DOI 10.1515/ama-2017-0024

PRESENT-DAY PROBLEMS AND METHODS OF OPTIMIZATION IN MECHATRONICS

Wojciech TARNOWSKI[*]

[*]Koszalin University of Technology, ul. Śniadeckich 2, 75-453 Koszalin, Poland

wojciech.tarnowski@tu.koszalin.pl

received 10 October 2016, revised 1 June 2017, accepted 2 June 2017

Abstract: It is justified that design is an inverse problem, and the optimization is a paradigm. Classes of design problems are proposed and typical obstacles are recognized. Peculiarities of the mechatronic designing are specified as a proof of a particle importance of optimization in the mechatronic design. Two main obstacles of optimization are discussed: a complexity of mathematical models and an uncertainty of the value system, in concrete case. Then a set of non-standard approaches and methods are presented and discussed, illustrated by examples: a fuzzy description, a constraint-based iterative optimization, AHP ranking method and a few MADM functions in Matlab.

Key words: Mechatronics, Optimization, Poly-optimization, Evolutionary Algorithms, Artificial Intelligence, Genetic Algorithm, Soft Optimization, Multi-Attribute Decision Making (MADM), Inverse Problem, Engineering Design, Fuzzy Control, Linguistic Criteria, Mathematical Models, Pareto Solutions, Trade-Off Solutions, Compromise Solutions, Fuzzy Logic

1. INTRODUCTION

As for now, optimization techniques are not very popular in the real engineering, though they are taught at universities and are recognized as very useful in design and in management. Probably the main reason is that mathematical models of objects are difficult to determine, and what more, criteria may not be precisely defined.

The goal of this paper is to demonstrate a special significance of optimization in mechatronic design, also the characteristic obstacles to formulate and solve solutions, and to present typical modern approaches and methods.

Also, the goals are: (1) to argue that the design task mostly is an inverse problem and, if so then (2): to propose that the poly-optimization approach is an efficient methodology to solve it. What is more, it may be a sound paradigm for devising CAD programs.

2. STATEMENT OF THE OPTIMIZATION PROBLEM

2.1. Particularities of Mechatronics as a Design Domain

There are few peculiarities of the mechatronic design among other engineering domains. First, it has a shorter design tradition, so the designer's experience is confined, as well as official standards for designing are currently limited. Second, a few science disciplines and engineering fields are engaged: mechanical, control, electronics etc. Third, mathematical models are complex. Thus, to effectively precede a design process a formal mathematical analysis and effective computer procedures are necessary. The optimization is a key action.

2.2. Notions

The following terms will be used:

- *decision variables*: parameters that are to be calculated in (poly-)optimization procedure;
- *variant*: a specific vector of decision variables; or a specific vector in performance parameters space;
- *constraints*: limits imposed on admissible variant decision variables and on performance parameters; they define the set of admissible variants;
- *criteria*: performance parameters, which must be at minimum or at maximum;
- *optimization*, optimality problem: searching for optimal decision variables;
- *solution of the optimality problem*: the optimal variant;
- *compromise variant* (equivalent names: a trade-off variant, Pareto-optimal variant; a poly-optimal variant): such permitted variant, which may not be improved on one criterion without violation on another criterion;
- *solution of the poly-optimization* (MADM, MCDM): set of compromise variants.

2.3. General Structure of the Design Process

Examples from the real life acknowledge that the design process is strongly iterative. This concerns the micro–scale as well as the macro–scale of the process. It is due to the fact that design tasks are inverse problems, because either the direct problem in a specific design case may not be formulated or there are more unknown parameters than the number of equations in a mathematical model.

Multiple repetitions of a design procedure are time– and labour–consuming, so in the design science methodological efforts are undertaken to convert the iterative structure to the linear one. In the next paragraphs some procedures are discussed.

For planning and management purposes, and for realization of the design process it is broken to smaller actions, which constitute the process structure. If the actions are elementary, the structure is called a micro-structure (Tarnowski, 1997). It is presented graphically in Fig. 1.

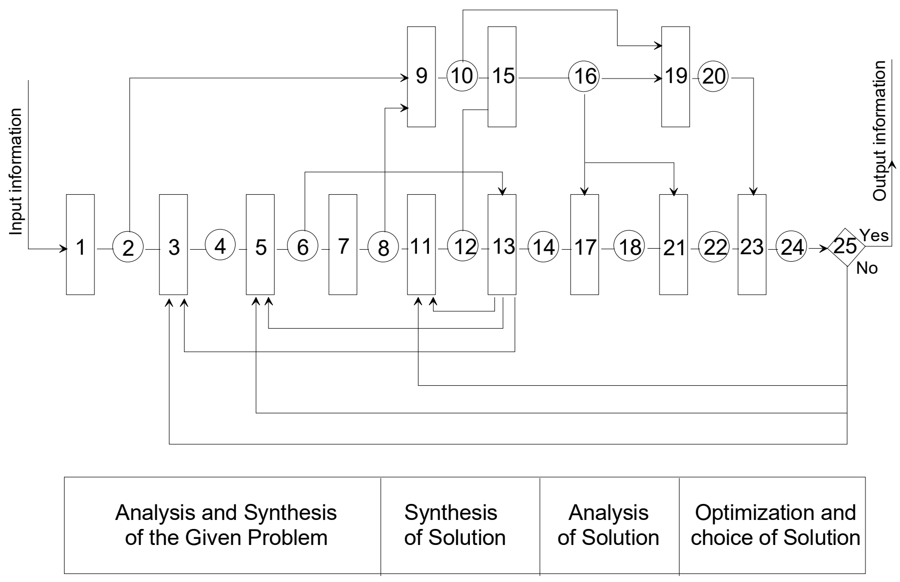

Fig. 1. The micro-structure of the design process (Tarnowski, 1997): 1 – analysis of the need; 3 – definition of the design problem; 5 – specifications; 7 – overall quality system; 9 – definition of the overall criterion of optimality; 11 – searching for possible variants of solution; 13 – selection of variants; 15 – definition of the quality measures; 17 – analysis and verification of feasibility of variants; 19 – definition of the criteria of optimality; 21 – evaluation; 23 – optimization or poly-optimization or choice proposal of a variant to the design processing in the next step; 25 – arbitrary decision. Rectangles denote elementary actions, circles denote results

You may see that few different back loops are possible from the action 13 and from the final decision 25, also. These loops – if they are formalized as problems – are inverse problems.

2.4. Decision Problem

The standard optimality problem is when the triad is given: $< x, \Phi(x), \Omega(x) >$, where x is the decision variables vector, $\Phi(x)$ is an overall optimality criterion and $\Omega(x)$ is a set of constraints, defined in a fully mathematical form. If not, the formal optimization is not possible, as it is often practically the case, but still people try to find the best solution, and they do it intuitively.

Another trouble may be, that even if the complete model of optimization is given or may be easily formulated, the computations are difficult (like in the case of the dynamic optimization) or time consuming (with plenty of decision variables – e.g. in design of machines or electronic devices) or the optimized object mathematical model is continuous in time and space.

