springer US, 2007. doi: 10.1007/978-0-387-68485-7.
[174] T. I. Zohdi and P. Wriggers, *An Introduction to Computational Micromechanics*. Berlin Heidelberg: Springer-Verlag, 2005. doi: 10.1007/978-3-540-32360-0.
[175] K. Terada, M. Kurumatani, T. Ushida, and N. Kikuchi, 'A method of two-scale thermo-mechanical analysis for porous solids with micro-scale heat transfer', *Computational Mechanics*, vol. 46, pp. 269–285, Jul. 2010, doi: 10.1007/s00466-009-0400-9.
[176] G. Beer, 'Finite element, boundary element and coupled analysis of unbounded problems in elastostatics', *Int. J. Numer. Meth. Engng.*, vol. 19, no. 4, pp. 567–580, Apr. 1983, doi: 10.1002/nme.1620190408.
[177] J. P. Carter and J. R. Booker, 'Finite element analysis of coupled thermoelasticity', *Computers & Structures*, vol. 31, no. 1, pp. 73–80, Jan. 1989, doi: 10.1016/0045-7949(89)90169-7.
[178] O. C. Zienkiewicz, R. L. Taylor, and J. Z. Zhu, *The finite element method: it's basics and fundamentals*. Amsterdam: Elsevier/Burretwoth Heinemann, 2005.
[179] W. Nowacki, *Thermoelasticity.* Kent: Elsevier Science, 2014. Accessed: Sep. 01, 2021. [Online]. Available: http://qut.eblib.com.au/patron/FullRecord.aspx?p=1828984

ABSTRACT

The dissertation investigates the subject of optimal design of mechanical systems for many criteria. In the literature review basic information on the process of design are presented with special attention given to the optimization tasks. Overview of finite element method as a simulation tool from the perspective of optimization of mechanical systems is included. Metrics and benchmark functions are reviewed in the context of evaluating performance of multiobjective optimization algorithms. Visualisation techniques enhancing post-optimization decision making process are described.

Next part of the dissertation presents developed multiobjective optimization algorithm belonging to a group of soft computing methods. Proposed algorithm is based on a differential evolution and game theory paradigms. Suggested algorithm takes advantage of a game theoretic cooperative approach and eliminates some of drawbacks of other soft computing methods in the optimization of mechanical systems. Elements of game theory are introduced along with basics of differential evolution. General idea a novel multiobjective algorithm is described and presented using a flowchart, a pseudocode and supplemented with an example. Implementation of the algorithm in C++ programming language is briefly described. Methods of information exchange between developed algorithm and FEM software responsible for numerically obtaining values of objective functions during the course of optimization are discussed.

In the subsequent part of the dissertation, the algorithm is comprehensively tested using previously introduced benchmark functions and performance metrics. A set of mathematical test functions exhibiting features distinctive of mechanical systems is utilised. Quality of results is assessed using a hypervolume indicator. Mean and standard deviation of metrics are calculated over 30 runs of algorithm and compared with results obtained by other multiobjective optimization algorithms: NSGA-II and NSGA-III. Test problems with a varied number of design variables (up to 20) and objective functions (up to 8) are investigated. Within analysed framework the developed algorithm was proven to be efficient and competitive with compared algorithms.

Ensuing part of the dissertation provides examples on application of the developed tool in the optimal design of mechanical systems. A set of three analytical problems: pressure vessel design, speed reducer design and stepped cantilever beam are examined. These analytical problems are produced by a transformation of constrained single-objective optimization problems into multiobjective problems. Furthermore, the algorithm is used to solve complex, numerical problems: airfoil optimization, electrothermal microactuators optimization and multiscale porous material optimization. In these problems, values of objective function are obtained by means of FEM based on parametric models. For each of analysed problems, algorithm found a set of new diverse designs. Results were presented in the form of tables, graphs, and figures. On the basis of additional established preferences and using proposed visualisation tools, a post-optimization decision making process was aided resulting in a narrowed down set of solutions.

STRESZCZENIE

W rozprawie podjęto tematykę optymalnego projektowania układów mechanicznych z uwzględnieniem wielu kryteriów. W przeglądzie literatury przedstawiono podstawowe informacje na temat procesu projektowania ze szczególnym uwzględnieniem zadań optymalizacyjnych. Dokonano przeglądu metody elementów skończonych jako narzędzia symulacyjnego z punktu widzenia optymalizacji układów mechanicznych. Dokonano przeglądu metryk i funkcji testowych w kontekście oceny wydajności algorytmów optymalizacji. Opisano techniki wizualizacji usprawniające proces podejmowania decyzji po optymalizacji.

W następnej części rozprawy przedstawiono opracowany algorytm optymalizacji wielokryterialnej należący do grupy metod obliczeń miękkich. Proponowany algorytm oparty jest na ewolucji różnicowej oraz elementach teorii gier. Proponowany algorytm wykorzystuje kooperacyjne podejście teorii gier i eliminuje niektóre wady innych metod obliczeń miękkich. Omówiony podstawy teorii gier i ewolucji różnicowej. Ogólna idea zaproponowanego algorytmu została opisana i przedstawiona za pomocą schematu blokowego, pseudokodu oraz uzupełniona przykładem. Krótko opisano implementację algorytmu w języku programowania C++. Omówiono metody wymiany informacji pomiędzy opracowanym algorytmem a oprogramowaniem MES odpowiedzialnym za numeryczne wyznaczanie wartości funkcji celu w trakcie optymalizacji.

W dalszej części rozprawy algorytm jest kompleksowo przetestowany z wykorzystaniem opisanych wcześniej funkcji testowych oraz metryk. Wykorzystywany jest zestaw matematycznych funkcji testowych wykazujących cechy charakterystyczne dla układów mechanicznych. Jakość wyników oceniana jest za pomocą wskaźnika hiperobjętości. Średnia i odchylenie standardowe metryk są obliczane dla 30 przebiegów algorytmu i porównywane z wynikami uzyskanymi przez inne algorytmy optymalizacji wielokryterialnej: NSGA-II i NSGA-III. Badane są problemy testowe o różnej liczbie zmiennych projektowych (do 20) i funkcji celu (do 8). Udowodniono skuteczność i konkurencyjność zaproponowanego algorytmu.

W dalszej części rozprawy przedstawiono przykłady zastosowania opracowanego narzędzia w optymalnym projektowaniu układów mechanicznych. Analizie poddano zestaw trzech problemów analitycznych: projekt zbiornika ciśnieniowego, projekt przekładni oraz projekt belki wspornikowej o zmiennym przekroju. Problemy te powstały w wyniku transformacji jednokryterialnych zadań optymalizacyjnych z ograniczeniami w zadania wielokryterialne. Ponadto, algorytm jest użyty do rozwiązywania złożonych zadań numerycznych: optymalizacji profilu lotniczego, optymalizacji mikroaktuatorów elektrotermicznych oraz optymalizacji wieloskalowych materiałów porowatych. W tych zadaniach wartości funkcji celu są uzyskiwane za pomocą metody elementów skończonych w oparciu o modele parametryczne. Dla każdego z analizowanych problemów algorytm znalazł zbiór nowych, zróżnicowanych rozwiązań. Wyniki przedstawiono w postaci tabel, wykresów i rysunków. Na podstawie dodatkowo ustalonych preferencji oraz przy wykorzystaniu zaproponowanych narzędzi wizualizacyjnych wspomagano proces podejmowania decyzji pooptymalizacyjnych, w wyniku którego otrzymano zawężony zbiór rozwiązań.