It is generally accepted that each decision should be optimal. But in the real practice in business, logistics, medicine, and even in engineering – where a decision–maker has for his/her disposal comparatively good unique mathematical models – it is very rarely the case. Various reasons of this situation were discussed in many publications (e.g. Tarnowski (2011))

A recognized cause is the lack of the adequate optimality criterion definition for a specific case or even no practical possibility to formulate it as a mathematical function of decision variables (like cost, reliability and others). Besides, in many concrete situations people do not consent which performance parameter(s) should be minimal (or maximal), or which are only limit constraints. But still they make the decisions intuitively and they insist their resolutions are optimal or quasi–optimal, at least.

2.5. Characteristics of a Design Parametric Problem

Here are three main features of the design parametric problem.

- number of design parameters X (which are to be found) strongly prevail over the number of performance parameters Y (which are given);
- formal mathematical model only partly define relations between X and Y (in another words it is not complete); remaining relations are the matter of intuition and experience of designer (they are heuristic);

In practical cases the model is defined as $\mathrm{Y} = \mathrm{f(X)}$ and it may not be converted to $\mathrm{X} = \mathrm{f^{-1}(Y)}$

Design tasks generally are inverse problems.

2.6. Standard Inverse Design Problems in Engineering (Tarnowski et al., 2009)

The considerations in this work are confined to the parametric design rather than to the conceptual design. So it may be assumed that at the start of the design process a general idea (or a structure of the object) is defined, as well as objectives and requirements (or specifications) of the object. Hence the mathematical model may be elaborated, even though it is usually non-complete in the sense, that there are more unknown parameters then equations. What is more, we assume that constraints are not to restrictive, what means that the set of admissible variants is not empty.

2.7. Procedure of the Inverse Problem Solution (Tarnowski et al., 2009)

To solve a given problem – especially by an algorithm – we try to find formal relations between the variables. Then we can set some constraints and define a mathematical model of the given

DOI 10.1515/ama-2017-0024

task. In engineering the models are built on the basis of physics and economics.

Let the mathematical model that may be formulated has the form:

$$Y = f(X) \tag{1}$$

In design Y is a vector of performance variables or performance functions, and X stands for design parameters. Symbol f denotes a functional relation. We assume, that the mathematical model is known (or may be elaborated).

A direct problem is when X is given and Y is to be determined (by computing the model, as a standard way).

However, in the parametric design and in diagnostics Y is given and X is to be found, and typically it is not possible to invert the model (Mainly for two reasons: (1) there are more unknowns X than givens Y, (2) in engineering most relations are of experimental origin and cannot be inverted – like in the stress-strain analysis, fluid mechanics, thermodynamics and others.)

$$X = f^{-1}(Y) \tag{1a}$$

so a designer must solve an inverse problem (In publications the term inverse problem is used in narrower sense: usually refers to situations, where solutions are defined in a continuous space (usually 2D or 3D), e.g. the mathematical model is built of partial differential equations – like in the pattern recognition problem.).

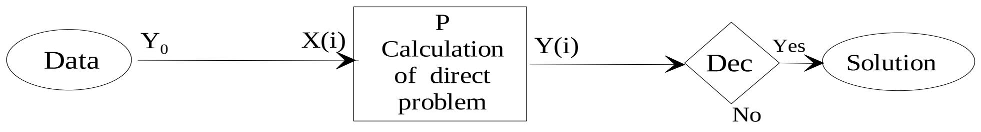

Fig. 2. General structure of the procedure of solving the direct problem

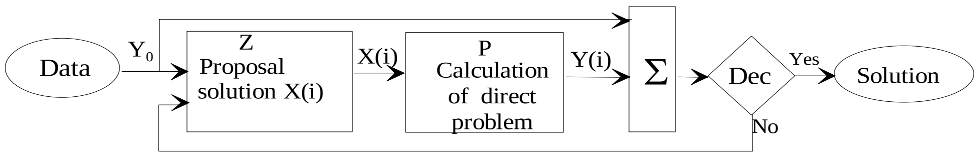

Fig. 3. General structure of the procedure of solving the inverse problem

The direct problem is being solved by one 'pass', as it is shown on the flow chart on Fig. 2, to be contrasted with the flow chart on Fig. 3 which maps the solving procedure of the inverse problem. Accordingly to Eqn. (1a), now Y is given, which is here denoted as $\mathrm{Y}(0)$. The X is to be found. The first activity (block Z) is to set a hypothesis about a solution X, i.e. to propose a preliminary solution $\mathrm{X(i)}$ (symbol i stands for the iteration number.). This may be a heuristic activity, completed by a human being based on his/her knowledge and/or intuition (even a guess). Or – as the opposed case – it may be fully algorithmic, aided by a computer (random sampling from a given set, for example). Typical is an optimization, by the gradient method, for example. In the last decade the Artificial Intelligence Methods are being developed for this purpose, like the Genetic Algorithms or Artificial Neural Algorithms.

The second activity (block P) is a computation of a vector $\mathrm{Y(i)}$ for the current vector $\mathrm{X(i)}$, on the known mathematical model (1), or – in another words – it is solving a direct problem.

The subsequent activity (the block denoted by Σ) is a comparison of the obtained vector $\mathrm{Y(i)}$ with the given vector $\mathrm{Y(0)}$. For example, it may be computation of the optimality function F, as it is in classical optimization algorithms, or it may be a calculation of the fitness function like in Genetic Algorithms. In a general case it may be a quadratic deviation:

$$F(i) = \sum_j \big(Y(j,i) - Y(j,0)\big)^2, \tag{2}$$

where $\mathrm{F(i)}$ is a total mean square deviation in i-th step, j is the index of a component of the vector Y.

The last block is a decision; and three decisions are possible: NO – means that the next trial pass $\mathrm{X(i+1)}$ is necessary, YES – that $\mathrm{X(i)}$ is acceptable as the solution of the inverse problem, and STOP – that the further continuation of the loop is aimless, what means that no acceptable solution can be found.

Further in this paper it will be proposed and proven, that the poly-optimization can be an efficient and the adequate approach to solve the inverse problems.

In engineering, inverse problems are formulated in design and in diagnosing.

2.8. Objects of Design

In engineering design it is possible to recognize three main groups of objects of design:

- design of (static) objects, like machines, buildings etc., and processes in steady states (for example a schedule of manufacturing process, like an assembly line, the busses network timetable etc.); in this case the mathematical model consists of algebraic equations and inequalities, sometimes it contains tables and diagrams, of a non–analytical type;
- design of dynamic objects: especially processes of the transient states of objects (for example the control in the presence of disturbances, the control of the set–up process of installations etc.); in this case the mathematical model contains, as a rule, non–linear differential equations, often partial differential equations, and what is more, some variables are time dependent;
- simultaneously design of a process and a machine to realize this process; what is the most complex problem, with great many variables.

2.9. Typical Optimality Problems in Design

— **Parametric design:** a design idea is given, a great number of continues design parameters are to be found, a few are discrete, some are symbolic (names). There are numerous constraints, many are nonlinear, some are fuzzy. Usually there are a few well defined but conflicted criteria of optimality. Some may be only estimated (like a cost). Typically there is an unlimited set of variants, defined by constraints. Example: a gear transmission of a given design with an electric motor from a catalogue.
— **Conceptual design:** typically there is a few or dozen of design concepts, given as verbal outlines or drafts in a matrix form of fuzzy performance parameters. Criteria and constraints are quantities or fuzzy and they define a fuzzy value system, and a set of constraints define a space of admissible solutions. Ranking of the concepts is to be presented to the designer for his final decision. Probably the effective decision aiding tool is AHP method.
— **Process:** functions in the time-space domain are to be found, like control and or shape functions. Examples: control function of a drive, or a shape of a pump rotor wing, a profile of a turbine rotor leaf. This is the case of the dynamic programming problem.
— **Design and Process:** Combination of design and process in one common optimality problem. Examples: design of control system and a control function of the controller to get the highest efficiency and minimal cost of the process, or to find a profile of a control valve piston and the control signal function of the actuator to obtain a required break force of a car in a specific traffic situation. Another example is a flying object and its control function.
— **Scheduling, deployment of resources:** A discrete problem with a great number of binary parameters (decision variables) and a great number of constraints (usually linear). A Cartesian product is a set of variants (for example schedules or resources). No mathematical analytical model is given. Sequence or combination of events is to be found, there are many simple constraints; example: schedule, procedure or plan of actions. Permutation is possible to generate variants. This case is not typical in mechatronic design. Examples: a schedule, a procedure, a route. To find the best solution the discrete linear programming class methods are used, for example the Branch and Bound (B@B) procedures are applied.

2.10. Main Obstacles in Optimization

There are two main problems (difficulties): one is a question of the mathematical formulation of the optimality problem and the other is a formal searching for a minimum of the optimality function. The value system is assumed by a decision maker, typically in a descriptive verbal form.

2.11. Mathematical Formulation of the Optimality Problem

First it is the criterion of optimality and constraints, all together these are a formal definition of the requirements.

Typical troubles are:
— some decision variables are fuzzy (not defined) like a color or design shape;
— some constraints are fuzzy (like compatibility, safety, aesthetics, or weakly defined (like reliability);
— both: some parameters and some values are fuzzy;
— it is difficult to accurately formalize some physical or economical relations – then an expert may define them in fuzzy terms;
— value system (criteria and preferences) may not be uniformly precisely defined as the unique optimality function.

2.12. Value System

The value system is a structure of criteria and constraints.

It is a standard situation, that there are few criteria of optimization, and they are in conflict (like the accuracy and velocity), and there is no clarity what in a specific design case is a criterion and what is a constraint (like energy consumption, effectiveness etc. etc). If a decision maker may not arbitrarily state what should be what, then he/she may:
— if a mathematical analytical model of criteria and constraints is available – poly-optimization or MADM techniques or using one artificial complex criterion as an artificial arbitrary function of criteria;
— if criteria are not properly defined as functions of decision variables – fuzzy;
— if one overall criterion may not be defined – soft constrained based optimization;
— if a discrete set of alternatives is given - complete the ranking of alternatives.

3. SELECTED OPTIMIZATION METHODS

3.1. Fuzzy Optimization (Tarnowski, 2011; 2015)

In real decision situations, objectives (i.e. criteria and constraints) may not be well defined (like aesthetics, effectiveness or safety) or are defined but may have no accurate mathematical models (like costs, production effectiveness or reliability). Still decision makers need procedures to aid their decisions.

A way to enhance an effective implementation of optimization and MADM techniques to a real practice is a fuzzy logic approach to modeling the optimization problem, if only an expert knowledge is at hand. It yields: a) avoiding the laborious and often not possible process of building an analytical model, b) makes possible to use fuzzy and imprecise notions and aspects, c) often considerably accelerate computations.

In Tarnowski (2015) it was proposed a procedure how to device and to handle such models in the MATLAB environment to get a Pareto set solutions, by the poly-optimization. The methodology is illustrated below on the example of a chemical reactor.

3.1.1. Example 1: Poly-Optimization With Two Fuzzy Criteria

The poly-optimal control parameters are to be found. First, having stated criteria, as fuzzy and/or non-fuzzy notions (product quality and effectiveness, in the example), an expert arbitrarily defines decision variables (process temperature, time and mixing velocity), and their membership functions, then a mathematical model is established as a set of rules if – then type. The given

DOI 10.1515/ama-2017-0024

object is a bio-chemical batch process executed in a heated container which is provided with a rotating mixer. What we are looking for are: temperature, duration time of the process and the mixer velocity. These are decision variables. Their optimal values are to be found.

Constraints are defined as arbitrarily ascribed admitted ranges of decision variables; for example:
— temperature: [50, 150] Centigrade,
— time (process duration) [1, 5] hrs,
— mixer velocity: [1, 10] rpm.

For each decision variable a discrete set of linguistic values have been assigned, for example for temperature it is: [low, medium, high]. Then, for each value a membership function has been defined, arbitrarily, be an expert.

There are chosen two criteria: quality of a product and effectiveness of the process. They are in conflict. Obviously they mainly are dependent on decision variables. In specific cases they both may be arbitrarily defined and measured, but in the present example we take them as linguistic terms and formalize as fuzzy notions. Consequently, it must be assumed that they are intuitively understand by an expert/decision maker and he/she is able to define a mathematical model of the process in a form of rules: IF … THEN … (Yager, 1993).

For example, rule no 2 is: IF temperature is low AND process duration is short AND mixer velocity is slow THEN effectiveness is medium AND quality of the product is “1”. Mark “1” is here a score of the quality in a arbitrarily scale 0 – 5. In the example twelve rules has been defined.

To each rule an importance factor may be assigned. On these data the fuzzy computer model has been built in Fuzzy Logic Toolbox within the MATLAB environment. A graphic interface is provided to do this, and a file of the type *.fis is created. Now, the poly-optimization [Multiple Criteria Decision Making] may be completed. Generally there are two approaches (Tarnowski, 2011):
— either by a survey of all possible solutions (Fig. 4) and reducing the set to the Pareto solutions (Figs. 5 - 7),
— or by a mathematical optimization algorithm.

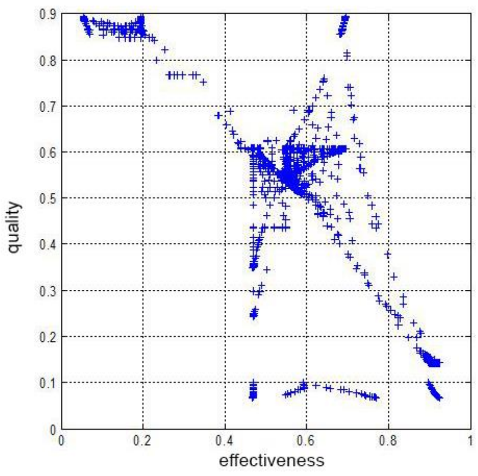

Fig. 4. All 64821 variants generated by the fuzzy system

The first method may be time-consuming, especially if there are many decision variables, but it may be accelerated by random generation of values. The second method may be realized by any known optimization method, for example the gradient method (Venkataraman, 2009) or the genetic algorithm.

In the example the first approach was applied, because this procedure does not impose any restrictions on the mathematical character of the model (it may be non-linear, incoherent, non-monotonous etc.). Also, it may be expected to have some local minima.

To conduct the first method a segmentation of the decision variables space was made for arbitrarily $(n_1 = 251)\ x\ (n_2 = 31)\ x\ (n_3 = 41)$ elements, obtaining the set R of all 64.821 nodes representing admissible solutions.

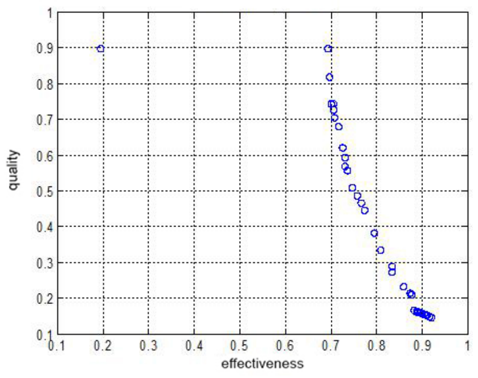

Fig. 5. Collection of 41 Pareto-optimal solutions in the criteria space: effectiveness and quality

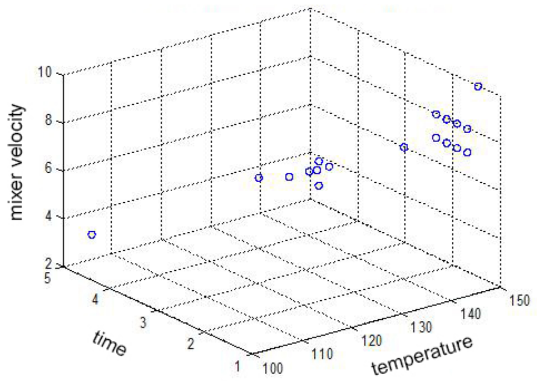

Fig. 6. Collection RPar of 41 Pareto-optimal solutions in the decision variables space: temperature within [50, 150] Centigrade, time [1, 5] hrs and mixer velocity [2, 10] rpm

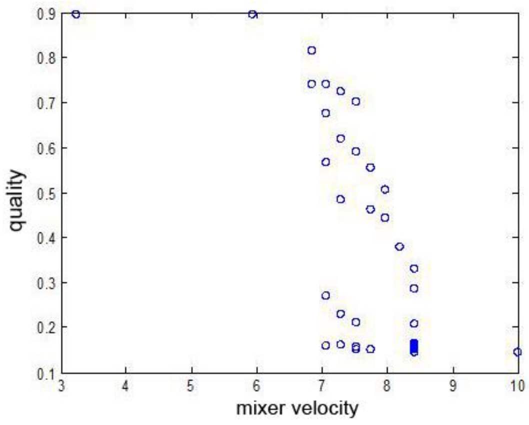

Fig. 7. Collection *RPar* of 41 Pareto-optimal solution: quality vs. velocity (example)

DOI 10.1515/ama-2017-0024

For each node of the grid the two criteria were calculated as fuzzy numbers, using the rules of fuzzy (linguistic) model. After de-fuzzyfication the set R was calculated as numbers. Finally the set R of all 64.821 permissible solutions may be plotted, as on Fig. 4.

Now the Pareto subset $RPar$ of the poly-optimal solutions is to be extracted of R on its definition (Tarnowski, 2011). 41 Pareto-optimal solutions were obtained (Figs. 5 – 7).

The decision maker has to choose one final solution of the RPar set. This intuitive compromise decision may be easier if the decision maker has an 'insight' into this set in various coordinates, especially as a function of decision variables, like Fig. 6 or 7, for example. For easier evaluation the 3D drawings may be rotated.

3.1.2. Conclusions On Fuzzy Poly-Optimization

Basic advantages of the fuzzy poly-optimization are:
- linguistic criteria and linguistic constraints may be included to the mathematical model of optimization and multi-attribute optimization, only if an expert understands their linguistic meanings,
- criteria must not be mutually independent (i.e. they may be mutually correlated),
- relations in the mathematical model may be nonlinear,
- the set of variants may be non-coherent.

Also, in the MATLAB environment:
- fuzzy formalization of the poly(-optimization) model is easy and computations are simple for the user;
- intuitive evaluation of the Pareto set is supported by the extensive graphic presentation capacity;
- criteria may be fuzzy and/or numerical in one problem.

3.2. Soft Multi-Attribute Optimization

Now we present the second non-standard approach to poly-optimization, it is an intuitive decision making without formal criteria, a computer aided procedure – what we may call a soft optimization or constraints-based poly-optimization.

The premise is that in real decision situations there are many various requirements that the chosen variant must or should meet, of various natures: economic, ergonomic, technological, etc. Some are to be extreme (minimal or maximal), these are criteria, and some are limitations, those are constraints. In the mathematical formulation of the optimization problem they must be recognized and specified. However, most often a commissioner and decision-maker may have doubts about a membership and/or about an importance of a specific parameter on criteria or on constraints.

It is blurring the distinction between criteria (which are to be minimized) and constraints (which are to fall into limits). For example, when buying a bicycle, its cost may be a criterion, may be a constraint, or both.

3.2.1. Example 2

This example illustrates the usual complexity of relations and a conflict nature of criteria and constraints, what makes an intuitive holistic assessment rather difficult.

Fig. A

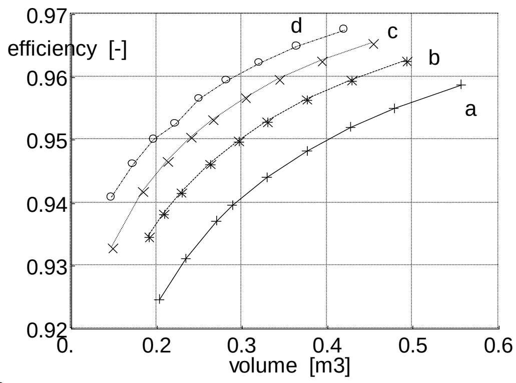

Fig. B

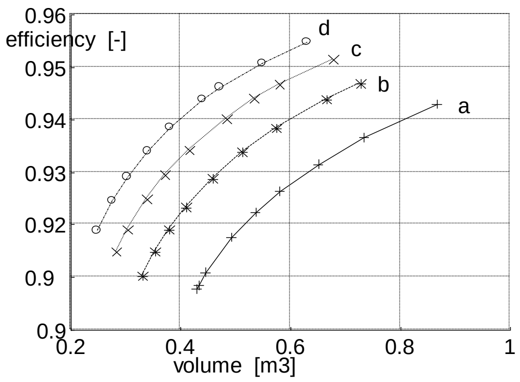

Fig. 8. The set of poly–optimal solutions in the criteria space (a goal space) for u_c= 20 (A) and u_c= 8 (B), for various materials: a) bronze (k_g=140 MPa, p_{dop}= 250 MPa;), b) steel (k_g= 160 MPa, p_{dop}= 350 MPa;), c) steel (k_g=180 MPa, p_{dop}= 90 MPa), d) steel (k_g= 250 MPa, p_{dop}= 780 MPa)

Fig. A

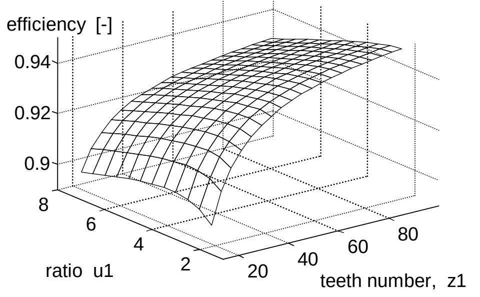

Fig. B

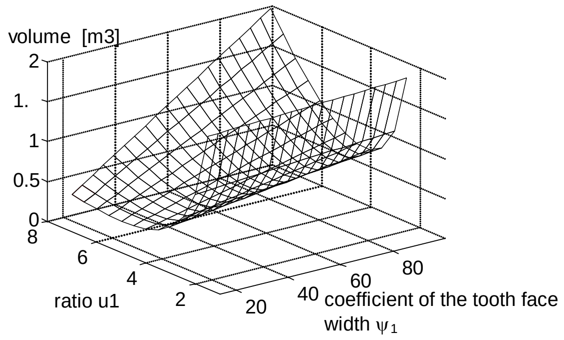

Fig. 9. Poly–optimal solutions in the decision space: the first criterion: efficiency (Fig. A) vs. u_1 (u_1 = (1...5, 7) and z_1 (z_1= (14, 92), and the second criterion (total volume) (Fig. B) vs. u_1 (u_1 = (1...5, 7) and the coefficient of the tooth face width Ψ_1 (Ψ_1 = (14, 92), for n_0=800 rpm, N_c=400 kW, Ψ_2=40, m_1=4 mm, m_2=8 mm, μ=0.01, $\alpha_0=20^0$, u_c=8

DOI 10.1515/ama-2017-0024

The object is a gear transmission. A poly–optimization has been completed and yields a numerous sets of Pareto solutions. But it is difficult for decision maker to grasp the play of constraints and to examine a conflict criteria situation, because there may be generated a great number of such diagrams. Exemplary Pareto solutions sets are presented in Fig. 8 and Fig. 9.

However, the more flexible approach exists, broadly known as a trial–and–error procedure, and the objective of this chapter is to show a hierarchical procedure - especially extensively aided by computer method – as a completely rational technique.

3.2.2. Problem Recognition

In most cases, the problem – which is a subject of the decision – is not quite new, e.g. the decision–maker has some (at least heuristic) knowledge of the field, so he/she may assess the quality of the specific variant. This evaluation may be done either on the scaled drawing displayed on a monitor or on printed charts (or diagrams) presenting performance characteristics of the variant.

Typically, there will be many variants, so the decision–maker will be searching for a compromise. If it is possible to express the problem mathematically, it could be defined as a poly–optimization or a multi–attribute decision problem.

3.2.3. Terminology

- $\mathrm{x}=[x_1,x_2,\dots,x_n]$ – vector of decision variables, it is a variant of solutions, defined in the decision variables space; $\mathrm{p}=[p_1,p_2,\dots,p_r]$ – set of constant-value parameters; $\mathrm{y}=[y_1,y_2,\dots,y_J]$ – set of performance variables; vector of performance parameters, it is a variant of solutions, defined in the performance (specifications) variables space; $\Omega_y=\{g_j(\mathrm{y},\mathrm{p})\le b_j j=1,\dots,J\}$ – set of functional constraints in the y space;
- $\Omega_x=\{g_j(\mathrm{x},\mathrm{p})\le b_j j=1,\dots,J\}$ – set of functional constraints in the x space;
- $D=[x_{ilower},x_{iupper}]\mathrm{i}=1,\dots,\mathrm{n}$ – set of lower and upper limits on decision variables. The decision–maker defines the limits by applying his experience, his/her common sense and obvious conditions (for example: dimensions must not be negative numbers).

The *permitted variant* (acceptable solution) x_{permit} is the one which satisfies inequalities Ω and membership relations D, i.e. meets all requirements and constraints specified in specifications.

The *satisfying variant* (or quasi-optimal) x_{satisf} is a permitted variant accepted arbitrarily by a decision-maker.

Generally, a decision–maker wants to identify the best solution. The “best” or *optimal* variant x_{opt} is such permitted variant *x*, which has an extreme value (maximal or minimal, according to the meaning) of the optimality function *F*:

$$\mathrm{x}_{opt}=\mathrm{x}\in D\cap\Omega:\left\{F(\mathrm{x})=\sup_{x\in D}F(\mathrm{x})\right\} \tag{3}$$

But what can be done if an optimality function F is not established and not defined? The general idea of the soft optimization is to generate (by a computer) as many solutions $\mathrm{x}^1,\mathrm{x}^2,\dots,\mathrm{x}^\mathrm{m}\in\mathrm{D}$ as possible, and then make a decision by a human expert. Leaving the final decision to the human being has a meaning of empowerment and privilege, as well as an obligation. To facilitate this arbitrary decision each variant should be presented in the most adequate way for the human decision–maker’s assessment, possibly graphically, and/or as a fuzzy numbers (values) set.

3.2.4. Problem Definition

It is assumed that a mathematical model of the object is given, as well as its all requirements and constraints. Besides, the decision-maker intuitively formulates some preferences and thus she/he is able to differentiate (evaluate) various submitted variants.

Let assume, that in a specific case a set of acceptable and permitted solutions exists. The proposed methodology requires only $<x,\Omega(x)>$, but an expert (an experienced decision–maker) is necessary to cooperate with the computer in a dialog mode.

A decision maker may rank requirements (constraints) according to their importance, the first criterion being crucial.

3.2.5. Outline of the Soft Optimization Idea

Decision-maker defines all constraints and applies computer to find any permitted solution and to visualize the variant and its characteristics. Then he/she arbitrarily change constraints and assesses a new variant, until he/she find a satisfying variant.

A special attention must be given to dynamic systems, when the mathematical model is a set of differential equations regarding the independent variables, which are time and three space dimensions.

3.2.6. Proposed Procedure

The overall idea is an interactive manipulation of threshold values of constraints, b_k’s.

Below a flow diagram of the proposed procedure is shown (Fig. 10). As it has been assumed, some actions are heuristic – not fully formalized - and a human actor completes them. Diamond shape figures show decision actions.

Step 1: If in **Step 6** a computer optimization procedure shall be used**,** set the optimality function $F=a$ to be constant (for example *a=0*), as contrary to the standard optimization procedure;
Step 2: For functional properties define reasonable upper values of $b_j j=1,\dots,J$ where b_j are upper boundaries b_j if the *“ the less b_j the better quality*” rule is applied;
Step 3: Choose one performance y_k parameter as a provisional criterion*: $k\in y$; k is taken from the set y.*
Probably you start with the constraint imposed on the most important performance parameter (a quality criterion), and *b* is the acceptable upper value, if *the less the better* (as the energy consumption, for example);
Step 4: Set the provisional value b_k; the chosen parameter must satisfy $k\le b_k$ condition;
Step 5: Set the initial vector of decision variables x_0 for the deepest gradient method of optimization. Start with $\mathrm{x}_0\in D$, unless you may define better starting point: $\mathrm{x}_0\in\Omega_x$. When the

Genetic Algorithm is used, initial population and other required parameters should be determined.

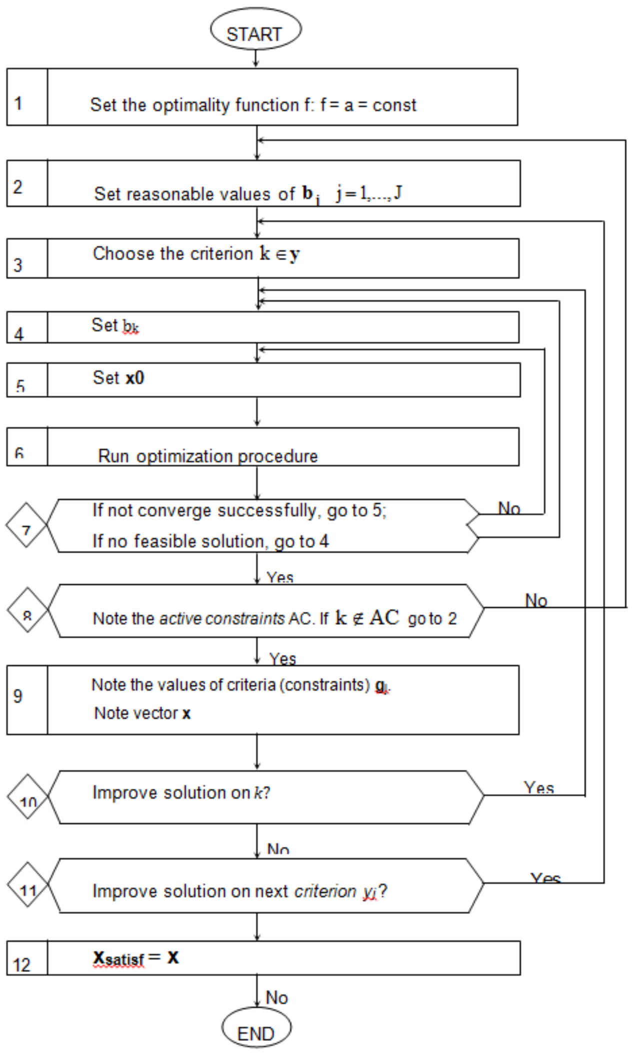
START
1 Set the optimality function f: f = a = const
2 Set reasonable values of $\mathbf{b}_j$ j=1,...,J
3 Choose the criterion $\mathbf{k} \in \mathbf{y}$
4 Set b_k
5 Set **x0**
6 Run optimization procedure
7 If not converge successfully, go to 5; If no feasible solution, go to 4
No
Yes
8 Note the *active constraints* AC. If $\mathbf{k} \notin \mathbf{AC}$ go to 2
No
Yes
9 Note the values of criteria (constraints) g_i. Note vector **x**
10 Improve solution on *k*?
Yes
No
11 Improve solution on next *criterion* y_i?
Yes
12 $\mathbf{x_{satisf}} = \mathbf{x}$
No
END

Fig. 10. Flow diagram of the 'soft' optimization algorithm

Step 6: Run optimization procedure, what is searching for *permitted variant* x_{permit}

For each constraint a penalty function is defined:

$$og_j = \begin{cases} 0 if\, g_j \le 0 \\ w_j \cdot g_j^2 if\, g_j > 0 \end{cases} \tag{4}$$

where w_j are arbitrarily chosen coefficients; typically $w_j\ \varepsilon = [0, .., 1]$ $j = 1, 2, \ldots J$ (for normalization purposes). Then any optimization procedure may be applied with the penalty function as the optimality function, where a known penalty function may be applied:

$$F = \sum_{j=1}^{J} og_j \tag{5}$$

The optimization is completed by a computer and may be accomplished by various techniques:

— The steepest gradient algorithm;
— Full survey of the decision space;
— Stochastic survey of the decision space;
— Finding and then applying an approximation function;
— Applying a trained artificial network;
— Applying the Genetic Algorithm;
— Typically it will be a gradient algorithm.

For example, if this approach is used in MATLAB, the *constr* or *fmincon* function can be used, where all constraints are defined and the value of optimality function is declared as an arbitrarily defined constant.

Observe how it proceeds; if not satisfactory, change the optimization procedure.

Decision 7: If the optimization procedure does not converge successfully, go to **STEP** 5 and change vector ***x0*;** if no feasible solution has been found, go to **STEP** 4 and set worse value (larger or smaller, what is the case) of b_k

Decision 8: Note the *active constraints* AC. If $k \notin AC$ go to 2

If the current constraint is not an active constraint *AC*, it may be the evidence that the threshold value b_k may be more restrictive – a better (smaller) value and/or other constraints are too restrictive (to "tight").

Step 9: Note the values of criteria (constraints) g_i; note the vector x; display the solution as a schematic drawing of the found object or as performance characteristics on the screen, to make easier assessment of the solution; then take the decision 10

Decision 10: Do you want to try further to improve the obtained solution against the current criterion k? If YES, lower the threshold b_k: go to 4 and set a smaller value, otherwise go to Decision 11

Decision 11: Now you may try to improve the obtained solution against another criterion k: $k \in y$. If YES go to 3 and take another crucial constraint, if NO, the procedure stops.

Step 12: Obtained solution x is the satisfying decision: *xsatisf*

3.2.7. Example 3 (Ociepa and Tarnowski, 2012)

The object is a servomechanism with two feedback loops, an outer loop from the position and the other one from the velocity signal, with two controllers. The design optimization problem is to determine seven decision variables (three parameters of each PID-type controller and the gain coefficient of a tachometric transducer). In the example constraints refer to characteristics of the step response: an overshot, the response time and the gain of a closed loop. There are few possible definitions of the optimality criterion: each of performance parameters could be adopted; these are the response time, a few standard integrals of error, the power consumption and others.

Following the proposed soft optimization algorithm, the response time was selected as the first constraint criterion, with the exemplary limit b_k = 60 s. In Fig. 10 exemplary step answers are depicted: by the proper optimality solution (Fig. 10A) and by a satisfactory solution (Fig. 10B) – soft optimization. The second characteristic may seem a bit worse (longer transient process and a small overshoot), but it requires smaller energy (see the smaller velocity pick) and was obtained in six times shorter computation time.

3.2.8. General Remarks on 'Soft' Optimization

The above given examples illustrate that there may be formulated various optimization problems, and various procedures may be applied to solve them.

There are several possible approaches to the 'soft' optimization. Typical ones are:

3.2.9. Searching for any Acceptable Solutions According only to the Given Constraints

Such approach may be justified by the fact, that typically the optimal solution is located on the edge of the acceptable solutions space ("on a constraint(s)"), although a question remains which one(s). Intuition and experience may suggest the limit: for example, in mechanical objects it is the stress/strain restriction. In MATLAB the function *constr* or *fmincons* is a comfortable technique, which finds the solution on constraint, near to the starting point, if a constant value of optimality function is declared. The human operator provides only the starting point.

Fig. A

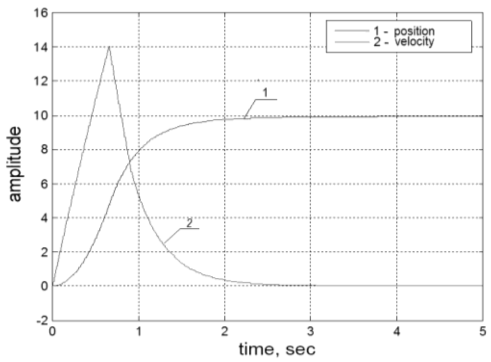

Fig. B

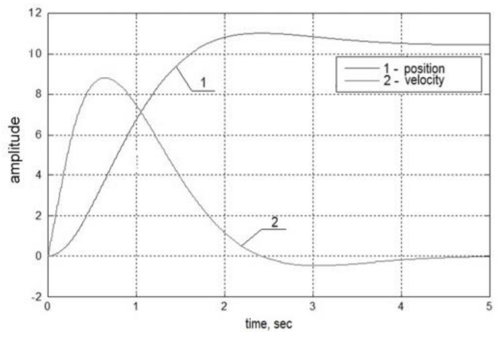

Fig. 10. Step answers of the servomechanism; A: the optimal on the ISE optimality function (computation time 4625s), B: 'soft' optimal (computation time 797s)

3.2.10. Trial–and–Error Procedure

The session is carried on by an operator and may be stopped at any point. The operator submits each probable solution (values of all decision variables) for verification.

3.2.11. Standard Analysis

An operator proposes a specific solution (values of all decision variables), and his/her computer calculates all programmed characteristics and/or a schematic view of the object. Example: a bridge or any other construction object – an architect is the operator and the computer completes computations of distribution of the stress/strain analysis; an automatic control system – the designer defines the structure and its parameters and the computer simulates its operation and displays characteristics.

In all cases an adequate mathematical model of the designed object is necessary. The resulted information about the decision (the selected variant i.e. the solution) should be displayed graphically on the monitor, to make an assessment easier for the human operator.

3.2.12. Conclusions about the Soft Poly-Optimization

A quasi–optimization (in the absence of optimality function) may be completed in a dialog procedure. The presented idea is based on the interactive mode of decision–making, which is a constitutive and inherent feature of the CAD systems. The proposed approach intends to encourage the interactive mode of the human–computer cooperation and make it more efficient. The quasi-optimal solution is achieved by tighten the constraints. The soft optimization may be especially efficient for objects with complex models: typically those containing non–linear partial differential equations, with many decision variables and numerous constraints.

The "soft" optimization creates a possibility for resolving non–unique inverse problems (Tarnowski and Krzyżyński, 2009).

The main advantages are: (1) no necessity to define an optimality function, (2) shorter computations, (3) more reliable procedure then the standard optimization.

The interactive (dialog) procedure for finding a satisfying solution may be recommended when:

- there are many challenging requirements and a decision maker cannot or denies to define one global optimality function which may represent a compromise;
- the mathematical model of the designed object is quite complex and difficult for computations, for example the model comprises non–linear differential (partial) equations; or there are many non–linear constraints and many decision variables. In such case the decision–maker may rather postpone formalizing and properly resolving a poly– optimization problem.

The proposed soft–optimization procedure reduces the computation time and does not require defining the optimality function. However, it needs a heuristic cooperation by a human operator, but yet this is also a feature of any other type of optimization.

Often problems with many decision variables and many functional constraints exist. Then optimization procedure is time consuming (for computations), especially when the model contains nonlinear differential partial equations, what is a rule for continuous time dependent systems.

3.2.13. Summary to the Soft Optimization

If a decision is subjected to many constraints, it may be disputable how to formalize an optimality function, and even multi–attribute approach is not necessarily the solution of the decision problem, because decision maker may not intuitively grasp the play of criteria.

In practical situations an optimality function may be difficult for standard optimization procedures (e.g. example of a pneumatic drive). What more, if the mathematical model contains non–linear partial differential equations, determining the Pareto solutions set may be a time–consuming process, and a decision maker may be discouraged to proceed. Then the dialog procedure of finding satisfactory solution may be recommended. In second part of this chapter, a kind of such interactive method called the "soft optimization" is proposed and presented.

4. Choice of the Minimization Algorithm

Classical gradient algorithms have limited applicability for continuous problems, with differentiability functions, with one minimum within the feasibility region. In the last decade a set of applicable algorithms has been substantially increased. Many have been programmed, coded and implemented, typically in MATLAB.

4.1. Poly-Optimization and Multi Attribute Optimization (MADM)

Typically a designer (or other decision-maker) seeks for a compromise between many requirements rather than to nominate one functional constraint to be an optimality criterion.

When a few criteria are to be adopted, there are possible two approaches:
- one is to build one artificial function of main constraints, and then to use it as the optimality function, what is usually called a multi-attribute optimization;
- and the other is to find a non-dominated variants and more-or-less arbitrarily to decide which single solution is to be decided what is called a poly-optimization.

The final result of poly-optimization is a set of non-dominated variants (decisions), then an arbitrary decision on one variant to further implementation must be taken by a human designer. This decision typically is made on heuristic premises, and is based on various paradigms, accordingly to specific believes of an individual decision-maker.

In Tarnowski (2011) there is a methodological discussion and many examples.

Besides, In Matlab there are two specific algorithms implementing the methodology: *fgoalattain* and f*minimaxi.*

4.2. Function Fgoalattain (MATLAB reference documentation)

It solves the multi-objective goal attainment optimization problem. Given are criteria, for each a preferable value is established (a vector goal) and its importance (weight).

A weighting vector controls the relative underattainment or overattainment of the objectives in fgoalattain. When the values of goal are all nonzero, to ensure the same percentage of under- or overattainment of the active objectives, set the weighting function to abs(goal). (The active objectives are the set of objectives that are barriers to further improvement of the goals at the solution).

4.3. Function Fminimax (MATLAB Reference Documentation)

The *Fminimax* Matlab function attempts to solve the minimax solution of several functions F_i and several variables x, with nonlinear constraints imposed on functions and variables:

$$\min_{x} \max_{i} F_i(x)$$

Criteria functions F_i and decision variables x can be vectors or matrices.

4.4. Particle Swarm Optimization (PSO)

In computer science, Particle Swarm Optimization (PSO) is a computational method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. It solves a problem by having a population of candidate solutions, here dubbed particles, and moving these particles around in the search-space according to simple mathematical formulae over the particle's position and velocity. Each particle's movement is influenced by its local best known position but, is also guided toward the best known positions in the search-space, which are updated as better positions are found by other particles. This is expected to move the swarm toward the best solutions [Wikipedia]. Examples are in Tarnowski (2011).

4.5. Analytic Hierarchy Process (AHP) (Saaty, 2008)

Procedure AHP is dedicated for a discrete set of solution variants and many criteria, when a decision maker rather prefers to answer a set of simple questions about his/her feelings rather than to define a strict value system e.g. an overall optimality function. It was developed by Thomas L. Saaty in the 1970s and has been extensively studied and refined since then. It was developed to optimize decision making when one is faced with a mix of qualitative, quantitative, and sometimes conflicting factors that are taken into consideration.

Tab. 1. AHP matrix of ranking of variants

Variant	A1	A2	…	Ai	…	An	S
A1	---						S_1
A2		---					S_2
…							
Aj				V_{ji}			S_j
…							
An						---	S_n

In the simplest case AHP may be used for ranking alternative design solutions. An empty square matrix $A = \{a_{ji}\},\ n\ x\ n$ of n alternatives is given. A decision maker puts his/her arbitrary opinion about a force of preference of the A_j over the A_i in an arbitrary scale, for example [-10, 10]. The sum S_j of partial marks V_{ji}: $S_j = \sum_{i=1}^{i=n} V_{ji}$ is a measure of value of the alternative A_j within the set of all alternatives A.

The advantage of this method is that no explicit criterion is necessary.

"Users of the AHP first decompose their decision problem into a hierarchy of more easily comprehended sub-problems, each of which can be analyzed independently. The elements of the hierarchy can relate to any aspect of the decision problem—tangible or intangible, carefully measured or roughly estimated, well or poorly understood—anything at all that applies to the decision at hand.

Once the hierarchy is built, the decision makers systematically evaluate its various elements by comparing them to each other two at a time, with respect to their impact on an element above

DOI 10.1515/ama-2017-0024

them in the hierarchy. In making the comparisons, the decision makers can use concrete data about the elements, but they typically use their judgments about the elements' relative meaning and importance. It is the essence of the AHP that human judgments, and not just the underlying information, can be used in performing the evaluations" [Wikipedia].

4.6. Dynamic Optimization (Dynamic Programming)

An optimal function is to be found in the time-space space (decision space or control function space decision) rather than a vector of numerical values of decision variables, like in the static optimization. The space may be continuous or discrete. By segmentation of a continuous coordinates we may transform a problem to easier formulation and then solve it by standard static optimization methods in recurrent procedure.

There are two approaches to the dynamic optimization. A classic one based on the Pontryagin's maximum principle yields an optimal solution as a continues function in the time-space space. A stress-strain beam example by L. Mikulski is given in Tarnowski (2011).

If a segmentation of the time-space space is possible, another group of simplified methods are at hand.

4.7. Bellman–Ford Algorithm

A segmentation of the decision space is necessary. The Bellman–Ford algorithm is an algorithm that computes shortest paths from a single source vertex to all of the other vertices in a weighted digraph [Wikipedia]. It is based on the principle of relaxation, in which an approximation to the correct distance is gradually replaced by more accurate values until eventually reaching the optimum solution [Wikipedia]. As an example is a long distance optimal route of an ocean motor-sail ship (in Tarnowski (2011)).

4.8. Transformation to Static Optimization

All coordinates of the time-space space are segmented, and the Cartesian product is a discrete set of variants. Then the static optimization may be completed. As an example is a problem of an optimal car control (Tarnowski, 2009).

Two optimal functions are found: the driving force and the breaking force between two points to get a minimal time and minimal power. Exemplary results of conflicted two-criteria are depicted on Fig. 12.

5. CONCLUDING REMARKS

Non-linearity of mathematical models, non-continuity of variables, fuzziness of parameters, relations and preferences, variety and conflicted preferences and objectives – these are reasons of difficulties of optimization, especially in mechatronic design.

Inverted Problem approach is the way to build the design procedure.

Variety of approaches to optimization and a broad spectrum of methods may discourage designers and the decision makers, but new nonconventional approaches and procedures may be a medium towards overcoming the discrepancy between real practice and socio-economical requirements.

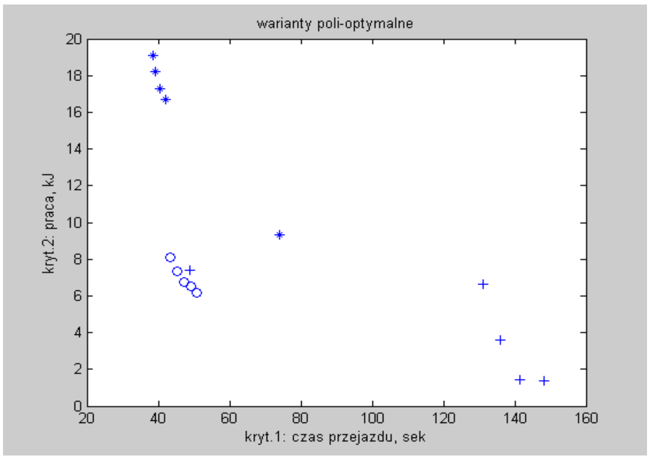

Fig. 12. Discrete set of the found compromise solutions, in the criteria space; kryt 1: time of the passage [sec]; kryt 2: used power for passing the car [kJ]: (+) after 65 iterations (generations) of a genetic algorithm, (*) after 130 and (o) after 200 iterations

Mathematical models in Mechatronics are exceptionally difficult for computer analysis, thus non-conventional algorithms must be used (evolutionary ones, for example), and a non-traditional formulation is necessary (e.g. fuzzy). Also, the definition of criteria in a concrete case may be questionable, so the MADM formulation may be the answer.

Presently new methods are being elaborated and implemented, in two related domains:

- algorithms for minimization (for example: Długosz (2013)), where evolutionary based methods for multi-scale are proposed), and
- novel approach for estimation of compromise solutions set (for example [13], where New measures are proposed for evaluation a set of poly-optimal solutions, like a distance measure to the ideal solution combined with a nearest solution as well as a ranking index and a dominance numerator within the set, what is a ratio of a dominant variant to all admissible variants.

Due to specific features of Mechatronic Design the optimization is of a special meaning in Mechatronic Engineering Design. The main advantage of the poly-optimization is a possibility of an 'in-side' to the problem of a conflict nature between requirements imposed on the design. Especially, it is possible to quantitatively understand a play of conditions and to estimate the sensitivity of influence of one criterion to the other one, and to find a rational compromise. This is particularly important in mechatronics, where the human intuition and experience may be unreliable, due to the short design tradition and to the complexity of co-operating various physical processes.

6. SUMMARY

On the ground of the Design Methodology it was proved that each design task is an inverse problem. As such, optimality approach is a proper methodology to solve it. What more, optimization may be applied as a methodological paradigm to build CAD/CAM systems.

DOI 10.1515/ama-2017-0024 acta mechanica et automatica, vol.11 no.2 (2017)

Reasons that optimization in design is of a crucial importance, specifically in mechatronics are discussed.

Typical classes of problems are specified.

The typical problems of optimization in design in mechatronics are presented. And it has been proposed – or suggested – the set of adequate methods.

7. FINAL NOTES

The monograph "Optimization and Poly-optimization in Engineering" by W. Tarnowski (2011) may be recommended: it aids to formulate mathematically problems and comprises methods, illustrated on numerous examples. The poly-optimization theory and applications you may find there, also.

For MATLAB implementation look the instructive reference manual book.

REFERENCES

1. **Długosz A.** (2013), *Multicriteria optimization in conjugated field problems* (in Polish), Wydawnictwo Politechniki Śląskiej, Gliwice.
2. MATLAB reference documentation
3. **Ociepa Z., Tarnowski W.** (2003), *Parametric Synthesis of Discrete Controllers Aided by Optimization Computer Procedures*, International Carpathian Control Conference ICCC'2003,Tatrzańska Łomnica, Slovak Republic, 532 – 535.
4. **Saaty T.L.** (2008), Decision making with the analytic hierarchy process, *Int. J. Services Sciences*, 1(1), 83-98.
5. **Tarnowski W.** (1997), *Foundations of the Engineering Design* (in Polish), WNT, Warszawa.
6. **Tarnowski W.** (2009), Simultaneous optimization of a machine and a process (in Polish), *Problemy Eksploatacji*: reprint from: *Bioagrotechnical Systems Engineering*, 2, 17-34.
7. **Tarnowski W.** (2011), *Optimization and Poly-optimization in Engineering* (in Polish), Wydawn. Pol. Koszalińskiej, Koszalin.
8. **Tarnowski W.** (2015), *Fuzzy and Soft Poly-Optimization in the Digital Environment – Examples*, chapter 9 in Fuzzy Optimization and Multi-Criteria Decision Making in Digital Marketing, red. A Kumar, IGI Global, Hershey, Pennsylvania USA, 180-200.
9. **Tarnowski W. Krzyzynski T., Maciejewski I., Oleskiewicz R.** (2009), Poly-optimization - a paradigm in engineering design in mechatronics, Archive of Applied Mechanics, *Archive of Applied Mechanics*, 81(2), 141-156.
10. **Venkataraman P.** (2009), *Applied optimization with Matlab programming*, John Wiley & Sons.
11. **Yager R.R.** (1993), *Fuzzy sets and systems*, Elsevier.